Classifying the suras by their lexical semantics: an exploratory multivariate analysis approach to understanding the Qur'an

By

Naglaa Ahmed Thabet

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Linguistics

School of English Literature, Language, and Linguistics
University of Newcastle upon Tyne

July 2006

Contents

List of Figures

List of Tables

Acknowledgements

The writing of a dissertation can be a lonely and isolating experience, yet it is obviously not possible without the personal and academic support of numerous people. The pages of this dissertation hold far more than the culmination of years of study. They also reflect the relationships with many generous and inspiring people whom I have met since beginning my graduate work. It is good to look back and acknowledge the team of people that had helped you to accomplish this goal. Those people who know how tremendous their help has been will also know how heartfelt and sincere my gratitude is. To them all a huge '*thank you*' is owed.

Many thanks and appreciation to my supervisor, Dr. Hermann Moisl, who is most responsible for helping me complete the writing of this dissertation as well as the challenging research that lies behind it. He gave willingly all the time, effort and expertise needed for the completion of this work which could not have been materialised without his excellent guidance, patience and support.

Special thanks and gratitude to my dear parents for their love, unconditional support and encouragement over the years, and for their unwavering faith and confidence in me and in my abilities. To my father, Dr. Ahmed Thabet for being my role model academically and personally, for his continuous interest, valuable discussions and insightful comments about my work. I'm no less indebted to my loving mother without whose prayers, love and support I could not have achieved what I have today.

To my invaluable network of supportive, forgiving, generous and loving friends without whom I could not have survived the process. To Neveen and Sameh Shaaban, Dina and Wael Nabeeh, Heba and Mohammed Loutfi, Safaa and Mohammed Shaaban, Entesar and Abdulrazzaq Naas, Irena and Kaycey Ihemere, Peter and Theodora Adegbie, Fahimeh Naseri and David Young, I thank them all for giving me their friendship, as deep and as rich as friendship can be.

Special thanks are due to Mona and Gasser El-Bishry and their family for opening their hearts and home to me and my little ones, for being there whenever I needed their help and for the cheerful times we spent together. Their friendship is invaluable.

Special thanks are due to my close companion Rasha El-Ibiary who although younger than me has been like a caring mother, for all her love, support and constant encouragement, for believing in me and for the many memories along the way, I am deeply grateful.

Special thanks are due to my dear friend Niveen Kassem for keeping my company while writing, for our endless fun at stressful times, for being supportive and caring, for help with proofreading and revisions of my work. In her I have a long-life friend and colleague.

Special thanks to Ali Al-Laham for being there whenever I needed his help and for help with proofreading the dissertation.

Last but surely the most, I would like to express my deep appreciation and love for my husband and closest companion, Hisham Bakr, who has suffered for this dissertation more than anyone else and has set aside his own career to accompany me as I wrote up this dissertation. He has endured with selflessness every emotion, all the fears and tears, and the countless hours of my solitude and detachment. I cannot imagine that I could have completed this work without his encouragement, care and support. I also have to thank my wonderful children, Nouran and Ahmed, not only for being a source of inspiration for me, but also for being so patient with me and for giving up much of our precious time together until I completed this work.

Abstract

The Qur'an is at the heart of Islamic culture. Careful, well-informed interpretation of it is fundamental both to the faith of millions of Muslims throughout the world, and also to the non-Islamic world's understanding of their religion. There is a long and venerable tradition of Qur'anic interpretation, and it has necessarily been based on literary-historical methods for exegesis of hand-written and printed text. Developments in electronic text representation and analysis since the second half of the twentieth century now offer the opportunity to supplement traditional techniques by applying the newly-emergent computational technology of exploratory multivariate analysis to interpretation of the Qur'an. The general aim of the present discussion is to take up that opportunity.

Specifically, the discussion develops and applies a methodology for discovering the thematic structure of the Qur'an based on a fundamental idea in a range of computationally oriented disciplines: that, with respect to some collection of texts, the lexical frequency profiles of the individual texts are a good indicator of their semantic content, and thus provide a reliable criterion for their conceptual categorization relative to one another. This idea is applied to the discovery of thematic interrelationships among the *suras* that constitute the Qur'an by abstracting lexical frequency data from them and then analyzing that data using exploratory multivariate methods in the hope that this will generate hypotheses about the thematic structure of the Qur'an.

The discussion is in eight main parts. The first part introduces the discussion. The second gives an overview of the structure and thematic content of the Qur'an and of the tradition of Qur'anic scholarship devoted to its interpretation. The third part

defines the research question to be addressed together with a methodology for doing so. The fourth reviews the existing literature on the research question. The fifth outlines general principles of data creation and applies them to creation of the data on which the analysis of the Qur'an in this study is based. The sixth outlines general principles of exploratory multivariate analysis, describes in detail the analytical methods selected for use, and applies them to the data created in part five. The seventh part interprets the results of the analyses conducted in part six with reference to the existing results in Qur'anic interpretation described in part two. And, finally, the eighth part draws conclusions relative to the research question and identifies directions along which the work presented in this study can be developed.

Chapter One

Introduction

The Qur'an is at the heart of Islamic culture. Careful, well-informed interpretation of it is fundamental both to the faith of millions of Muslims throughout the world, and also to the non-Islamic world's understanding of their religion. There is a long and venerable tradition of Qur'anic interpretation, and it has necessarily been based on literary-historical methods for exegesis of hand-written and printed text. Developments in electronic text representation and analysis since the second half of the twentieth century now offer the opportunity to supplement traditional techniques by applying the newly-emergent computational technology of exploratory multivariate analysis to the interpretation of the Qur'an. The general aim of the present discussion is to take up that opportunity.

The discussion is in eight main parts. The first part introduces the discussion. The second gives an overview of the structure and thematic content of the Qur'an and of the tradition of Qur'anic scholarship devoted to its interpretation. The third part defines the research question to be addressed together with a methodology for doing so. The fourth reviews the existing literature on the research question. The fifth outlines general principles of data creation and applies them to creation of the data on which the analysis of the Qur'an in this study is based. The sixth outlines general principles of exploratory multivariate analysis, describes in detail the analytical methods selected for use, and applies them to the data created in part five. The seventh part interprets the results of the analyses conducted in part six with reference to the existing results in Qur'anic interpretation described in part two. And, finally, the eighth part draws conclusions relative to the research question and identifies directions along which the work presented in this study can be developed.

Chapter Two

The Qur'anic interpretative tradition

2.0 Introduction

The Qur'an occupies an important position amongst the greatest religious books of the world and represents the primary source of every Muslim's faith and practice. By Muslims it is regarded as a revelation from God. It is used in their public and private devotions, and is recited at festivals and family occasions. It is the basis of their religious beliefs, their ritual and law, the guide of their conduct, both public and private. It moulds their thought, and its phrases enter into literature and daily speech. It demands serious study, for it is by no means an easy book to understand and appreciate. The language, style, and structure of the Qur'an cause special problems not only for the Western reader, who must rely on a translation, but also for Muslims, Arab and non-Arab alike. It can be fully appreciated only in Arabic, and yet it contains foreign words, Arabic words with foreign meanings, words with two or more completely different meanings or a general and specific meaning that cannot always be distinguished, and words whose meanings have changed in Arabic usage over the fourteen centuries since the time of Mohammed. It is a religious document containing the initial statements on the main Muslim beliefs and practices. It is a document of sacred law, with regulations on marriage, divorce, food, finance, contracts, wills, bequests, etc. Its contents are not confined to a particular theme or style, but cover a whole spectrum of issues, which range from specific articles of faith and commandments to general moral teachings, rights and obligations, crime and punishment, personal and public law, and a host of other private and social concerns. These issues are discussed in a variety of ways, such as direct stipulations, reminders of Allah's favours on His creation,

admonition and rebukes and stories of past communities followed by lessons to be learnt from their actions and subsequent fates. It is scripture with some of the same characteristics and functions of the other scriptures, and it is literature, the first written book in Arabic and in some ways the progenitor of Classical Arabic prose. Yet it does not conform to the style and literary forms of the other scriptures and it cannot be judged by the standards of other literatures. The problems involved in gaining an appreciative understanding of the Qur'an are numerous and complex and critical opinion on the basic issues is more divided now than ever before. So the quest for sound knowledge of this unique book must continue.

2.1 History and structure

Muslims believe that the Qur'an is the last Book of Allah sent for the guidance of humanity through the last Prophet, Mohammed, via the Angel Gabriel. Beginning in 610 CE, they believe it has been revealed piecemeal throughout a period of about 23 years until shortly before the Prophet's death in 632 CE in two phases: the period in Mecca, before the *hijra* (Mohammed's migration to Medina, 622 CE) and the period in Medina, after the *hijra*. That the verses of the Qur'an were not revealed in one place but rather in stages over a period of 23 years during the Prophet's mission is authenticated not only by historical evidence but also from evidence from the Qur'an itself: "A Qur'an We have distinctly separated its verses, that you may read it to mankind at intervals, and We have been sending it down successively[1]" (*Sura Al-Israa*:106). The contents of the chapters and verses are directly related to the events, circumstances and different needs of the period of the Prophet's mission. Because the Prophet's prime task in the earlier Meccan phase was to call the people to Islam, the main themes of the revelations in this phase are belief in Allah and His unity (*tawhid*), the coming resurrection, the day of Judgement, reward and

[1] i.e. by successive revelations on different occasions.

punishment, the prophethood of Mohammed and stories of the past prophets and their nations who defied the message of their prophets. In the Medinan phase, the basic Meccan themes remain but are augmented by themes relating to Muslims growing together into a community, the *umma*. The Medinan *suras* are concerned mainly with ritualistic aspects of Islam, lay down moral and ethical codes, criminal laws, social, economic and state policies, and give guidelines for foreign relations and regulations for battles and captives of war. The dating does not go beyond the bare description of the *sura* as Meccan, or Medinan; and these descriptions do not necessarily apply to the *sura* as a whole. Muslim scholars have always been ready to admit that *suras* are composite, and that one marked as Meccan may contain one or more Medinan passages, and vice versa. These descriptions, then, are to be regarded merely as the judgements of the compilers, or of early scholars, about the period at which the main content of each *sura* was revealed.

Because Mohammed could neither read nor write, it is believed that he recited what was revealed by Gabriel, and that his companions either memorized or wrote down what was said, so that in the Prophet's own time his revelations circulated in both written and oral form. In the period of his rule from 644 - 656 CE, the Umayyad caliph Uthman ibn Affan headed a committee that gathered these materials and for the first time established a canonical text of the Qur'an. This remains the standard text to the present day, and Muslims believe that it exactly represents the revelations of Allah to Mohammed.

Structurally, the text of the Qur'an is arranged in 114 chapters called *suras*. These are unequal in length, some being several pages long while others comprise only a few lines. The chapters are not arranged in a way that reflects the order of the revelation. In fact, they seem to be in roughly the opposite of the chronological order, and appear to be arranged by length, going from the longest to the shortest. The longest *sura* is number 2 '*Al-Baqarah*'

which consists of 286 verses and the shortest is *sura* number 108 '*Al-Kawthar'* which consists of 3 verses. *Suras* are traditionally identified by their names rather than their numbers. The principal function of the *sura* name is to provide convenient labels for identifying the *suras* and distinguishing them from one another. These names are normally distinctive or unusual words that appear somewhere in the early part of the *sura*, for example *Al-Qamar* ('The Moon'), *Al-Teen* ('The Fig'), *Al-Falaq* ('Daybreak'), and *Al-Balad* ('The Land'). In other instances, the name is a word which occurs elsewhere in the *sura*, but is rare in the Qur'an as a whole. This is the case with *sura Al-Baqarah* ('The Cow') and *Al-Nahl* ('The Bee'). In a few other instances, the name does give some indication of a *sura*'s contents: for example *Al-Nisa* ('Women') which deals with legislation concerning women. *Suras* which bear the names of prophets such as *Ibrahim, Nuh, Yunus* and *Hud* usually mention stories of a few other prophets as well. The only exception is *sura Yusuf* which is almost entirely devoted to the story of Joseph. The *suras* are further subdivided into verses called *ayat* (sing. *ayah*). The length of these *ayat* varies notably from *sura* to *sura*; in general, the earlier Meccan *suras* tend to have shorter verses than the later Medinan ones, with legal verses being particularly long. Twenty-nine of the chapters begin with seemingly disjointed letters which are referred to as the "mysterious letters", which may convey some secret religious meaning, or may just signify a filing system for organizing the Qur'an. However, there is no unanimous agreement among Muslim exegetes as to what their exact meaning is.

2.2 Qur'anic studies

That the Qur'an needed explanation and interpretation was recognised by Muslims right from the start, and over the fourteen centuries of the Islamic era a very large body of commentary has been produced. For present purposes it seems best to regard these

Qur'anic studies as a broad field covering four main areas: (1) exegesis (*tafsir*), or the study of the meaning of the text and the history of its interpretation, (2) the roles of the Qur'an in Muslim life and thought (in ritual, theology, etc), (3) the translation of the Qur'an, and (4) the linguistic significance of the Qur'an.

2.2.1 Exegesis

The literal meaning of *tafsir* in Arabic is to explain, interpret or comment, and thus, the science of *tafsir* is a branch of knowledge in which the meanings of the Qur'an are explained and interpreted. This task of interpretation was initiated by Mohammed who was asked to recite and explain the teachings of the Qur'an to his people. Addressing Mohammed in *sura Al-Nahl*, Allah says: 'We have sent down to you the Remembrance that you may make evident to mankind what has been sent down (ever since) and that possibly they would meditate' (16:44). The Qur'an also says in *sura Al-Imran*: 'Allah has indeed readily been bounteous to the believers as He sent forth among them a Messenger from among themselves, who recites to them His signs and cleanses them, and teaches them the Book and Wisdom' (3:164). This role of interpreting the Qur'an was taken over by Mohammed's companions after his death, and since then it became necessary that the *tafsir* of the Qur'an be preserved as permanent branch of knowledge so that, along with the words of the Qur'an, its correct meaning as well stands protected and preserved for the Muslim community. The area of *tafsir* is thus mainly concerned with the study of ways the text has been interpreted to meet the needs and concerns of generations of Muslims. Its goal is the critical analysis and interpretation of the text including such basic issues as its composition, dating, history, style, literary forms, grammar, etc. and the quest for ever increasing knowledge and understanding of its meaning, beginning with the question of what it most likely meant to its first hearers. Each generation of Muslims since the time of

Mohammed has produced its own exegesis and commentaries on the Qur'an. Abdul-Roaf (2001a) divides them into six main categories: (1) *Linguistic Exegesis* concerned with the grammar, stylistic analysis, and rhetoric of Qur'anic discourse, (2) *Philosophical and Rationalistic Exegesis* concerned with explaining and refuting philosophers' views and arguments, (3) *Historical Exegesis* concerned with Qur'anic parables and the history of nations and peoples mentioned in the Qur'an, (4) *Intertextual Exegesis* concerned with interpreting the Qur'an through the Qur'an itself or the *Hadith* (the sayings of the Prophet Mohammed), (5) *Jurisprudence Exegesis* concerned with jurisprudence matters and the different views of Muslim theologians, and (6) *Independent Judgement Exegesis* concerned with interpreting the Qur'an according to one's own judgement and personal point of view. These commentaries and other works are not all of equal value. The most important ones for critical historical study today are those written during the "classical" and "modern" periods, i.e., during the first six centuries after the death of the Prophet and the last two centuries. The classical commentaries contain a wealth of information, including detailed grammatical analyses, explanations of different passages, obscure terms, and vague allusions, and a variety of accounts giving the 'occasions of revelation' or filling in gaps in certain Qur'anic stories. Several commentaries of the Qur'an have been written since the period of Mohammed's Prophethood. Among the most prominent classical exegetes are: Al-Tabari (1968), Al-Baghawi (1985), Al-Zamakhshari (1995), Ibn Attiyah (1977), Ibn Al-Jawzi (1964), Al-Qurtubi (1997), Ibn Kathir (1970), Al-Razi (1981), Al-Andalusi (1993), Al-Mahalli & Al-Suyuti (1989). Modern exegetes include: Al-Alusi (1855), Qutub (1982), Maududi (1988) and Al-Ghazali (2000).

2.2.2 The role of the Qur'an in Muslim life

The focus of the great majority of the second area of studies has been the theological and legislative aspects of the Holy Book, for the Qur'an provides Muslims with detailed guidance on their everyday lives. Together with the sayings, actions, and recommendations of Mohammed, the Qur'an has been the ultimate source of legal authority for Muslims over the past fourteen centuries. Muslim scholars have painstakingly examined, analyzed and interpreted the various verses of the Holy Book; detailing the requirements the Qur'an imposes on Muslims in order for them to achieve spiritual purity. Thus, in addition to its legislative and theological value, the Qur'an has also served as a source of spiritual guidance for the followers of Islam. Several studies focused on the role of the Qur'an and its impact and influence on the life of Muslims. Abdul Haleem (1999) gives a general introduction to the importance of the Qur'an in the life of Muslims with specific reference to rules of marriage and divorce and the laws regulating war and peace. Malek (1997) focuses on different aspects of Muslims' life with respect to Qur'anic teachings such as adultery, women's rights, divorce, polygamy, the veil and gambling. According to Nanji (1991) the Muslim *ummah* or community is seen as the instrument through which Qur'anic ideals and commands are translated at the social level. He makes clear that the Qur'an is not all about the relationship with God but also its role in building a moral Muslim community. Nanji (1991,108) adds that:

'The Qur'an affirms the dual dimension of human and social life- material and spiritual – but these aspects are not seen in conflictual terms, nor is it assumed that spiritual goals should predominate in a way that devalues material aspects of life. The Qur'an, recognising the complimentarily between the two, asserts that human conduct and aspirations have relevance as acts of faith within the wider

human, social and culture contexts. It is in this sense that the idea that Islam embodies a total way of life can best be understood.'

2.2.3 Translation of the Qur'an

The third area is concerned with the translations of the Qur'an. Since the first century of *hijra*, continuous efforts have been made by Muslims to make the meanings of the Qur'an accessible to non-Arab communities. Despite the enormous amount of effort that has been made during the past centuries, and notwithstanding the numerous translations that have appeared in many languages of the world and the new ones that continue to appear almost every year, an error-free translation of the Qur'an still remains an unfulfilled dream. The Qur'an translator does not only need a sound linguistic competence in both Classical Arabic and the target language but also an advanced knowledge of Arabic syntax and rhetoric and most importantly familiarity with the tradition of the past 14 centuries and the works of major exegetes in order to derive and provide the accurate underlying meaning of a given Qur'anic expression. It is an inexhaustible source of meaning, meeting the needs of changing times, though its words and diction remain fixed and unchangeable. Thus the human effort to extract its meanings is an unending task of interpreting and understanding. In every era efforts have to be made to translate the Book in the language of the day and to interpret it in the light of accumulated human knowledge and Qur'anic scholarship. In view of all this, there is the need to develop a sound and systematic methodology for translating and interpreting the Qur'an in different languages of humankind, for to translate the Qur'an is to interpret it. The translation, however, should not be looked at as a replacement of the original version of the Qur'an in Arabic but rather as *interpretative* translation. Abdul-Raof (2001a,1) comments on the translatability of the Qur'an:

'Qur'anic expressions and structures are Qur'an-bound and cannot be reproduced in an equivalent manner to the original in terms of structure, mystical effect on the reader, and intentionality of source text. Inaccuracies and skewing of sensitive Qur'anic information will always be the by-product of any Qur'an translation...The translation process has to be based on the fact that the output will be an interpretation of the underlying meanings of the Qur'an rather than a substitution for the original text.'

Abdul-Roaf (2001a) identifies two types of Qur'an translation. The first type is semantic translation which also adopts archaic language and some literal word order such as the translations by Ali (1983), Arberry (1980), Asad (1980), Bell (1937), and Pickthall (1969). These translations 'allow the source language to have dominance over the target language' (Welch 1990, 272). The second type is that which provides a communicative translation and introduces the Qur'an in contemporary English such as translations by Akbar (1978), Irving (1979), and Turner (1997).

Several organisations and companies have been established in recent years dealing with Qur'anic translation studies. Their main goal is to publish reliable translations of the Qur'an in various languages so that a unique world collection of the translations of the Qur'an, complete and partial, accessible to scholars and researchers throughout the world, can be established. Software companies also plan to make Qur'anic translations available electronically for the use of researchers and scholars for future work. Translating the Qur'an is a huge task and there has been a magnificent achievement with the contribution of existing translations. However, Qur'anic scholars and experts throughout the world are still called upon to assist it in the task of precise evaluation of the existing translations in

different languages for the purpose of making the Qur'an and its translations available to Muslim communities throughout the world.

2.2.4 The linguistic significance of the Qur'an

The fourth area, which has received far less attention than the other aspects of the Qur'an, is its linguistic significance. A few studies have focused on the impact of the revelation on the form and content of the Arabic language. Omran (1988) highlights the importance of the Arabic language as an effective medium for the communication of the message of Islam and how it served as a means for preserving the cultural and religious heritage of Arabic-speaking and Muslim people. He also argues that while the Arabic language played an important role as the medium for the revelation of the Qur'an, the language benefited enormously from its association with Islam and the Qur'an. The need to preserve the accuracy and pronunciation of the Qur'an demanded efforts in refining the Arabic alphabet and grammar. Daily worship and recitation of the Qur'an also helped in maintaining and keeping the language alive. He also comments on the great impact the Qur'an has on the enhancement of the beauty of the Arabic language, by introducing new styles, forms of expression, figures of speech, rhythmic patterns, and structures. Moreover, the use of narratives in the Qur'an remains one of the most creative and innovative aspects which has profoundly enriched the Arabic language. There are several other works focusing on the linguistic aspects of the Qur'an and how they were used in illustrating and underlining important aspects of the Qur'anic message. Abdul-Raof (2003) presents a detailed description of the various linguistic aspects and features such as consonance, presentation techniques, lexical cohesion, parallelistic structures, syntactic chunking, intertexuality, the use of rhetoric and other phonetic features. Watt & Bell (1970) comment on some of the linguistic features of the Qur'an with specific reference to the use of dramatic structures,

rhymes and strophes, and various didactic forms. Abdul-Roaf (2001b) examines the prototypical linguistic, phonetic, prosodic, and rhetorical features of Qur'anic discourse. He also focuses on Qur'anic texture and the unique cohesion system and its textural constituents employed in Qur'anic discourse. Robinson (1996) comments on the morphology, structure and coherence of the Qur'an with specific reference to the role of sound and intertextuality and the dynamics of Qur'anic discourse. El-Awa (2002) studies textual relations in the Qur'an and the mechanisms and dynamics of the Qur'anic text structure with detailed analysis of two Qur'anic passages.

Chapter Three

Research question and methodology

3.1. Research question

As already noted, the *suras* of the Qur'an are not arranged in a way that reflects the order of revelation, and, to the naive reader, there is no obvious reason why the *suras* are sequenced as they are in the text. They are not in chronological order of the Prophet's life, for example, nor is there any clear thematic arrangement. Figure 1 plots the 114 *suras* in order of their occurrence in the Qur'an on the x-axis against *sura* size in kilobytes on the y-axis, and shows that, apart from *sura* 1, *Al Fatiha*, which is invocatory and very short, they are in fact ordered roughly by length.

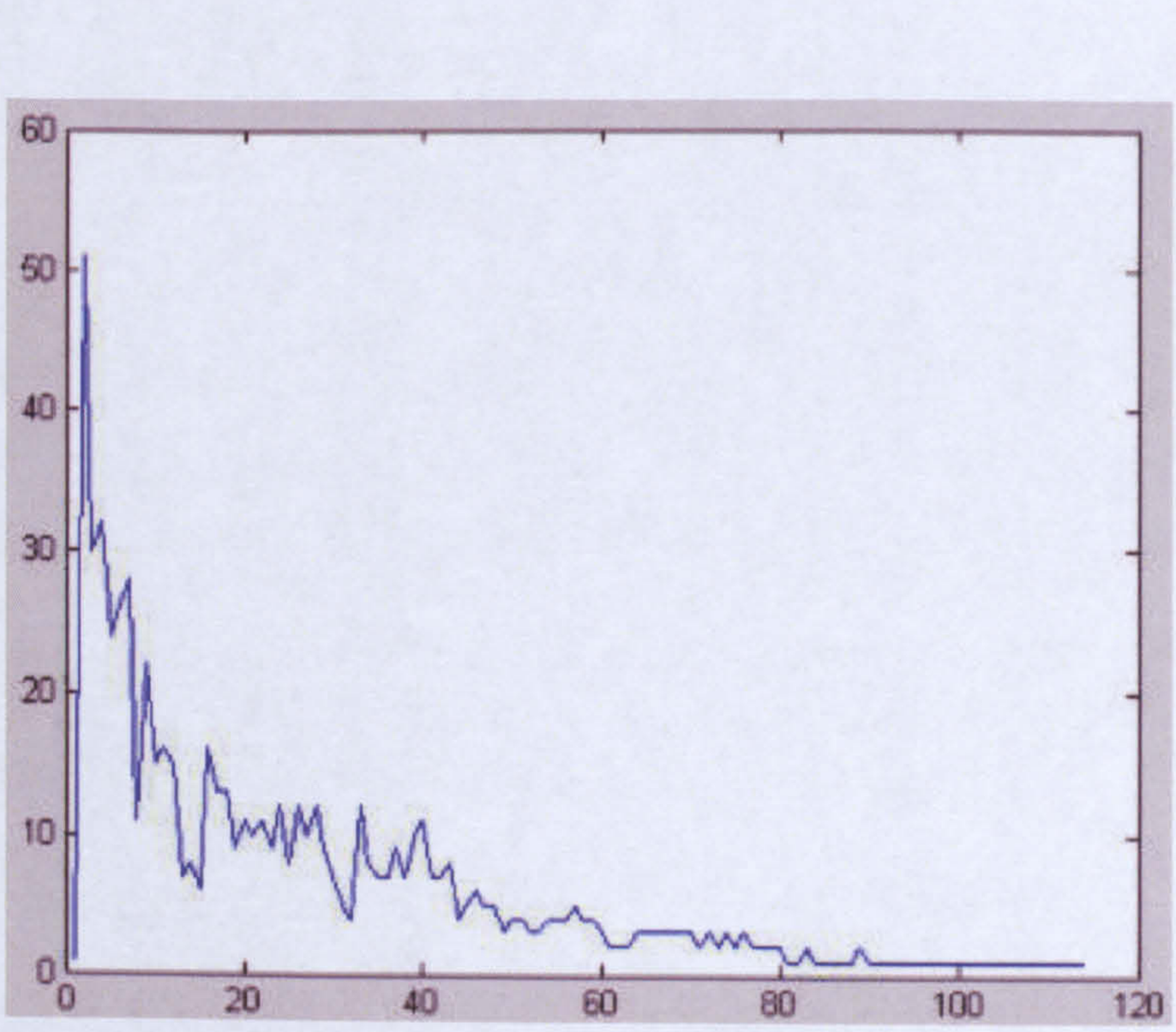


Figure 1: Plot of *sura* lengths

Given this apparently arbitrary sequencing, the key to interpretation of the Qur'an must be to understand the thematic relationships that link the *suras* to one another in a conceptually

coherent way. The Qur'anic scholarly tradition has long been devoted to such understanding by careful cross-referencing of thematic elements across the *suras*; the general aim of the present discussion is to see whether current exploratory multivariate analytical methods can usefully contribute to this task. More specifically, the aim is to see whether the *suras* can be classified in a thematically coherent way on the basis of their semantic content.

3.2. Methodology

The advent and proliferation of digital technology since the second half of the twentieth century has generated an explosive production of electronically-encoded text of all kinds, most visibly in the now-billions of documents accessible via the World Wide Web. The situation at present is such that traditional, paper-based methods for representation, storage, search, and interpretation of text have rapidly been overwhelmed by sheer volume, and a range of technologies have been and continue to be developed in order to make the deluge at least tractable.

Computational implementation of existing and newly-developed exploratory multivariate analysis (EVA) techniques has emerged as a key technology in this (Tukey 1977; Kachigan 1991; Grimm & Yarnold 1995; Hair *et al.* 1998; Grimm & Yarnold 2000; Tinsley & Brown 2000; Everitt & Dunn 2001; Tabachnik & Fidell 2001). EVA aims to discover regularities in data that can serve as a basis for formulation of hypotheses about the domain from which the data comes, and more specifically offers a wide range of mathematically and statistically based computational methods for discovering classes in data and for identifying any interestingly-structured relationships among such classes. The research question is addressed by applying a selection of these methods to data abstracted from the Qur'an.

Another key technology in dealing with electronic text is Information Retrieval (IR), which is largely concerned with design and implementation of computational systems that, with respect to some given document collection, attempt to select from the collection those documents that are most relevant to a specific user query (van Rijsbergen 1979; Salton & McGill 1983; Baeza-Yates & Ribeiro-Neto 1999; Belew 2000). In the present study, IR supplements EVA in the following important ways:

- It provides a conceptual basis for classification of the *suras*. Classification of natural language text on the basis of semantic content has been and remains a central issue in the IR research community. A foundational principle is that documents cannot only be classified on the basis of their semantics, but on the basis of their lexical semantics more specifically: documents containing words like 'field', 'crops', and 'yield' constitute a class because their lexical semantics indicate that they are about the same kind of thing --here farming-- and this class is distinct from another containing, say, 'computer', 'keyboard', and 'mouse'. That document semantics is determined solely by lexical content is, of course, theoretically indefensible. In the currently-dominant paradigm for the modelling of natural language, generative linguistics, the semantics of any linguistic unit more complex than a morpheme is a function of the constituent structure of its syntax. Sentence semantics, for example, is determined by Frege's principle of compositionality --the meaning of a sentence is a function both of the meanings of its constituent words and of their 'manner of combination', that is, of the sentence's constituent structure, so that 'dog bites man' differs in meaning from 'man bites dog' even though the words remain the same. And, because documents are multiple-sentence collections, this principle extends to them as well, though clearly document semantics is not an additive function of its constituent sentence semantics (see for example Fodor 2001;

Fodor & Pylyshyn 1988; Larson & Segal 1995). Mainstream IR ignores this syntactic component of document semantics on the empirical grounds that document semantics based solely on lexical content suffices for efficient classification in practical applications (see discussion in Salton & McGill 1983,131 & 266-7). This study does the same.

- It provides a range of techniques for selecting from the documents in a collection the lexical variables that are most useful for document classification.
- It provides a paradigm for representation of documentary data, vector space representation, which is compatible with EVA analytical techniques.

This study regards the Qur'an as a collection of 114 documents, the constituent *suras*, and applies IR techniques for selection of lexical variables and vector space representation to creation of data for analysis from this collection. EVA methods are used to analyze this data, and these analyses are then interpreted relative to the research question.

To carry out the analysis, a range of software tools is required. Where possible existing applications are used, details of which are given where appropriate in the course of discussion. It has, however, been necessary to write new applications for task-specific purpose. These are attached, with descriptive documentation, in Appendix 1.

Chapter Four

Literature Review

4.0 Introduction

Given the apparently arbitrary sequencing of the *suras* in the Qur'an, the research question is to see whether they can be classificd in a thematically coherent way on the basis of their semantic content using computational techniques from exploratory multivariate analysis and information retrieval. This section reviews existing work on that question, together with methodologically related work which guided the analytical approach taken by the present discussion.

4.1 Literature review

A consequence of the apparently arbitrary sequencing of the *suras* is that the revelation on any particular topic may be, and in many cases demonstrably is, spread across two or more non-contiguous ones. The whole text of the Qur'an was revealed gradually, piece by piece, in varying lengths, giving new teaching or commenting on events or answering questions according to circumstances. This process is described in several verses in the Qur'an:

> 'And a Qur'an We have distinctly separated its verses, that you may read it to mankind at intervals, and We have been sending it down successively (by successive revelations on different occasions)' (*Sura Al-Israa*: 106)

> 'Surely We only have been sending down the Qur'an on you, a successive sending down (by stages)' (*Sura Al-Insan*: 23)

The Qur'an itself recognizes the problem and proposes a solution in the command of *ratil*, 'arrangement of similar things together':

> 'Meditate during the night, except rarely. Half of it, or a little less. Or a little more. And arrange (*ratil*) the Qur'an in its arrangement (*tarteel*)." (*Sura Al-Muzzammil*: 2-4).

This 'arrangement of similar things together' has been one of the major topics in the philological tradition of Qur'anic scholarship, and various arrangements have been proposed. Some scholars, for example Al-Zarkashi (1988), have suggested classification by length: *al-tiwal* (long *suras*: 2, 3, 4, 5, 6, 7, 10), *al-mi'un* (*suras* with approximately 100 *ayat*: 9, 11, 12, 16, 17, 18, 20, 23, 26, 37), *al-mathani* (repetitive *suras* in which parables are regularly repeated: 8, 13, 14, 15, 19, 22, 24, 25, 27-36, 28-48), and *al-mufassal* (short *suras*: 49-114). Others have developed different partitions of the text, for example *juz'* (pl. *ajza'*) which literally means 'part' or 'portion': the Qur'an is divided into 30 portions of approximately equal length for easy recitation during the thirty nights of a month, especially of the month of Ramadan. Usually they are indicated by the word and the number of it given alongside, (e.g. *juz'* 30 beginning with *sura* 78) (Watt & Bell 1970, 57). Abdel Haleem (1993, 71) argues that a most relevant and fruitful approach to understanding the text of the Qur'an is by means of two concepts developed by Muslim scholars in the Classical period: context and internal relationships. The importance of context (*maqām*) was recognised and formulated for the study of the text of the Qur'an by Muslim linguists whose work in this respect anticipated by many centuries modern linguistic thinking about the crucial importance of context in understanding discourse. Another key tool of Qur'anic exegesis is the internal relationships between different parts of the Qur'an, expressed by Qur'anic scholars in the dictum: *al-Qur'an yufassir ba'duhu*

ba'da ('some parts of the Qur'an explain others') - in modern linguistic terms 'intertextuality' – which, given the structure of the text, was argued to provide the most correct method of understanding the Qur'an. Many Qur'anic verses/passages can only be properly understood in the light of explanations provided in other verses or *suras*. As Abdul-Raof (2003, 362) puts it: 'the Qur'an as a text explains itself through the relevance of statements to each other'. Intertextuality is an important aspect in the Qur'an exegesis. Al-Shinqiti (1996) refers to the type of exegesis which relies on explaining a verse in the Qur'an through reference to other verses as 'intertextual exegesis'. Abdul-Haleem (1999, 6) states that 'the Qur'an is not a legal textbook that treats each subject in a separate chapter. It may deal with matters of belief, morals, ritual and legislation within one and the same *sura*'. Maududi (2005) claims that:

> 'It would be foreign to the very nature of the Qur'an to group together in one place all verses relating to a specific subject; the nature of the Qur'an requires that the reader should find teachings revealed during the Medinan period interspersed with those of the Meccan period, and vice versa. It requires the juxtaposition of early discourses with instructions from the later period of the life of the Prophet. This blending of the teachings from different periods helps to provide an overall view and an integrated perspective of Islam, and acts as a safeguard against lopsidedness.'

Certain themes have been treated in more than one place in the Qur'an. For example the topic of divorce is covered in four different *suras* (*Al-Baqarah*, *Al-Nisa, Al-Ahzab*, and *Al-Talaq*) which, when placed together, provide a full picture of the regulations and laws regarding divorce. References to the stories of earlier prophets were repeated occasionally throughout the Qur'an; the particular form of the narrative mentioned varies according to

the situation which the prophet was facing, and thus arranging these episodes together will give a full account of the stories of these earlier prophets. In fact, the only single subject that has not been scattered throughout the Qur'an is that of Joseph in *Sura Yusuf* where his story is given sequentially.

Despite all this, no canonical or even preferred arrangement of the *suras* has emerged to date, and *ratil* by the reader remains fundamental to understanding of the text. The only classification that everyone agrees on (Robinson 1996) is that the *suras* can be divided into two groups defined by their time of composition and by thematic and other criteria: those composed in Mecca between 610 CE and 622 CE, and those composed in Medina between the *hijra* in 622 and the Prophet's death in 632 CE.

The broad themes of the Meccan *suras* are Allah and his unity (*tawhid*), the coming resurrection and judgement, and the role of the Prophet is that of announcer and 'warner'; the Medinan *suras* include Meccan themes but augment them by additional themes relating to Muslims' growing together into a community, the *umma,* and by regulations having to do with correct living. In addition to the thematic distinction, the *sura* length often offers a guiding criterion for distinguishing Meccan and Medinan revelations: Meccan *suras* are usually short while Medinan ones are longer, e.g. *Sura Al-Baqarah*. Also, the form of address differs between Meccan and Medinan *suras*. Often the address 'O you who believe' and 'O people of the book' indicates a Medinan origin while 'O Mankind' is usually of Meccan origin. The distinction between Meccan and Medinan revelations is one of the important branches of *Ulum al-Quran* ('the Sciences of the Qur'an'). It is not merely of historical interest, but is also particularly important for the understanding and interpretation of the respective verses.

Finally, it should be noted that, though there is broad agreement on which *suras* belong to the Meccan and Medinan groups, the attribution of a few remains controversial. For example, it has been disputed whether *sura Al-Bayyina* has been revealed in Mecca or Medina. Similar arguments are held for other *suras* as *Al-Zalzalah* and *Al-Rahman.* Moreover, Robinson (1996) states that Nöldeke (1909) classified the *suras* 13, 55, 76 and 99 as Meccan, whereas the editors of the standard Egyptian edition of the Qur'an subsequently classified them as Medinan. The standard Egyptian chronology is not, however, based on unanimous tradition, and for each of these four *suras* it is possible to find classical authorities who favoured a Meccan date.

Many *suras* contain material from both periods of revelation, and in some cases there are differences of opinion among scholars concerning the classification of a particular passage.

As regards existing work on exploratory multivariate analysis of the Qur'an, the literature review is easily written: there is none. In fact, to the present writer's knowledge, there is no computationally-based analytical work of any kind on the Qur'an, an assertion supported by Berg (2001, 394) who states that:

> Except for isolated efforts..., little has been done with computer-assisted analysis of the text...Thus, for the present, computer-assisted analysis of the Qur'an remains an intriguing but unexplored field.

A few efforts featuring the use of computational technology on the Qur'an have been confined to the creation and representation of electronic versions of the text, along with search engines, rhythmic citations and online translations of the text in different languages[2]. Another contribution to the computational analysis of the Qur'an is a

[2] See http://www.islamicity.com/mosque/quran/ and http://quran.muslim-web.com/

morphological analyzer developed by Dror *et al.* (2004). They present a system for morphological analysis and annotation of the Qur'an for research and teaching purposes. The system provides a variety of queries on the Qur'anic text that makes reference not only to the words but also to their linguistic attributes. The results of their analysis are stored in a database and are accessed through a graphical user interface which facilitates the presentation of complex queries.

As such, it appears that the present analysis is the first of its kind. There are, of course, whole disciplines devoted to computational analysis of text --for example, computational linguistics, natural language processing, information retrieval, and data mining-- each with a large associated literature, but to review these would amount to writing an account of the history and present state of computational text analysis. This is clearly out of question. Instead, aspects of these disciplines which are methodologically relevant to the present study are described as they arise in the course of discussion.

Chapter Five

Data Creation

5.0 Introduction

This section is in two main parts. The first part outlines general principles of data creation, and the second applies these principles to construction of the Qur'an-based data used in this study. The aim is clarity: conceptual issues in data creation are kept separate from the details of Qur'an data creation.

5.1 Data: general principles

5.1.1 The nature of data

> The world is a place of unbelievable complexity. No matter how closely we look at some facet of the world, there is an infinite depth of detail. Yet our brains and minds construct meaningful (for us) simplicities from the stunning complexity that surrounds us. Using these simplicities we make representations of the world that we find useful, like lunch and banks. And using these simplicities, we can collect and record impressions about various facets of them, which we call data. It is this data that we then explore...to understand something about the reality of the world --to discover information' (Pyle 1999, 46).

'Data' is the plural of 'datum', the past participle of Latin 'dare', 'to give', and means 'that which is given'. A datum is therefore something to be accepted at face value, a true statement about the world. What is a true statement about the world? That question has been debated in philosophical metaphysics since antiquity and probably before, and, in our own time, has been intensively studied by the disciplines that comprise cognitive science. The issues are complex, controversy abounds, and the associated academic literatures are

vast: saying what a true statement about the world might be is anything but straightforward. One response –the one typically adopted by the world of data processing— is simply to ignore the metaphysics. 'Data' is taken to be primitive. The assumption is that everyone shares a working definition of it, and discussion proceeds on the basis of that presumed common intuition. Given the scope of the present discussion, the only viable alternative is to do the same here.

Data is ontologically different from the world. The world is as it is; data is an interpretation of it for the purpose of scientific study. The weather is not the meteorologist's data – measurements of such things as air temperature are. A text corpus is not the linguist's data –measurements of such things as average sentence length are. Data is constructed from observation of things in the world, and the process of construction raises a range of issues that determine the amenability of the data to analysis and the interpretability of the analytical results. The importance of understanding such data issues to exploratory multivariate analysis can hardly be overstated. On the one hand, "no matter how powerful the exploring tools, or aggressive the explorer, nothing can be discovered that is beyond the limits of the data itself" (Pyle 1999, 46). On the other, failure to understand relevant characteristics of data can lead to results and interpretations that are distorted or even worthless. For these reasons, a fairly detailed account of data issues is given prior to moving on to exploratory multivariate methods.

5.1.2 Variable selection

Given that data is an interpretation of some aspect of the world, what does such an interpretation look like? It is a description of the selected aspect of the world in terms of variables. A variable is a symbol, and as such is a physical entity with a conventional semantics, where a conventional semantics is understood as one in which the designation

of a physical thing as a symbol together with the connection between the symbol and what it represents are determined by agreement within a community. The symbol 'A', for example, represents the phoneme /a/ by common assent, not because there is any necessary connection between it and what it represents. Since each variable has a conventional semantics, the set of variables chosen to describe a domain of inquiry constitutes the template in terms of which the domain is interpreted. Selection of variables is, therefore, crucial to the success of any data analysis, and care must be taken to choose appropriate ones.

Which variables are appropriate in any given case? That depends on the nature of the research. Data can only be created in relation to a research question that provides an interpretative orientation in the domain of interest. Without such an orientation, how does one know what to observe, what is important, and what is not? The fundamental principle in variable selection is that the variables must describe all and only those aspects of the domain that are relevant to the research question. In general, this is an unattainable ideal. Any domain can be described by an essentially arbitrary number of finite sets of variables; selection of one particular set can only be done on the basis of personal knowledge of the domain and of the body of scientific theory associated with it, tempered by personal discretion. In other words, there is no algorithm for choosing an optimally relevant set of variables for a research question.

5.1.3 Data representation

If it is to be analyzed using mathematical methods, the selected data variables need to be mathematically represented. A widely used way of doing this is vector space representation (Salton *et al.* 1975; Salton & McGill 1983, ch. 4; Pyle 1999, 202 ff; Belew 2000, 86-7). A

vector is a sequence of scalars indexed by the positive integers 1, 2, ...*n*, where a scalar is a single number:

V = | 1.6 | 2.4 | 7.5 | 0.6 |
    | 1 | 2 | 3 | 4 |

Figure 2: An example of a vector

A vector space is a geometrical interpretation of a vector in which:

- the dimensionality of the vector, that is, its index length *n*, defines an *n*-dimensional space. There are various possible types of space, but for present purposes the type of space is taken to be the Euclidean one familiar from elementary mathematics, in which the axes are straight lines orthogonal to one another.
- the sequence of scalars comprising the vector specify coordinates in that space. These coordinates are relative to the scales of the orthogonal axes, which are arbitrary and can be chosen for convenience to the purpose at hand.
- the vector itself is a point in the space

For example, assume a vector **v** = [12, 20]. This vector defines a two-dimensional space, and its two components are coordinates in that space.

A vector **v** = [12, 20, 10] defines a 3-dimensional space, and its values in the chosen coordinate system place it at the corresponding position in the space.

A length-4 vector defines a point in 4-dimensional space, and so on to any dimensionality *n*. Mathematically there is no problem with spaces of dimension greater than 3: the conceptual and formal frameworks apply to *n*-dimensional spaces, for any *n*, as straightforwardly as to 2 or 3 dimensional ones. The only problem lies in the possibility of

visualization and intuitive understanding. As the number of variables, and thus dimensions, grows beyond 3, graphical representation and intuitive comprehension of it become impossible -- who can visualize points in a four-dimensional space, not to speak of a 40-dimensional one? The key is to remember that mathematical dimension has no necessary connection with the dimensions of the physical world in which we live.

Data sets typically consist of more or less numerous data items each of which is described in terms of the selected variables. Where vector space representation is used, each data item is described by a vector, and the data is consequently a set of vectors. Such a set of vectors is conveniently represented as a matrix in which the rows are the data items and the columns the variables. Thus, data consisting of m items each of which is described by n variables is represented by an m x n matrix D in which D_i (for $i = 1...m$) is the i'th data item, D_j (for $j = 1..n$) is the j'th variable, and D_{ij} the the value of variable j for data item i.

5.1.4 Variable value assignment

The semantics of each variable determines a particular interpretation of the domain of inquiry, and the domain is 'measured' in terms of the semantics. That measurement constitutes the values of the variables: height in metres = 1.71, weight in kilograms = 70, and so on. Measurement is fundamental in the creation of data because it makes the link between data and the world, and thus allows the results of data analysis to be applied to the understanding of the world (Pyle 1999, ch. 2).

Measurement is only possible in terms of some scale. There are various types of measurement scale, and these are discussed at length in the relevant textbooks (Hair *et al.* 1998, 6-9; Pyle 1999, ch. 2), but for present purposes the main dichotomy is between numeric and non-numeric. The multivariate methods discussed in due course assume

numeric measurement as the default case, and for that reason the same is assumed in what follows.

5.1.5. Data validation

The most basic characteristic of data is that it be complete and accurate, where 'complete' means that all variables for all cases in the data set have values associated with them, and 'accurate' that all values assigned to variables faithfully reflect the reality they represent. These are stringent requirements: most datasets large enough to have multivariate analysis usefully applied to them probably contain error to greater or lesser degrees. Measurement error arises in numerous ways –tolerances in measuring instruments, human inaccuracy in the use of the instruments, corruption at one or more points in data transmission, and so on. Such deviation from absolute exactitude is known as noise.

Because error in data distorts analytical results, it has to be eliminated as much as possible. This is a two-step process: the first step is to determine the amount and nature of the error, and the second is to mitigate or remove it. Methods for error identification and correction are discussed in the relevant textbooks, for example Hair *et al.* (1998, ch. 2).

5.1.6 Data transformation

Once the data has been constructed and validated, it may be necessary to transform it in various ways prior to analyzing it. Three types of transformation are considered: (i) normalization for variation in document length, (ii) minimization of data sparsity, and (iii) data linearization.

For clarity, the general concept of data which has been used so far will henceforth be confined to data derived from natural language text on the grounds that the Qur'an is a

natural language text, and understanding of data in its full generality is not required. More specifically, the discussion of these issues is oriented in such a way as to be maximally relevant to construction of the Qur'an data matrix: the assumption will be that the data is in the form of an m x n matrix Q in which:

- the rows i, for $i = 1..m$, represent the m documents d_i of a document collection D.
- the columns j, for $j = 1..n$, are the variables that represent the n unique lexical types t_j which occur in D, where 'lexical type' is understood as an abstraction over a set of identical lexical tokens, 'lexical token' as a string of alphanumeric symbols, and 'abstraction' as a set label; the lexical type CAT = $\{x \mid x = \text{'cat'}\}$, for example. This understanding of lexical types and tokens is standard in disciplines concerned with text processing such as corpus linguistics, natural language processing, and information retrieval (for example Manning & Schütze 1999, 21-3 & 124-130; Palmer 2000).
- The matrix elements Q_{ij} are integers representing the frequency of lexical type j in document i. Each of the m rows in Q is therefore a lexical frequency profile for the associated document.

5.1.6.1 Document length variation

It is a simple fact of life that documents in any given collection can vary considerably in length, from short memos to novels. Where variables represent lexical frequency, as here, such length disparity must be taken into account. To see why, consider counting the number of occurrences of some lexical type w in a multi-document corpus in which the documents vary in length. Say there are 3 occurrences of w both in documents 1 and 2. Knowing only these frequencies, one would judge the two documents identical on this variable. But 1 is 5000 words long, and 2 is only 500. It is clear that, though they both have

the same number of occurrences of *w*, the significance of their respective frequencies is far from identical: *w* is relatively infrequent in 1 in the sense that its probability of occurrence is only 3/5000, and relatively frequent in 2 because its probability of occurrence is 3/500, or ten times as great; if 2 had been 5000 words long instead of 500, the frequency of *w* would, on the basis of its observed probability, have been 10 x 3 = 30, and on that basis the two documents would be judged as very different on variable *w*. In short, the longer a document, the more likely in general a given word with a specific probability of occurrence is to occur in it, and, if it occurs, the higher the frequency of occurrence is in general likely to be. Weighting to compensate for variation in text length is therefore necessary to remove this effect. Two frequently used methods are:

a. Normalization by document vector length

In linear algebra (for example Fraleigh & Beauregard 1995, 21-3) the norm or magnitude of an *n*-dimensional vector *v* is defined as:

$$| v |= \sqrt[2]{v1^2 + v2^2 + ... + vn^2}$$

where $|v|$ is the standard notation for the vector norm, and the v_i are indices to the values in the components of the vectors. The unit vector *u* corresponding to *v*, that is, the vector having the same direction as *v* but length 1, is then defined as:

$$u = v / |v|$$

This has the effect of adjusting vector values in proportion to document length. The length of each document row vector Q_i reflects the length of the document that the row represents in the sense that it contains the frequencies of the lexical types for that row. The greater the

length, the smaller the result of the division and thus the smaller the post-normalization values in Q_i, and vice versa as the Q_i vector length grows shorter.

By replacing all the document vectors in the data matrix Q with unit vectors, it becomes possible to determine the relative closeness in *n*-dimensional space of any two row vectors Q_i and Q_j, and thus the proximity of the two documents that these vectors represent, using the inner product of Q_i and Q_j, where the inner product is:

$$Q_i \cdot Q_j = Q_{i1}Q_{j1} + Q_{i2}Q_{j2} + \ldots + Q_{in}Q_{jn}$$

The cosine of the angle θ between Q_i and Q_j is defined as:

$$\cos(\theta) = (Q_i \cdot Q_j) / (|Q_i||Q_j|)$$

This is the *de facto* standard method of calculating relative proximity of documents in Information Retrieval, where it is known as cosine normalization (Singhal *et al.*1995; see also variants of cosine normalization in Singhal *et al.* (1996a, 1996b).

b. Normalization by average document length

Individual document vectors can be normalized in relation to the average length of documents in the collection:

$$Q_{i,n} = Q_{i,u} (\mu / l_{Qi})$$

where:

- $Q_{i,n}$ is a normalized document vector in the data matrix Q, for i = 1..number of rows in Q m or, equivalently, the number of documents in the collection.
- $Q_{i,u}$ is an unnormalized document vector, for i as above.

- l_{Qi} is number of lexical types in document Q_i.
- μ is the mean length, measured by the number of lexical tokens in each document,

so that:

$$\mu_c = (\sum_{i=1..m}(l_{Qi})) / m$$

Thus, the values in each document vector Q_i are multiplied by the ratio of the average number of lexical tokens per document across the collection to the number of lexical tokens in Q_i. The longer the document the numerically smaller the ratio, and vice versa; the effect is therefore to decrease the values in the vectors that represent long documents, and increase them in vectors that represent short ones as a function of average document length.

On document length normalization see Baeza-Yates & Ribeiro-Neto (1999, 29), Belew (2000, 89-92), Lebart & Rajman (2000, 477-505), Singhal *et al.* (1995, 1996a, 1996b).

5.1.6.2 Sparsity minimization

Sparsity is a major issue in data analysis generally. The concept of the manifold is central to the understanding of why this is so. It comes from mathematical topology (Mendelson 1975, Munkres 2000), a branch of pure mathematics concerned with geometrical properties; for present purposes it can be understood as the shape of data in *n*-dimensional space. What is the 'shape' of data (Pyle 1999, 84-6)? Consider a reasonably large data set of, say, 1000 3-dimensional real-valued vectors, no two of which are identical. If these vectors are plotted in 3-dimensional space, they form a cloud of points with an identifiable shape within the general space, as in Figure 3:

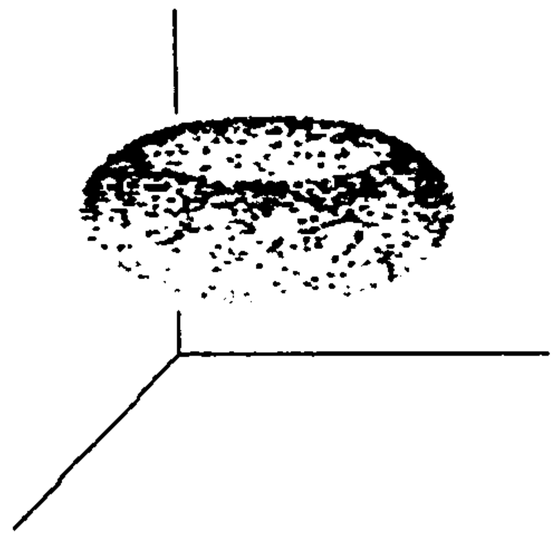


Figure 3: A manifold in 3-dimensional space

That shape is a manifold. The idea extends directly to any dimensionality, though such general spaces can no longer be shown graphically. For the purposes of this discussion, therefore, a manifold is a set of vectors in *n*-dimensional space. For linear data the manifold is a line or a plane, but in the nonlinear case much more complex shapes like the one above are also possible, depending on the nonlinearity.

To discern the shape of a manifold, it is intuitively clear that there have to be enough data points to give it adequate definition. If, as in Figure 4a, there are just two points, the only reasonable manifold to propose is a line; any number of alternative manifolds are, of course, possible --the two points could come from a far more complex manifold like Figure 4c-- but to propose this on the basis of just two points would clearly be unjustified.

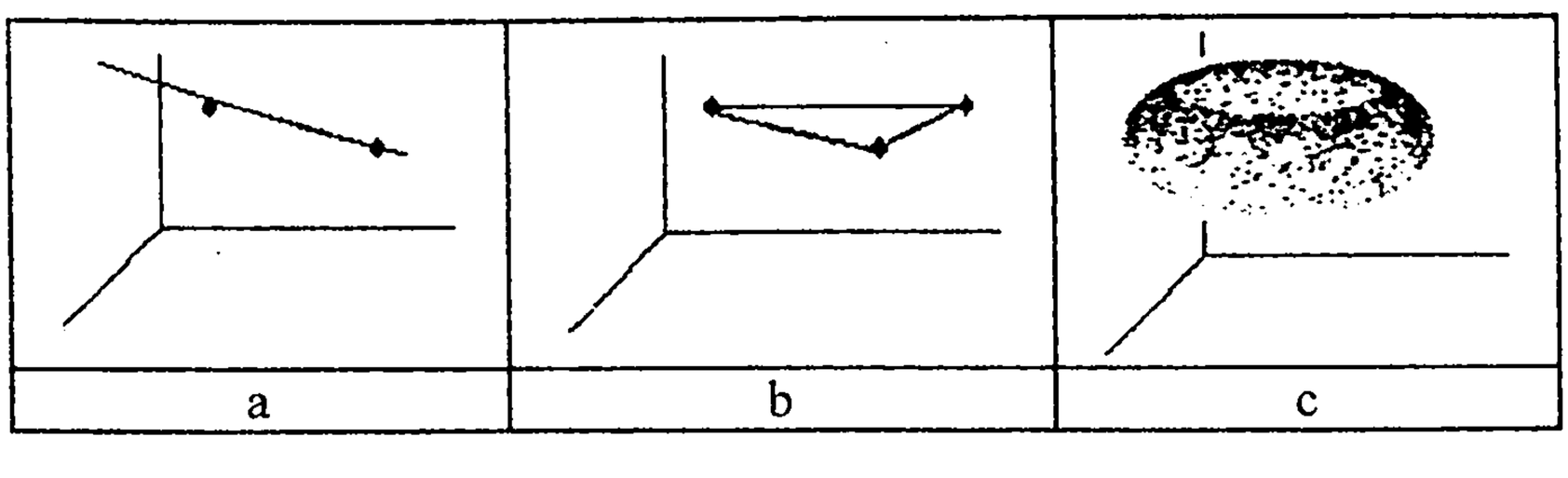


Figure 4: Degrees of manifold definition

Where there are 3 points, a plane as in Figure 4b, or a curved line, joining the points would be reasonable. But it is only as the number of data points grows that the true shape of the manifold emerges, as in (4c). The general rule, therefore, is: the more data the better.

Clearly, there comes a stage where increasing the amount of data becomes redundant in the sense that it simply confirms an already-clear manifold shape, but it doesn't do any harm.

In dealing with high-dimensional data, however, having too much is rarely a problem. Quite the opposite --the usual situation with high-dimensional data is that there is far too little. High-dimensional spaces are inherently sparse, and, to achieve adequate definition of the data manifold, the amount of data required very rapidly becomes intractably large; this phenomenon was described as the 'curse of dimensionality' by Bellman (1961). To see the problem, consider three data sets each of which contains 10 items:

- Set 1 is univariate, and the single variable can take integer values in the range 1..10. The ratio of data points to possible values is 10/10 = 1, that is, the data points completely fill the data space.
- Set 2 is bivariate, and each of the two variables can take integer values in the range 1..10. The ratio of data points to possible value pairs is 10 / (10 x 10) = 0.1, that is, the data points occupy 10% of the data space.
- Set 3 is trivariate, and each of the three variables can take integer values in the range 1..10. The ratio of data points to possible value triples is 10 / (10 x 10 x 10) = 0.01, that is, the data points occupy 1% of the data space.

And so on for increasing dimensionality 4, 5...*n*: for a data set of fixed size *d*, the ratio of actual to possible points in the data space is d / r^n, where *r* is the number of different values that each variable can take (assuming for simplicity that all variables are identical in this respect). In other words, as dimensionality increases, the ratio of actual to possible points in the data space decreases at an exponential rate. In principle, therefore, fixed-size data manifolds very rapidly become sparser as the dimensionality of the space in which they are embedded grows; to maintain resolution of the manifold at any preferred ratio, the

amount of data required must therefore grow exponentially with the dimensionality, and so getting enough data becomes a serious problem even at relatively low dimensionalities, and an insuperable one soon thereafter. In practice the problem is not as severe as all this might suggest, since a typical real-world data set is not in general evenly or randomly spread around its space, but rather tends to be concentrated in one or more distinct regions of the space. Dimensionality nevertheless remains a potential problem for data analysis in any given application, and the moral is that data dimensionality should be kept as low as possible consistent with the need to describe the domain of inquiry adequately. For discussion of issues relating to high-dimensional data see Bishop (1995, chs. 1 & 8), Pyle (1999, ch. 2 & 355-60, 424 ff), Scott & Thompson (1983), Verleysen (2003), Verleysen *et al.* (2003).

Data sparsity has a particular relevance to the present application. As noted, the aim is to create a matrix M in which the rows are the *sura* data points, the column variables are lexical types, and the value at any given matrix location M_{ij} is the frequency of lexical type *j* in *sura i*. There are 6696 lexical types in the Qur'an after the various preprocessing steps to be described below. The frequencies of these types were calculated, sorted into descending order of magnitude, and plotted. The result is shown in figure 5:

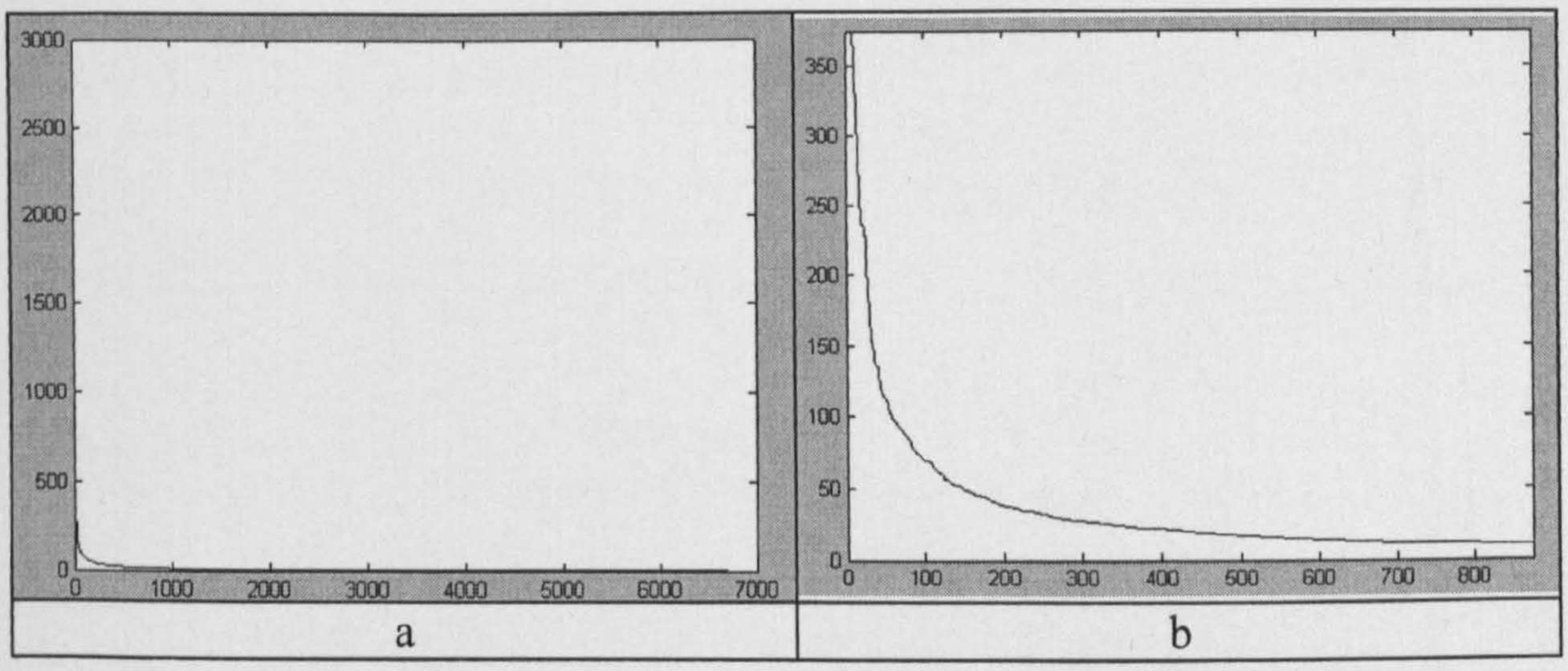


Figure 5: Frequencies of 6696 lexical types in the Qur'an

Figure 5a is the full plot, and 5b is a zoomed-in region near the origin to display the shape of that region more clearly. There is a relatively small number of very frequent types, a moderate number of moderately frequent types, and a large number of very infrequent types. This distribution is characteristic of lexical frequency distributions in natural language text generally (Baayen 2001; see also Manning & Schütze 1999, 20-29). It is known as the LNRE ('Large Numbers of Rare Events') distribution, and its shape remains pretty much constant even for NL text corpora many orders of magnitude larger than the Qur'an: the number of the few very frequent types continues to grow quickly as the corpus size grows and the number of moderately frequent types continues to grow moderately quickly, but the frequencies of the very infrequent types change hardly at all --instead, more and more types are added to the list. It is, therefore, clear that lexical frequency data derived from natural language text will, in general, be very sparse on account of the large number of very infrequent lexical type variables; in the case of the Qur'an, there are 114 data points in a 6696-dimensional space, which is very sparse indeed.

The obvious solution to sparsity, both in general and in the present application, is to select an optimal set of variables at the data design stage, but this is more easily said than done. As already noted, there is no algorithm for choosing an optimally relevant and therefore minimal set of variables for a research question, and therefore no way of knowing a priori whether the dimensionality of a given data set is as low as it can be. Because of this, a range of methods for transforming data matrices so as to reduce their dimensionality has been developed, and, in cases where the data is sparse, application of one or more of these methods can very substantially improve analytical results by giving the manifold better definition. These are described in what follows, and then applied to the Qur'an data.

5.1.6.2.1 Stemming

Lexical types were defined earlier as sets of identical tokens. By modifying this definition, substantial reduction in the number of types that any given document collection contains is possible, which in turn reduces the dimensionality of a token-frequency matrix derived from it.

Fundamental to the morphologies of many languages, including Arabic, is the process whereby prefixes and suffixes are attached to lexical stems, and/or the lexical stems themselves mutated in some way, in order to mark syntactic function or some modification to the primitive semantic denotation of the stem. In English, an example of syntactic function marking is the nominative/non-nominative alternation of pronouns (I/me, he/him) and of semantic modification of 'minister' / 'administer' / 'administration' (Carstairs 2002). In Arabic morphological variants of words are formed either by attaching suffixes or prefixes to word stems (concatenative morphology) as in قلم *qalam* 'pen' / قلمه *qalamahu* 'his pen'/ قلمها *qalamiha* 'her pen', or by undergoing morphological changes to the root/stem (nonconcatenative morphology) as in the formation of broken plurals قلم *qalm* 'pen'/ اقلام *aqlaam* 'pens' (Watson 2002). Document collections written in such languages typically contain more or less numerous morphological variants of lexical stems. These variants of words which have similar semantic interpretations are considered to belong to the same stem and to be equivalent for purposes of text analysis and information retrieval. For this reason, a number of stemming algorithms have been developed in an attempt to reduce such morphological variants of words to their common stem.

Now, if a lexical type is defined as the set of identical alphabetic strings, then each variant of a given stem is treated as a distinct lexical type and assigned a column in the data matrix. If, however, all the morphological variants of a stem are collapsed into an equivalence class

which then constitutes the lexical type, so that, for example, the type CAT = {x | x = a morphological variant of 'cat'} such as 'cats', 'catty', 'cattery' and so on, the number of types and thus columns of the frequency matrix can be more or less substantially reduced, depending on the morphological characteristics of the language in question and the composition of the document collection being used. The frequency of a lexical type so defined in the frequency matrix is then the sum of the frequencies of the aggregated variants.

At first glance, it might seem that such creation of equivalence classes loses information, and that this loss is bound to adversely affect the validity of analyses based on the data. Just the opposite is true, however. If lexical types are regarded as sets of identical tokens, then each type is represented as a separate variable column in the data matrix, and all columns are treated equally in the analytical methods described later. The implication is that morphologically related tokens are treated exactly the same as unrelated ones. In other words, there is no distinction between the semantic distances among morphological variants of a single stem on the one hand, and those between unrelated stems on the other -- the semantic difference between 'administer' and 'administration' is taken to be the same as that between 'administer' and 'cow'. If, as here, the aim is to classify documents on the basis of their lexical semantics, this is bound to distort the data and thus the analytical results based on it. Creation of equivalence classes based on morphological relatedness eliminates this distortion.

That said, discretion is necessary in the creation of equivalence classes:

- The equivalence classes must be defined in a way that is sensitive to the application domain. For example, collapsing the morphologically related 'administer', 'administration', and 'administrative' into a single lexical type is probably justified in

a document collection having to do with, say, the history of art, but might remove important semantic distinctions in a business management collection.

- An underlying assumption in stemming is that affixation and/or stem mutation involve more or less negligible modifications to lexical semantics, and that these can be disregarded. The validity of this assumption is generally accepted for inflectional morphology, such as plural and tense markers, but it is less clear-cut for derivational morphology --does one, for example, want to claim that the semantic difference between 'neutron' and 'neutralise' (van Rijsbergen 1979, 22) is negligible? .

In short, collapsing morphologically-related tokens into equivalence classes needs to be carefully tuned for particular applications, both in terms of the morphological characteristics of the language in which the documents of interest are written, and of the research aims of the application in question.

In practice, construction of equivalence classes for morphologically related variants is usually done automatically using a stemming algorithm implemented in computer software. Depending on the stemmer being used, human discretion in the definition of equivalence classes will have been incorporated into the algorithm to greater or lesser degrees, but the assumption in the information retrieval community is that there will be some degree of error or at least non-optimality in the derived lexical types, though the general view is that the error/non-optimality rate is low enough for it not to degrade document retrieval significantly, and that it can be tolerated (van Rijsbergen 1979, 22). It is also possible to review and where necessary edit the stemmer output, though this can be very time consuming for large document collections.

On stemming algorithms and their application in computational text processing see Lovins (1968), Porter (1980), Frakes (1992), Krovetz (1993), Hull (1996), Kraaij & Pohlman (1996), Xu & Croft (1998), Fuller & Zobel (1998), Ziviani (1999, 168-9), Belew (2000, 44-7), Hollink *et al.* (2003). Detailed discussion of stemming of Arabic in particular is deferred until section 5.2.1.2.2 below.

Stemmers have been developed for a wide range of languages and for a variety of purposes. The structure of these stemmers range from the simplest technique of removing suffixes, to a more sophisticated design which uses the morphological structure of words to derive a stem. The designs of stemmers are normally language-specific and therefore require some linguistic expertise in the language.

Several different techniques were proposed for stemming English text. These stemmers are classified by Al-Sughaiyer & Al-Kharashi (2004) into strong and weak stemmers. Strong stemmers tend to overstem words by removing endings which are not real suffixes but an actual part of the word. Weak stemmers remove a fewer number of suffixes and hence causing words that belong to the same stem to be conflated separately.

The most common stemming algorithms for English were produced by Porter (1980) and Lovins (1968). The Porter stemmer applies a set of rules to remove suffixes from a word until none of the rules apply; it suffers from conflation of words that have different meanings and it ignores prefixes completely. The Lovins stemmer (Lovins 1968) is similar in mechanism but has a larger set of suffixes and does not apply its rules iteratively. Krovetz (1993) attempted to resolve some of the limitations of the Porter and Lovins stemmers, by taking morphological structure into account. Xu & Croft (1998) proposed a technique to stemming that relies on corpus-based word co-occurrence statistics. Their algorithm does not actually remove suffixes, but instead defines equivalence classes of

words that should be conflated. Stemmers have been developed for a wide range of other languages such as French, Indonesian, Malay, Latin, Dutch (Savoy 1993; Asian *et al.* 2005; Ahmad *et al.* 1996; Schinke *et al.* 1996; Kraaij & Pohlman 1994).

5.1.6.2.2 Keyword selection

A seminal principle in Information Retrieval, extensively confirmed by empirical results, is that not all lexical types in a document collection are equally useful in document classification (van Rijsbergen 1979, ch. 2; Salton & McGill 1983, ch. 3). Various ways of identifying relatively more useful variables exist, and the remainder of this section describes the ones used in subsequent discussion. The selected methods include those that appear most often in the relevant IR literature, with one exception: keyword discrimination (Salton & McGill 1983, 66-71; Salton & Buckley 1988; Belew 2000, 88). Like the other methods, keyword discrimination was assessed in relation to the initial Qur'an lexical frequency matrix, and the results were found to be very close to those from several other methods. To avoid unnecessary duplication, the decision was made not to use it.

A fundamental observation about these methods needs to be made at this stage: they can be used either for keyword selection or for keyword weighting. In both cases, a given method is used to calculate a score for each of the variables / columns in the data matrix. For selection, a threshold is then chosen and a binary decision made about whether or not to include each variable in the data, based on where its score falls relative to the threshold. For weighting there is no threshold. Instead, some function of the variable's score is used to update the values in the corresponding matrix column. Weighting seems more attractive in principle. For one thing, it ranks the variables in terms of their importance relative to whatever criterion the method in question uses, thereby enhancing the usefulness of variables that are relatively more important in distinguishing documents from one another

and suppressing those that are relatively less important, whereas selection simply classifies the variables into 'useful' and 'not useful' categories with no attempt at ranking. For another, selection requires a threshold to be defined by the analyst, thereby possibly introducing an undesirable subjectivity, whereas weighting requires no such threshold. It is for these reasons that weighting is standardly used in Information Retrieval. Nevertheless, in the present application the methods are used for selection rather than weighting in what follows, for two reasons:

1. The object of the exercise is dimensionality reduction, and, to eliminate the less important variables, a threshold is required in either case. This obviates one of the advantages of weighting.
2. Though, as just noted, a set of variables that have been systematically ranked in importance relative to a specified criterion seems intuitively preferable to a set of variables that has not, in the present application weighting would distort the data and thus do more harm than good. Justification for this claim is deferred to section 5.2.6.3.1 to avoid pre-empting the discussion. For the moment, it will simply be noted that variable selection rather than weighting is adopted in order to avoid this potential problem.

a) Dimensionality reduction based on lexical type frequency

Luhn, one of the founders of Information Retrieval, proposed that the relative frequency of lexical types in a document collection is a fundamental criterion for classifying documents relative to one another (Luhn 1957, 1958; discussed in Salton & McGill 1983, 60-63; van Rijsbergen 1979, 15f; Belew 2000, 76 ff). The intuition underlying this is simple: if an author uses a word repeatedly in a text, then the text is more likely to be about what the word denotes than it is to be about the denotation of a word that is infrequently used;

documents with similar lexical frequency profiles are classified together and distinguished from those whose profiles are different. Given a binary-valued matrix Q, in which the value at Q_{ij} records only the presence or absence of lexical type *j* in document *i*, Q_{ij} can be weighted by multiplying the binary value by the frequency of type *j* in document *i* (for other weightings see Singhal *et al.* 1995):

$$Q_{ij} = Q_{ij}freq_{ij}$$

Luhn also proposed, however, that the usefulness of a word for document classification does not increase monotonically with lexical frequency, and more specifically that very frequent words on the one hand and very infrequent words on the other are less useful for the purpose than medium frequency ones. He therefore proposed an algorithm for selecting the most useful words in a collection as keywords, and discarding the less useful ones:

1. Given some document collection, compile a list of all the word types or 'terms' that occur in the collection as a whole
2. Calculate the frequencies of all the terms across the entire collection
3. Sort the terms in descending order of frequency
4. Select a maximum frequency threshold and eliminate all terms with frequencies greater than the threshold
5. Select a minimum frequency threshold and eliminate all terms with frequencies less than the threshold
6. Retain the medium-frequency terms between the maximum and minimum thresholds as the keywords for distinguishing between and among documents

Substantial dimensionality reduction can be achieved by applying this algorithm to a data matrix. Luhn did not, however, provide a way of determining upper and lower thresholds,

and there is consequently the ever-present danger that too many or too few types will be eliminated, thus compromising classification based on the set of retained variables. Various proposals for determining appropriate thresholds have been made since Luhn's time, but the general position remains that selection of thresholds remains an empirical matter based on relative lexical frequency in specific applications --in other words, select the thresholds that give the best results.

b) Dimensionality reduction based on variance

As we saw in the foregoing discussion of data, any variable x is an interpretation of some aspect of the world, and a value assigned to x is a measurement of the world in terms of that interpretation. If x is to describe more than one object --the heights of 1000 people, say-- or a single object over time, such as the temperature on successive days, then it must take values characteristic of each person or day. Unless all 1000 people are exactly the same height, or the temperature is constant, these values will vary. This possibility of variation gives x its descriptive utility: a constant value for x says that what x represents in the world does not change, moderate variation in the value says that that aspect of the world changes only a little, and widely differing values that it changes substantially. In general, therefore, the possibility of variation in the values assigned to variables is fundamental to the ability of variables to represent reality.

Classification of documents or of anything else therefore depends on there being variation in their characteristics --identical objects, say cars of the same make having the same colour and technical specification, cannot be meaningfully classified. When the objects to be classified are described by variables, then the variables are only useful for the purpose if there is significant variation in the values that it takes. If, for example, a large random collection of people was described by variables like 'height', 'weight', and 'income', there

would be substantial variation in values for each of them, and they could legitimately be used to classify the people in the sample. On the other hand, a variable like 'has nose' would be effectively useless, since, with very few exceptions, everyone has a nose --there would be almost no variation in the Boolean value 1 for this variable. In any classification exercise, therefore, one is looking for variables with substantial variation in their values, and can disregard variables with little or no variation.

Mathematically, the degree of variation in the values of a variable is described by its variance (Pyle 1999, 162 ff). Assume a variable x with a distribution of n numerical values across some range. The mean, or average, is a measure of the central tendency of x, and is calculated by adding all n values and then dividing by n:

$$mean(x) = (\sum_{i=1..n}(x_i))/n$$

The mean hides important information about the distributions of values in a variable. Consider, for example, these (fictitious) runs of student marks on a percentage scale:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**
A	40	58	92	31	27	85	67	77	73	30	58
B	55	60	56	64	59	58	57	54	58	59	58

Table 1: Example data

The means for students A and B are identical, but the variations across the mark scale differ strikingly: student A is capable of both failure and excellence, and student B is remarkably consistent. Knowing only the averages one could not make the distinction; both the average and an indication of the spread of marks across the range are required. Assessing the spread can be problematic in practice, however. Where the number of marks

is few, as in the above example, visual inspection is sufficient, but what about longer runs? Visual inspection quickly fails; some quantitative measure that summarizes the spread of marks is required. That measure is variance.

The variance of a set of variable values is the average deviation of those values from their mean. Assume a set of n values $\{x_1, x_2...x_n\}$ assigned to a variable x. The mean of these values μ is $(x_1 + x_2 + ... + x_n) / n$. The amount by which any given value x_i differs from μ is then $x_i - \mu$. The average difference from μ across all values is therefore $\Sigma_{i=1..n} (x_i - \mu) / n$. This average difference of variable values from their mean almost but not quite corresponds to the definition of variance. One more step is necessary, and it is technical rather than conceptual. Because μ is an average, some of the variable values will be greater than μ, and some will be less. Consequently, some of the differences $(x_i - \mu)$ will be positive and some negative. When all the $(x_i - \mu)$ are added up, as above, they will cancel each other out. To prevent this, the $(x_i - \mu)$ are squared. The standard definition of variance for n values $\{x_1, x_2...x_n\}$ assigned to a variable x, therefore, is:

$$v = (\sum_{i=1..n} (x_i - \mu)^2) / n$$

Thus, in table 1, the variance of A is $((40\text{-}58)^2 + (58 - 58)^2 + (92 - 58)^2 ... + 30\text{-}58)^2) / 10 =$ 594.44. Doing the same calculation for student B, the variance works out as 8.00. It is clear that the variability in A's run of marks is much greater than B's.

Interpretation of variance is not as straightforward as it appears to be. In the above example, what do the magnitudes mean in absolute terms? When several variances are compared, the relativities of the magnitudes reflect degrees of variation, but what if one is trying to interpret a single variance without reference to others? What, in absolute terms, does 594.44 indicate about the amount of variation in A? Is it a large variation or a small

one? The problem is that the squares quantities are not readily interpreted in terms of the original units of measurement. To recover the original units, it is only necessary to take the square root of the variance. Doing this for the above variances, the square root of 594.44 is 24.38, and for 8.00 it is 2.83. The interpretation is that, for student A, the average divergence of marks to either side of the mean is 24.38, and for B it is 2.83, the reasonableness of which a quick glance at the runs of marks will confirm. Variation expressed in terms of the original variable scale is much more intuitively meaningful than in terms of variances, which are just numbers whose only interpretable significance is the difference in magnitude. Because of their interpretability relative to the variable values on which they are based, such square roots are standardly used in preference to variances in quantifying the spread of values across a range, and are known as standard deviations.

Given a data matrix Q in which the rows are cases and the columns are lexical type variables describing the cases, and also that the aim is to classify the cases on the basis of the differences among them, the application of variance / standard deviation to dimensionality reduction is straightforward: eliminate all variables with low variance, that is, variables whose values do not vary enough for them to be useful in document classification. As with the upper and lower thresholds discussed in the preceding section, this begs the question of how low is too low, that is, of selecting a threshold. There is, however, an additional caution in using variance as a selection criterion. One might be tempted to assume that the higher the variance of a variable, the more useful it is for document classification, but this would be a false assumption. Variance only measures the spread across a variable's distribution of values, but says nothing about the shape of that distribution. Specifically, there is no systematic relationship between variance and degree of randomness in a distribution: even a very high-variance variable can be entirely random. The implication of this is discussed later in this section; in the light of that discussion, and

relative to some given data set, variance will emerge as a good way of eliminating useless variables from the data, but not as a way of ranking the remainder in terms of their usefulness.

Variance is a fundamental concept in probability and statistics, and any in a wide range of textbooks can provide additional information --see, for example, Milton & Arnold (2003).

c) Lexical frequency distribution

In 1972 Spärck-Jones proposed what was to become a standard principle in Information Retrieval: that a lexical type's usefulness is determined not by its absolute frequency across a collection, but by the pattern of variation in its frequency across the documents. To gain an intuition for this, assume a collection of documents related to the computer industry. At one end of the range are very low frequency words that, as expected, are of little or no use for document classification: a word like 'coffee' that occurs a few times in one or two documents that caution against spills into keyboards is insignificant in relation to the semantic content of the collection as a whole, and a word like 'bicycle' that occurs only once tells us only that the document in which it appears is unique on that criterion, which we already know. At the other end of the range, a word like 'computer' and its morphological variants is likely to be both very frequent across the collection and to occur in most if not all the documents, and as such is a poor criterion for classifying documents despite its high absolute frequency: if all the documents are about computers, being about computers is not a useful distinguishing criterion. In short, word frequency on its own is not a reliable classification criterion. The most useful words are those whose occurrences are, on the one hand, relatively frequent, and on the other are not, like 'computer', uniformly spread across all collection documents but rather occur in clumps, such that a relatively few documents contain most or all the occurrences, and the rest of the collection

few or none; the word 'debug', for example, can be expected to occur frequently in documents that are primarily about computer programming and compiler design, but only infrequently if at all in those about, say, word processing. On this criterion, lexical types are selected in accordance with their 'clumpiness' of occurrence across documents in a collection. The three most often used methods for determining such clumpiness are described in what follows: term frequency / inverse document frequency (TF.IDF), signal-noise ratio, and Poisson term distribution.

i. *TF.IDF*

The conceptual basis of Spärck-Jones' term TF.IDF is the notion of 'clumpiness' of lexical type occurrence introduced above: the usefulness of lexical types for classifying the documents in a collection is on a continuum from those whose pattern of occurrence is diffused across most or all the documents and are thus poor criteria for classification, to those whose pattern of occurrence is localized to a relatively few documents and are thus good criteria for classification. Specifically, TF.IDF measures the importance of a word for document classification as 'proportional to the standard occurrence frequency of each term k in each document i ($freq_{ik}$) and inversely proportional to the total number of documents to which each term is assigned' (Salton & McGill 1983, 63). IDF on its own is defined as:

$$IDF_{wD} = \log_2 (n_D / df_w)$$

where

- D is a document collection
- IDF_{wD} is the inverse document frequency of lexical type w across the total number of documents in D
- n_D is the total number of documents in D

- df_w is the number of documents in D that contain w
- $\log_2$ is not conceptually part of IDF, but merely scales n_D / df_w to a convenient numerical interval.

For example in a collection of 1000 documents, if term A occurs in 100 documents, B in 500, and C in 900, then the IDF is 3.32 for A, 1 for B, and 0.15 for C , that is, the smaller the number of documents in which a term occurs, the larger its IDF.

It is easy to see that IDF assigns the greatest importance to types that occur in only one document: for the above 1000 documents, the IDF for a word that occurs in only one document is $\log_2$ (1000/1) = 9.97, for one that occurs in two documents $\log_2$(1000/2) = 8.97, and so on. Because the definition of IDF takes account only of the number of documents that a type occurs in and not the number of times it occurs, the implication is that types which occur only once in one document are among the most important keywords in a collection. This is not only counterintuitive, but also flatly contradicts both Luhn's principle that very infrequent words make poor keywords for document classification and standard practice, based on Luhn, of eliminating all frequency-1 types as a first step in keyword selection. The problem is resolved by modifying the definition of IDF so as to take lexical type frequency into account. This modification is known as term-frequency IDF, or TF.IDF:

$$\text{TF.IDF}_{wD} = f_w \log_2 (n_D / df_w)$$

where f_w is the frequency of w across all documents in D, and the other terms are as for IDF. A type that occurs 10 times in a single document thereby has a TF.IDF ten times as large as a word that occurs only once, which is both intuitively satisfying and also compatible with Luhn.

Given, then, a data matrix Q in which the rows are cases and the columns are lexical type variables describing the cases, and given also that the aim is to classify the cases on the basis of the differences among them, the application of TF.IDF to keyword selection is straightforward: calculate the TF.IDF for each column of Q and retain only those columns with a TF.IDF above a specified threshold; as before, selection of a suitable threshold is a subjective matter.

For discussions of TF.IDF see: Spärck-Jones (1972, 1974), Roberston & Spärck Jones (1976), Salton & McGill (1983, 63), Salton & Buckley (1988), Buckley (1993), Baeza-Yates & Ribeiro-Neto (1999, 29-30), Belew (2000, 84-5), Spärck Jones *et al.* (2000), Spärck Jones *et al.* (2003), Robertson (2004), Robertson & Spärck Jones (2004), Spärck-Jones (2004).

ii. *Signal-noise ratio*

This dimensionality reduction technique is based on ideas from information theory, which is concerned with issues in the transmission of signal sequences generated by a signal source over some channel to a signal receiver; for present purposes, the signals are taken to be discrete, that is, symbols.

- At any point in a sequence, the receiver is in a state of uncertainty about what the next symbol will be. When the next symbol is received, the uncertainty decreases. The amount of decrease of uncertainty is the information that the symbol provides to the receiver.
- The total information, in the above sense, across all possible symbols that the source can generate is the entropy of the source.

These notions of uncertainty and entropy need to be made more precise before dimensionality reduction based on information theory can be described.

Consider a signal source that emits tokens of whatever symbol types it can emit with equal probability. If it can emit only one symbol type, say 'A', so that every sequence is just a string of 'A'-tokens, there is no uncertainty at all about what the next symbol will be. If it can emit two symbol types 'A' and 'B', there is some uncertainty, if three 'A', 'B', and 'C' there is more uncertainty, and so on: the more symbol types a signal source can emit tokens of, the greater the uncertainty about what the next will be. Now, in the first case, the probability of the single symbol type is 1/1 = 1, in the second it is 1/2 = 0.5, in the third it is 1/3 = 0.333..., and so on. Uncertainty increases as probability decreases, that is, uncertainty is inversely proportional to probability. For any given symbol *s* emitted by a signal source, therefore,

$$u_s = 1 / p_s$$

where p_s is the probability of *s*. In information theory it is usual to express uncertainty in terms of binary digits or bits, and so this expression becomes

$$u_s = \log_2 (1 / p_s)$$

which can be rewritten as

$$u_s = -\log_2 (p_s)$$

And, because the information provided by a symbol from a given source is defined as the decrease in the receiver's uncertainty when it appears, it follows from the above that, the lower a symbol's probability and thus the higher its uncertainty, the greater the information it carries. This makes intuitive sense: if the receiver knows to expect only one of two

possible symbols there is a little uncertainty, but if there are 100 possible symbols there is much more, and the appearance of the next symbol in the latter case resolves much more uncertainty than in the former.

Extension to signal sources in which the symbol types have different probabilities of being emitted is a straightforward matter of weighting the uncertainty associated with a given symbol *s* by its probability:

$$u_s = - p_s \log (p_s)$$

This formulation of uncertainty includes equal-probability sources as a special case, and is used in what follows.

Finally, the total uncertainty across all *n* possible symbol types that can be emitted by a source is:

$$H(x) = \sum_{i=1..n} - p_i \log_2 (p_i)$$

where *x* denotes an event, that is, the emission of a symbol by the source, and H is the entropy of the source.

Now, Salton & McGill (1983, 63-6) pointed out that the more evenly distributed the occurrence probabilities are spread across symbols, the higher the entropy. For example, Table 2 shows three different lexical frequency vectors together with their corresponding probabilities and entropies.

Lexical frequency vectors	Probability	Entropy
[2 2 2 2 2]	0.2 0.2 0.2 0.2 0.2	2.32
[1 1 1 3 4]	0.1 0.1 0.1 0.3 0.4	2.05
[1 1 1 1 6]	0.1 0.1 0.1 0.1 0.6	1.77

Table 2: Lexical frequency vectors with corresponding probabilities and entropies

Salton & McGill also pointed out that this observation can be adapted to identification of clumpiness of lexical type occurrence across a document collection. The higher the entropy of a column in the frequency matrix, the more evenly distributed the frequencies are, and thus the less useful the lexical type associated with that column is for differentiating the documents in the collection; conversely, the lower the entropy, the more clumpy the distribution, and the more useful the variable. They rename entropy as 'noise', and define the 'signal' for any given column *k* of the frequency matrix as an inverse function of the noise:

$$signal_k = \log_2(totalfrequency_k) - noise_k$$

where $totalfrequency_k$ is the sum of frequencies in column *k*. The signals for all the columns in the matrix are calculated and sorted into descending order of magnitude and all the columns with signals below a specified threshold are eliminated. Again, selection of a suitable threshold is subjective.

For an accessible introduction to information theory see Pierce (1980); more mathematical is MacKay (2003). Application to term weighting in information retrieval, and thus closely related to the present concern with dimensionality reduction, is described in Salton & McGill (1983, 63-6) and Belew (2000, 83-4).

iii. *Poisson term distribution*

The term distribution approach to variable selection assesses the clumpiness of lexical type occurrence in a document collection by the degree to which words are non-randomly distributed across the documents: at one extreme, a type that is randomly distributed across all the documents is held to be useless for classification and can be eliminated, and on the other a type that has a non-random pattern of occurrence, that is, once that occurs frequently in a relatively small subset of documents and little or not at all in others, is held to be a useful classification criterion and is retained as a keyword. In this approach, the degree of randomness in a type's occurrence is usually assessed in terms of the degree to which its pattern of occurrence fits one of the standard distributions in statistics, the Poisson.

In statistics, the Poisson distribution is used to model the number of times that a random and rare event occurs in some specified spatial or temporal interval (Clarke & Cooke 1998, ch. 20). More specifically, it models data generated by physical stochastic processes, where a physical stochastic process is one that generates events randomly over some interval of time or space in a domain of interest ----the fluctuation in share values on the stock market over a given week, for example. A Poisson process is a stochastic process in which (i) the random events occur independently of one another, and (ii) the probability of occurrence of the random events over some designated interval i is described by the following probability density function P:

$$P(X = r) = (e^{-\lambda}\lambda^{r}) / r!$$

where:

- The symbol P stands for 'probability'

- X is a discrete random variable
- r is the number of events that occur over an interval i
- e is the base of the natural logarithm 2.71828...
- λ is the average value of X over many intervals i

For a Poisson process whose average rate of occurrence of events λ over the designated interval i is known, therefore, this function gives the probability that some number r of events occur over i. For example, assume that 7 cars pass through a rural intersection on Thursday, 3 on Friday, and 5 on Saturday; the average number of cars λ passing through the intersection on a given day is 5. What is the probability that 4 cars will pass through on Sunday?

$$P(X = 4) = (2.72^{-5} \times 5^4) / 4!$$

$$= (0.0067 \times 625) / 24$$

$$= 0.175$$

Table 3: Example calculation of Poisson probability

The Poisson distribution can be used to test whether the values in a given data variable are random or not: if there is a close fit between the data and the theoretical distribution, it is probable that the data was generated by a random process. This applies to identification of superfluous words in document collections as follows:

1. The production of a text document, and more specifically the successive occurrence of words which constitute the document, is taken to be a Poisson process. A document collection D generated by a Poisson process and containing m documents is assumed.
2. If D contains n word types, then each type w_i (for $i = 1..n$) is a random variable x_i.

3. For any given x_i

- The designated intervals of interest are the m individual documents.
- The average rate of occurrence λ_i of w_i in the m documents is the total number of occurrences of w_i in D divided by m.
- The actual number of occurrences of w_i in a document $d \varepsilon$ D is r_i
- The question being asked with respect to the random variable x_i is: given that each document d is expected, on average, to contain λ_i tokens of w_i, how probable is the actual number of occurrences r_i in each of the $d \varepsilon$ D?
- If the probability of x_i is high across all $d \varepsilon$ D, then it fits the Poisson distribution, that is, the occurrence of w_i is random and it can therefore be eliminated. If, however, the probability of x_i is low for one or more of the documents, then the fit of x_i diverges from the Poisson distribution --in other words, w_i occurs nonrandomly to a greater or lesser degree and thereby becomes a candidate keyword. In the ideal case, the probability of x_i is low for a proper subset of D, and high elsewhere, indicating that occurrence of w_i is nonrandom in some documents and random in the remainder, and therefore a good criterion for document categorization.

It is, of course, absurd to suggest than any document or document collection is generated as a stochastic process. Writers are not the proverbial monkeys sitting at keyboards; authors choose the words they use for highly specific reasons. As such, human-generated documents violate a fundamental Poisson assumption: that events --word type occurrences-- are independent, and it is therefore not to be expected that the distribution of any word type in a natural language text has a true theoretical Poisson distribution. Empirical results have, however, shown that several categories of word type are almost-Poisson:

- The so-called 'function' words like determiners, prepositions, and conjunctions, which occur at a more or less constant rate across documents in any given language.
- 'Content' words, that is, non-function words, which occur very frequently across all documents in a collection.
- Content words that are very infrequent across documents in a collection.

Given a set of candidate word types extracted from a document collection, how can one determine which of them is Poisson or near-Poisson? A characteristic of theoretical Poisson distributions is that their means and variances are identical (Clarke & Cooke 1998, 472-3). Given a frequency matrix whose columns represent the variables of interest, therefore, the degree to which any column / variable *j* diverges from Poisson can be determined by calculating the degree to which *j*'s mean and variance differ. Because, as noted, the Poisson is not an ideal model for lexical type distribution in natural language text, exact correspondence cannot in general be expected, but the relative degree of divergence from mean-variance equivalence can be used to distinguish lexical types on a continuum of randomness, ranging from those that are near-Poisson and can therefore be eliminated to those that are far from Poisson and should therefore be retained as keywords.

To apply term distribution to dimensionality reduction, the disparities between mean and variance for all columns in Q are calculated and columns below a specified (and subjective) threshold value are eliminated.

On the theory of Poisson distribution see Clarke & Cooke (1998, ch. 20). On Poisson distribution as a model for distribution of lexis in document collections see Bookstein & Swanson (1974), Bookstein & Kraft (1977), Croft & Harper (1979), van Rijsbergen (1979, 27-9), Roberston & Walker (1994), Church & Gale (1995a, 1995b), Belew (2000, 73 ff).

5.1.6.2.3 Variable redefinition

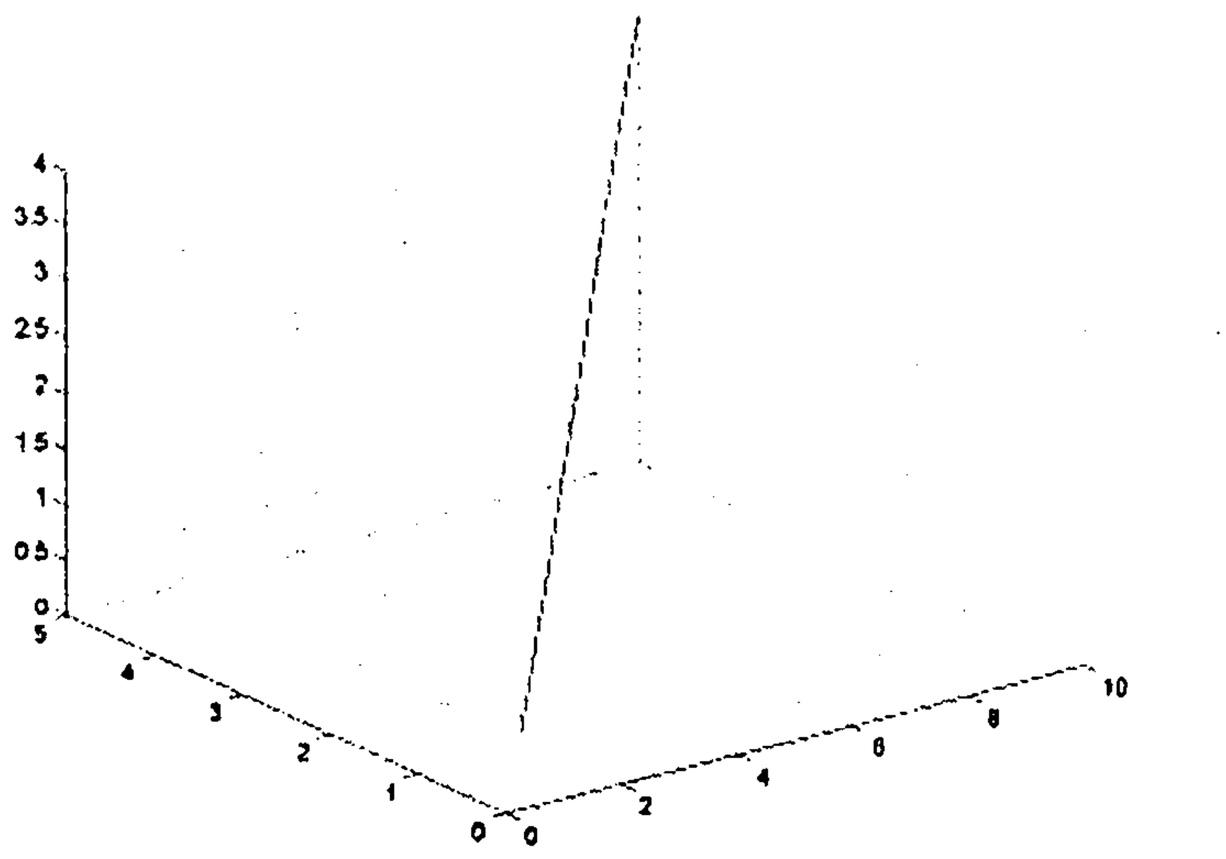
Dimensionality reduction can be achieved by replacing the variables that have been chosen to describe the domain of interest with different variables that describe the domain as well as, or almost as well as, the originals, but are fewer in number. In describing how this is done, the present section first looks at some underlying concepts, and then gives a detailed account of two such variable redefinition methods.

a) Concepts

We have seen that a data set of n-dimensional vectors defines a manifold in *n*-dimensional space, where *n* is a positive integer. In such an *n*-dimensional space, it is possible in principle to have manifolds whose dimensionality is *k*, where $k < n$. Consider the 3-dimensional data set in 6a:

v1	v2	v3
1	0.5	0.4
2	1	0.8
3	1.5	1.2
4	2	1.6
5	2.5	2
6	3	2.4
7	3.5	2.8
8	4	3.2
9	4.5	3.6
10	5	4

a



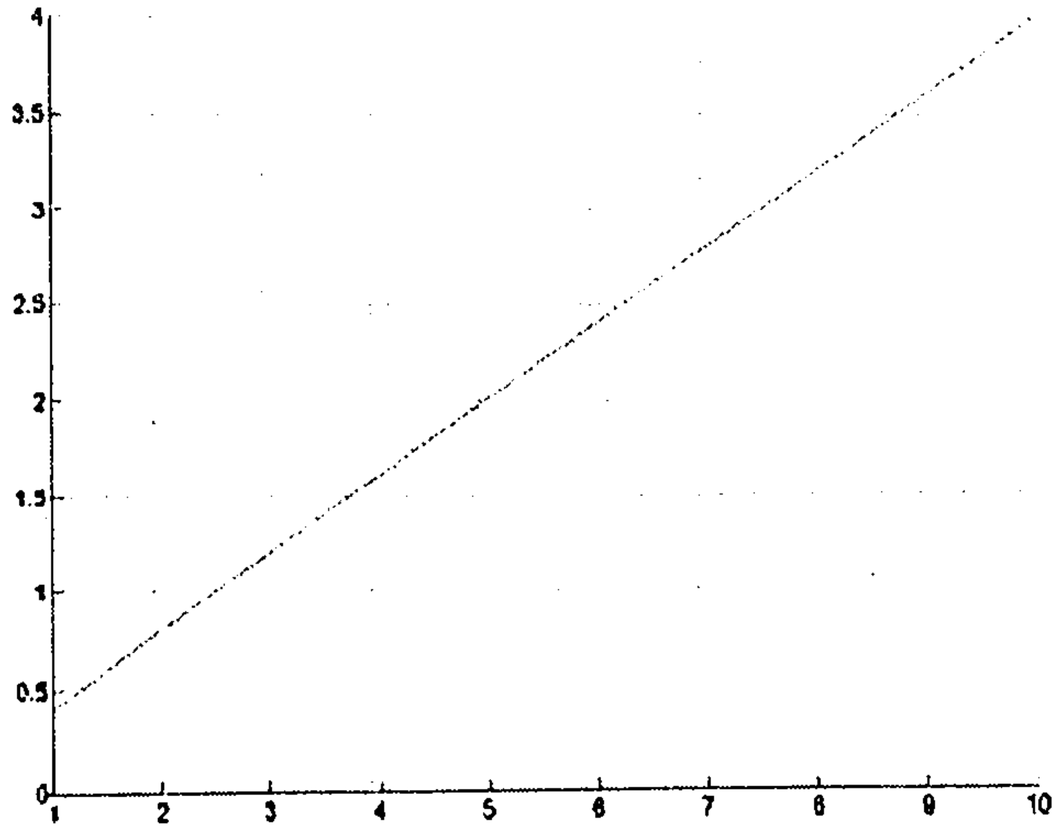
b

Figure 6: A one-dimensional manifold in 3-dimensional space

Plotting this data in 3-dimensional space (Figure 6b) shows it to describe a line. But that line can be redescribed in 2 dimensions:

v1	v2
1	0.4
2	0.8
3	1.2
4	1.6
5	2
6	2.4
7	2.8
8	3.2
9	3.6
10	4

a



b

Figure 7: A one-dimensional manifold in 2-dimensional space

In fact, the line can be redescribed in 1 dimension -- its length 10.63-- by its distance from 0 on the real-number line:

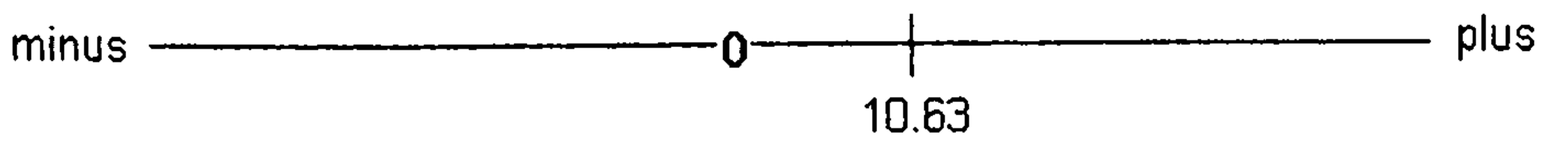


Figure 8: A one-dimensional manifold in 1-dimensional space

Consider another example --a plane in 3-dimensional space:

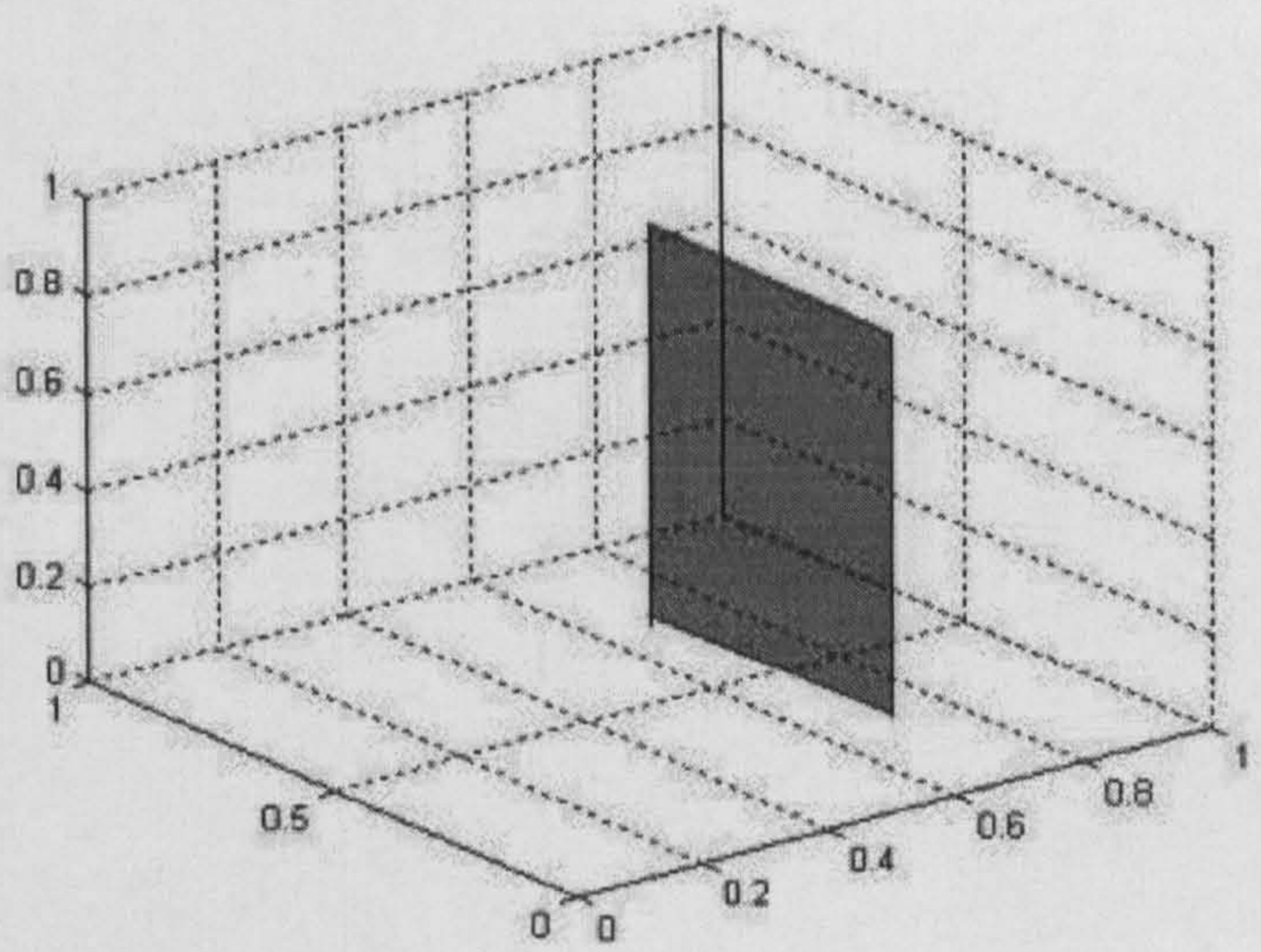


Figure 9: A two-dimensional manifold in 3-dimensional space

This plane can be redescribed in 2-dimensional space

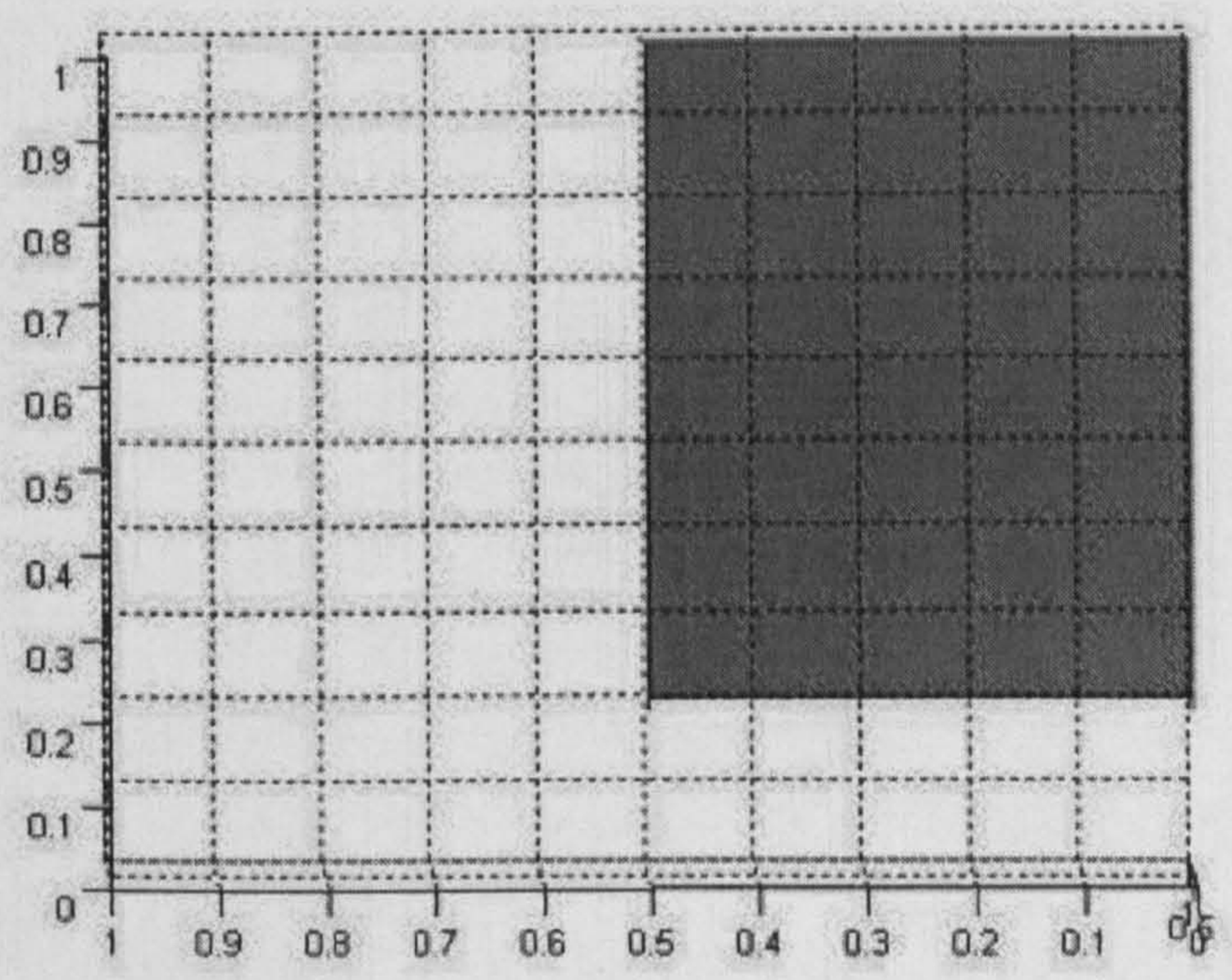


Figure 10: A two-dimensional manifold in 2-dimensional space

And, as usual, this concept extends straightforwardly to any dimensionality.

In general, therefore, a line can be described in one dimension, two dimensions, three dimensions, or any number of dimensions one likes. Essentially, though, it is a 1-dimensional object, its 'intrinsic dimensionality' (Verleysen 2003) is 1. In other words, the minimum number of dimensions required to describe a line is 1; higher-dimensional descriptions are possible but unnecessary and, in a sense, redundant. A plane was described in two and three dimensions. Could it also, like a line, be described in one dimension? No: the intrinsic dimensionality of a plane is 2 --the corresponding data set must be 2-dimensional at least, giving the coordinates of the points that describe it. Similarly, the intrinsic dimensionality of a cube is 3, in that that minimum-dimensionality data set that can describe it is the three *x*, *y* and *z* coordinates of the points that comprise it. A cube can of course, exist not only in 3-dimensional space but also in 4, 10, 20, and *n*-dimensional spaces, in which case it would be a *k*-dimensional manifold of intrinsic dimensionality $k = 3$ embedded in n-dimensional space, where $k < n$.

The concept of intrinsic dimensionality applies straightforwardly to dimensionality reduction. The informational content of data is conceptualized as a *k*-dimensional manifold in the *n*-dimensional space defined by the data variables. Where $k = n$, that is, where the intrinsic dimensionality of the data corresponds to the number of data variables, no dimensionality reduction is possible without significant loss of information. However, the foregoing discussion of data creation noted that, when describing a domain of interest, selection of variables is at the discretion of the researcher. It is therefore possible that the selection of variables in any given application will be suboptimal in the sense that there is redundancy among the variables, that is, that they overlap with one another in terms of the information they represent about the domain; where there is a significant amount of redundancy, it is possible in principle to represent this information using a smaller number of variables, thus reducing the dimensionality of the data. In such a case, the aim of

dimensionality reduction of data is to discover its intrinsic dimensionality k, for $k < n$, and to redescribe its informational content in terms of those k dimensions.

For example, consider some invented employee data:

	Height (metres)	**Weight (kilograms)**	**Salary scale (1..18)**	**Income (Euros/month)**	**Satisfaction level (1..10)**
Employee1	1.70	75.7	3	900	3
Employee2	1.73	76.2	12	3600	7
Employee3	1.58	72.2	14	4200	8
Employee4	1.91	81.8	7	2100	6
Employee5	1.86	78.7	10	3000	5
Employee6	1.79	79.7	13	3900	6
Employee7	1.85	80.1	3	900	3
Employee8	1.82	79.8	2	600	4
Employee9	1.64	73.5	7	2100	5
Employee10	1.68	75.6	6	1800	4

Table 4: Example data

Even cursory examination of this data reveals considerable redundancy between and among variables. This redundancy emerges clearly when one plots pairs of variables against one another. The salary scale / income plot, for example, looks like this:

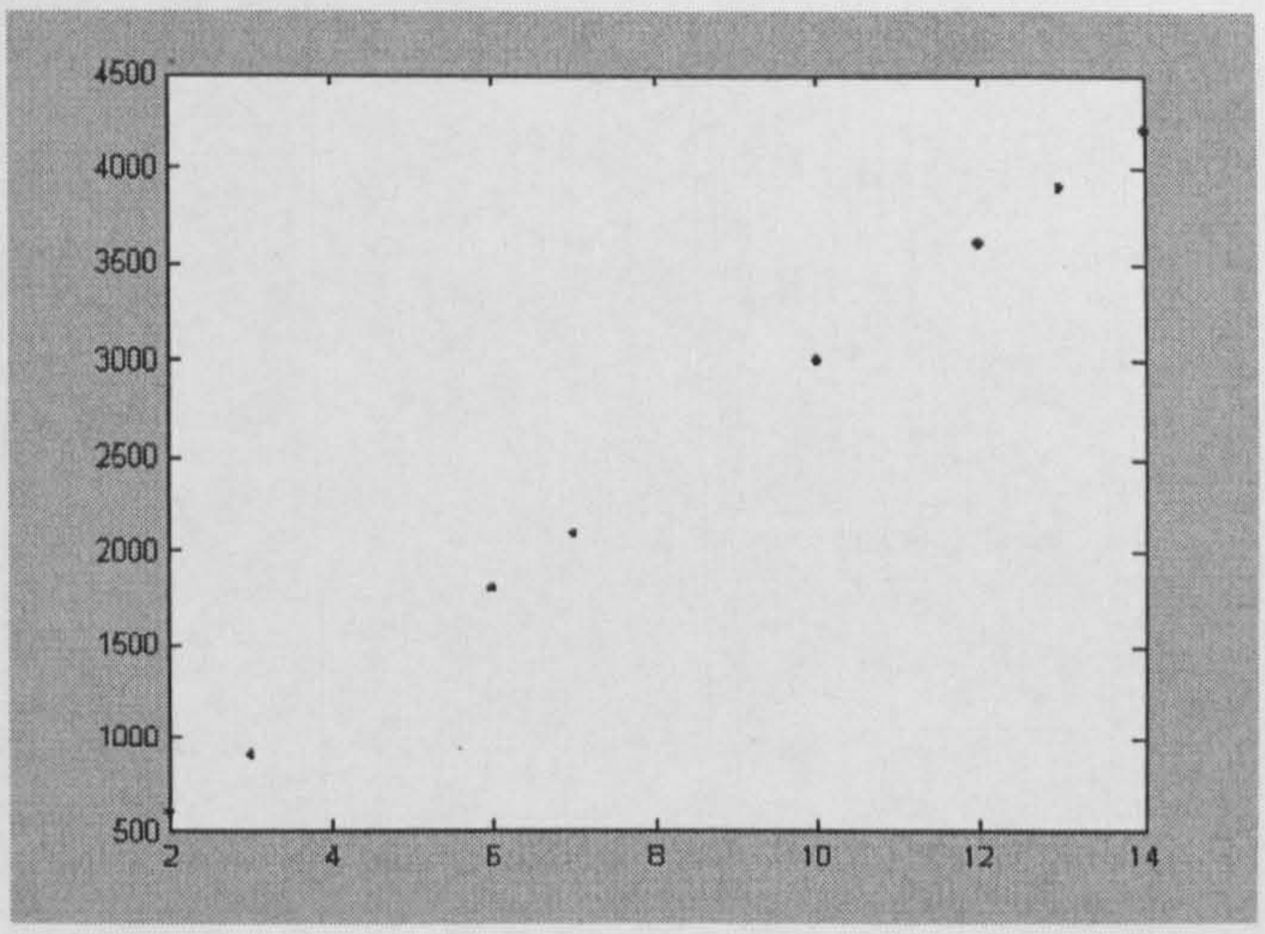


Figure 11: Relationship between salary scale and income in example data

There is a perfect linear relationship between salary scale and income: to know the salary scale is to know the income, and vice versa. These two variables capture exactly the same information about each employee, and are thus redundant. One of them can be completely eliminated without any loss of information.

If one plots height against weight the relationship is less exact, but there is still a clear tendency in the arrangement of data points which says that, as height increases, so in general does weight, though with some variation. This general tendency is well captured by the line drawn through the data points. Because there is a strong relationship between these variables, they too are largely redundant, and virtually all the information about employees which they contain could be captured by a single variable, called something like 'size', whose value for each of the employees would be on the line.

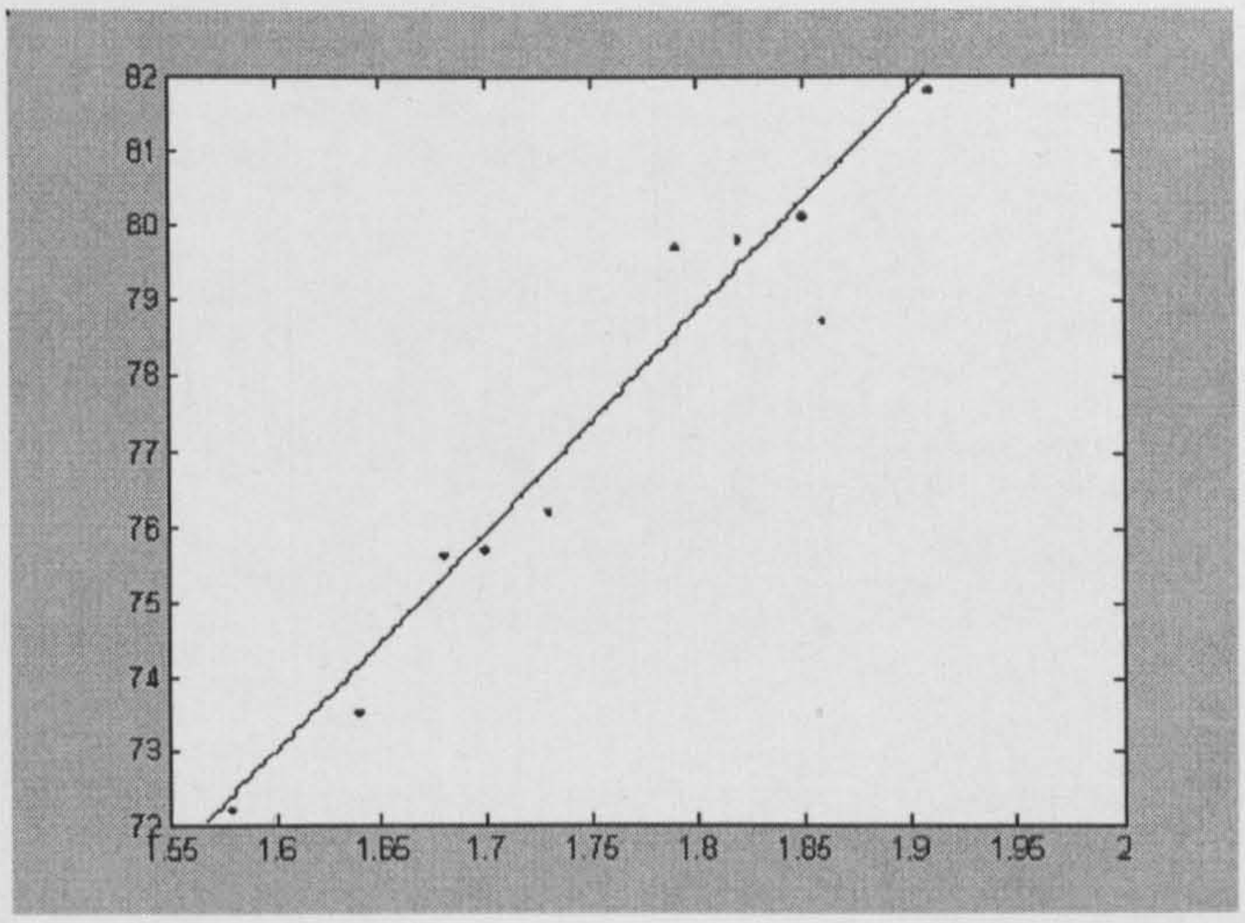


Figure 12: Relationship between height and weight in example data

Plotting salary scale against satisfaction level shows a much looser relationship that might just conceivably be captured by the line, but to amalgamate satisfaction level with salary scale and salary point into a single variable called something like 'quality of life', as was done for height and weight > size, would probably lose too much information; satisfaction level should probably be maintained.

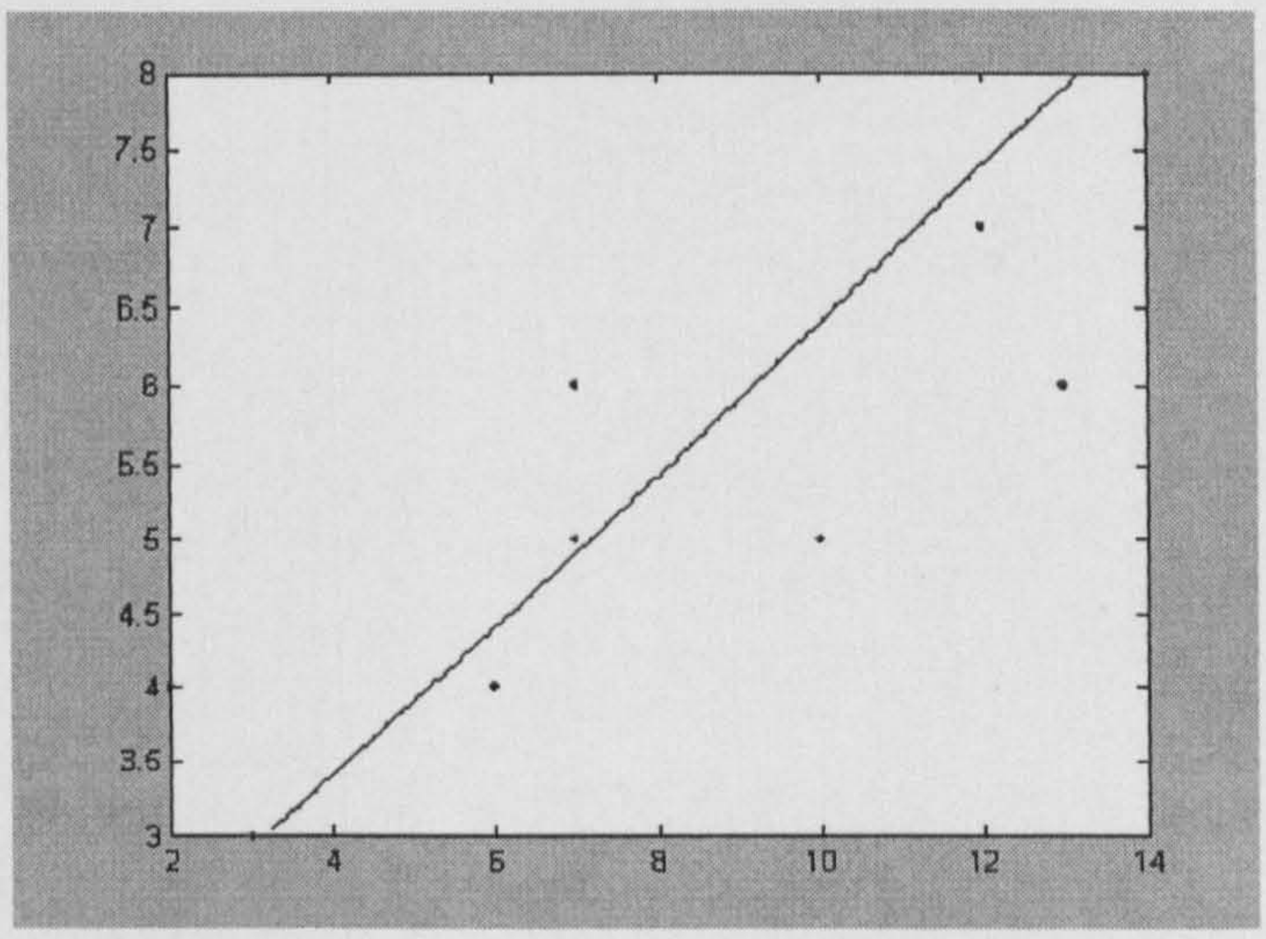


Figure 13: Relationship between salary scale and satisfaction level in example data

And, as a final example, if one plots height against satisfaction level there is no obvious relationship at all, and therefore little or no redundancy between these variables. They capture different information about employees, and must be retained separately.

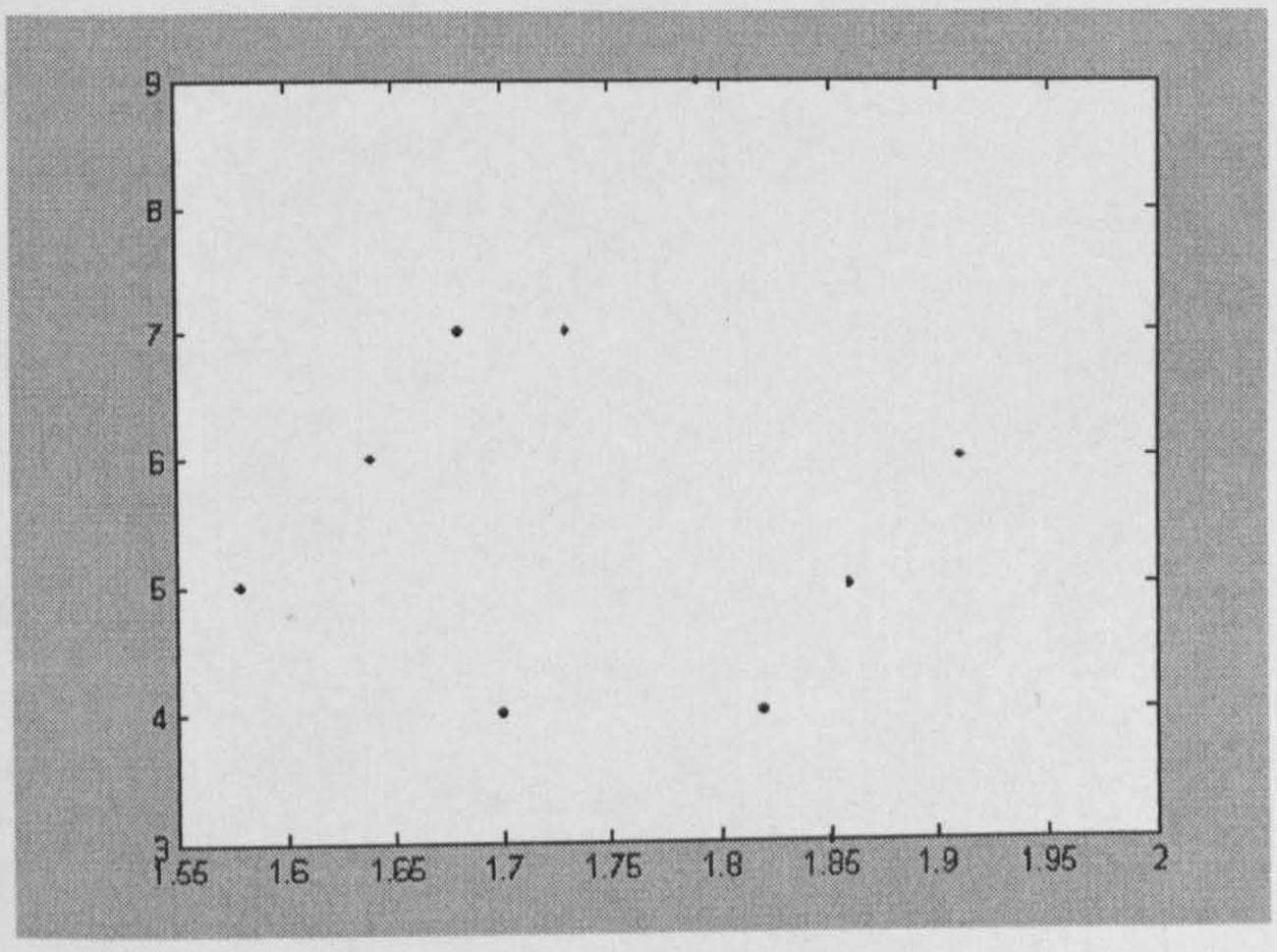


Figure 14: Relationship between height and satisfaction level in example data

On the basis of these and similar possible plots, then, one could redescribe the employees originally described by 5 redundant variables with 3 less redundant or possibly even non-redundant ones -- 'monthly salary', 'size', and 'satisfaction level'-- with minimal loss of information by combining closely related variables into a single variable that represents most or all the information that the original variables contained. The intrinsic dimensionality of this data manifold, in other words, is 3, and it was originally embedded in a 5-dimensional space.

The remainder of this section describes two techniques for discovering the intrinsic dimensionality k of n-dimensional data and, if $k < n$, of redescribing that data in k dimensions.

b) Principal Components Analysis (PCA)

This section outlines the *de facto* standard technique for dimensionality reduction by variable redefinition: principal components analysis (PCA). Its aim is to transform a set of correlated variables into a --usually smaller-- set of uncorrelated ones. The discussion of PCA is in two parts: the first part introduces some concepts necessary for understanding PCA, and the second then describes PCA itself.

i. *Covariance and correlation*

Given two variables x and y, covariance is the extent to which, as the values of x change, those of y change correspondingly:

$$\mathrm{cov}(x,y) = \sum_{i=1..n} \frac{(x_i - \mu_x)(y_i - \mu_y)}{n}$$

where n is the given number of values of x and y, and μ_x and μ_y are the means of x and y respectively. Covariance is a quantification of the intuitive notion of relatedness: a positive covariance indicates that there is a positive relationship between the variables, i.e., as one increases or decreases, so does the other increase or decrease, whereas a negative covariance indicates a negative relationship in which, as one variable increases, the other decreases, and vice versa. The magnitude of the covariance indicates the relative strength of their relatedness.

Division of the covariance of x and y by the product of their standard deviations yields their correlation coefficient ρ:

$$\rho(x,y) = \frac{\mathrm{cov}(x,y)}{\sigma_x \sigma_y}$$

where σ_x and σ_y are the standard deviations of x and y respectively. Like covariance, the correlation coefficient is a measure of the degree to which the variables co-vary, but it has the considerable advantage that scaling problems do not apply. Correlation coefficients are always in the range -1..1. This means that, on the one hand, the absolute magnitude of the correlation coefficient can readily be interpreted as the degree of relationship between two variables: -1 indicates that the variables are perfectly negatively related, 0 that they are completely unrelated, 1 that they are perfectly positively related, and all degrees in between. And, on the other, correlation coefficients for multiple variable pairs can usefully be compared because they are all on the same scale. On covariance and correlation see any statistics textbook --for example, Clarke & Cook (1998).

ii.*Vector space basis*

Given any set V of n k-dimensional vectors v_1, v_2...v_n and any set S of n scalars s_1, s_2, ... s_n:

- Multiplication of any vector $v_i \varepsilon$ V by any scalar $s_j \varepsilon$ S (for $i, j = 1..n$) is defined as $v_{new} = (s_j \times v_{i,1}) + (s_j \times v_{i,2}) + ... + (s_j \times v_{i,k})$. The effect is to alter the length of the vector. For example, for $k = 2$, $v_i = [0.4\ 0.4]$, and $s_j = 2$, v_i is shown in Figure 15a, and v_{new} in Figure 15b:

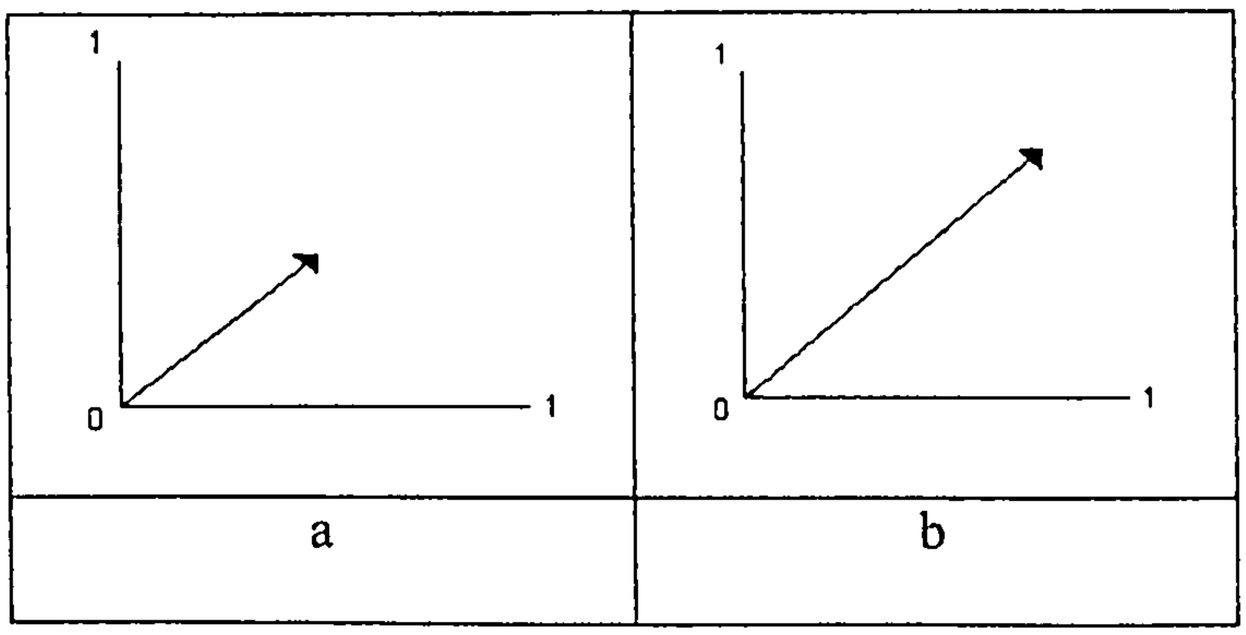


Figure 15: Vector-scalar multiplication

- Two vectors v_i and v_j are linearly independent if neither is a scalar multiple of the other. For v_i = [0.4 0.4] and v_j = [0.5 0.3]:

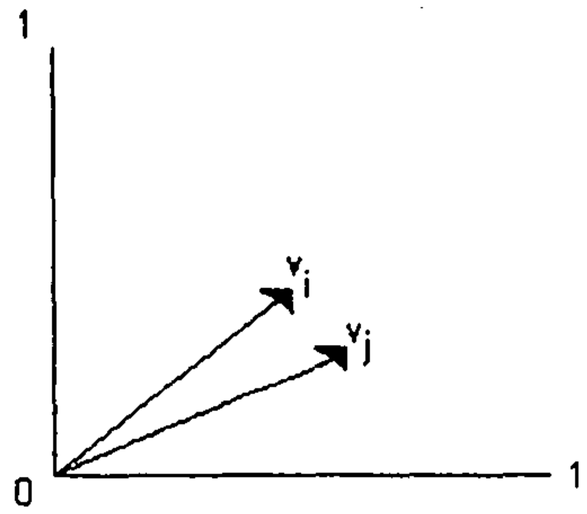


Figure 16: Linearly independent vectors

- A linear combination of any two vectors v_i and v_j is itself a vector defined as v_{lc}= v_i + v_j. For v_i = [0.4 0.4] and v_j = [0.5 0.3], v_{lc} = [0.9 0.7]:

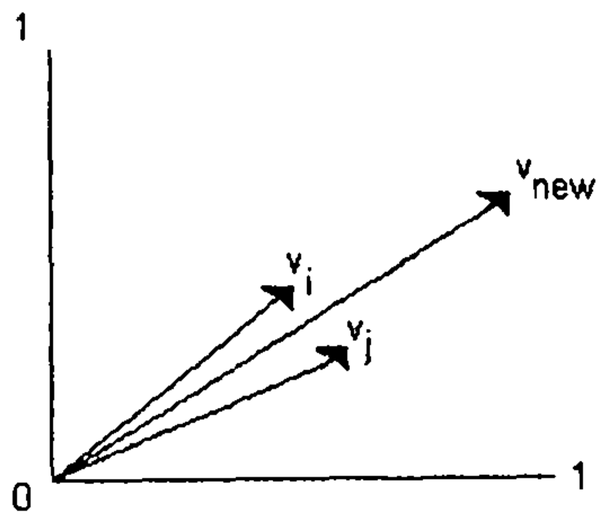


Figure 17: Linear combination of vectors

- A basis for V is any set B of k linearly-independent k-dimensional vectors. B is called a basis for V because all the vectors in V can be generated from those in B by linear combination of the $b_i \varepsilon$ B.
- There is an infinite number of bases for V. The present discussion is interested one particular class of bases: those in which the vectors are perpendicular or, in standard terminology, orthogonal. Figure 18a shows a two-dimensional basis and Figure 18b a three-dimensional one; the idea extends to any dimensionality.

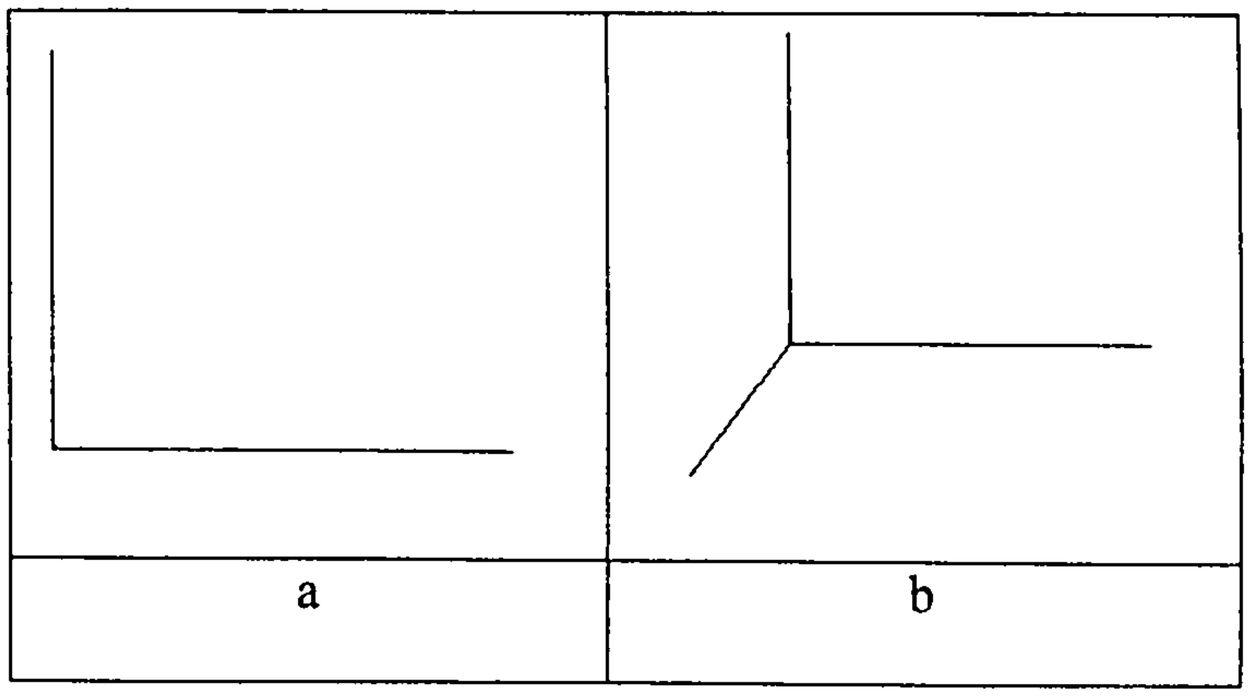


Figure 18: Orthogonal bases

For the above ideas, see any textbook on linear algebra such as Fraleigh & Beauregard (1995).

iii. *Linear regression*

A classic problem in statistics is to model the relationship between variables on the basis of observed values. Assume a two-dimensional data set of variables 'height' and 'weight' in a sample of humans. A plot of such a data set might look like this:

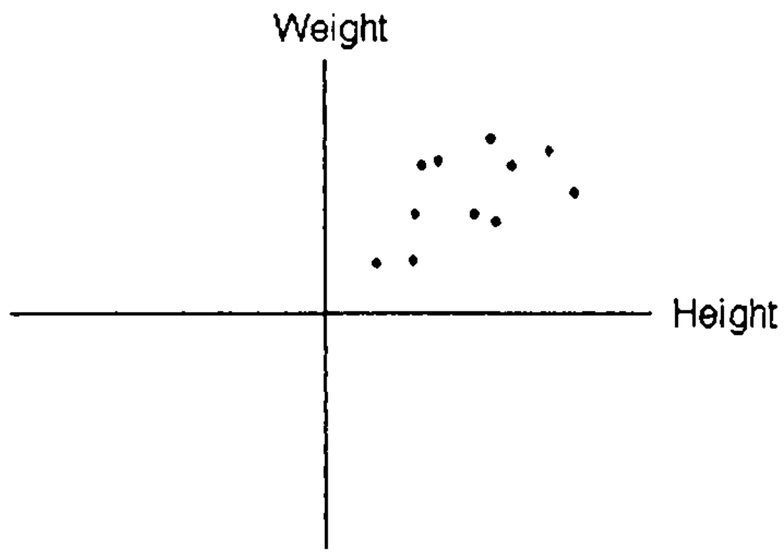


Figure 19: 'Height' and 'weight' data plot

The simplest interpretation of this data is that there is a systematic and more specifically linear relationship between height and weight --as height increases so does weight-- and this is modelled by drawing a straight line through the points in a way that best captures the

relationship among them. The problem is that there is a theoretically infinite number of lines that can be drawn, and it is not obvious to the eye which of them best captures the relationship between the variables. For example:

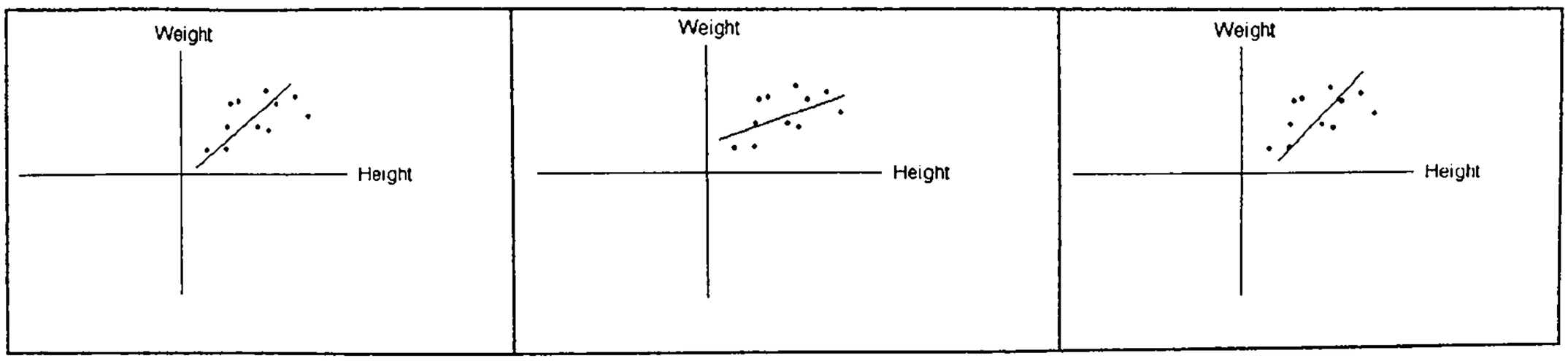


Figure 20: Possible linear models for data

What is required is an objective measure of the 'goodness of fit' of a line to the data points. The most frequently used measure is the least-squares criterion: the line is chosen such that the sum of the distances *d1* + *d2* + ...+ *d11* between the data points to the line is minimized:

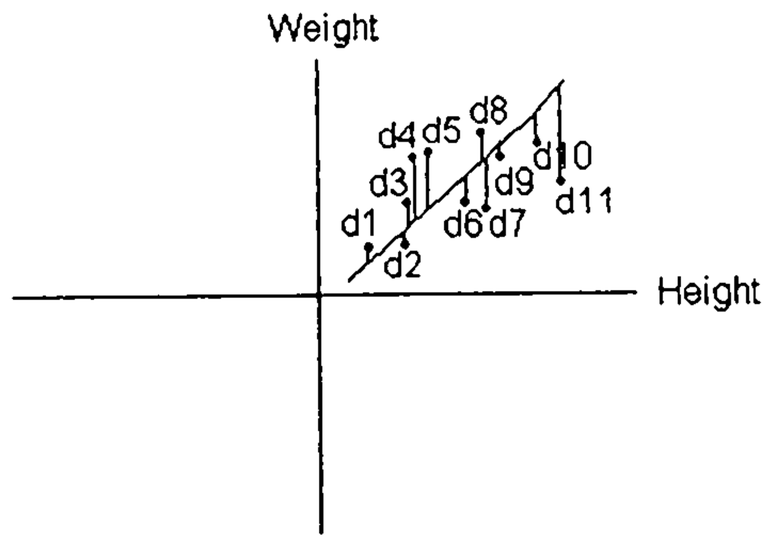


Figure 21: Least squares fit of linear model to data

The method for calculating this minimum is not directly relevant here and is thus omitted, though it is accessible in any statistics textbook. The essential point is that linear regression provides a way of calculating a line of best fit to a set of data points.

iv. *Principal components analysis*

This section is in two subsections. The first gives a graphically-based sketch of how PCA works, the aim of which is to provide an intuitively-accessible introduction. The second then goes on to describe the numerical method by which principal components are calculated.

Assuming the standard two-dimensional Cartesian basis, and given a data set of two-dimensional vectors with dimensions x and y, a plot showing the relationship of the vectors might look like this:

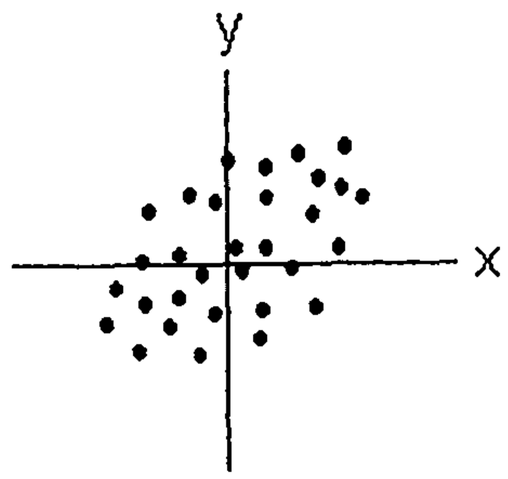


Figure 22: Two-dimensional data distribution with orthogonal basis

It is possible to find a different orthogonal basis for this distribution such that the axes are a best fit for the main directions of variation in the data. The line of best fit X' is drawn through the data, and the line of second-best fit Y' along such that Y' is orthogonal to X':

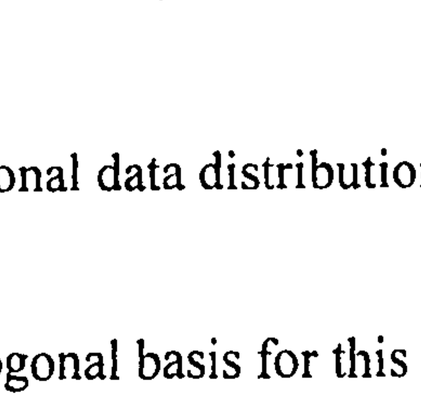


Figure 23: Alternative orthogonal basis for data

The data vector coordinates are then recalculated relative to the new basis. In terms of dimensionality reduction this doesn't get us any further, since it merely restates the original data in two dimensions with reference to a different orthogonal basis. Consider, however, the following distribution in which the data vectors are highly correlated:

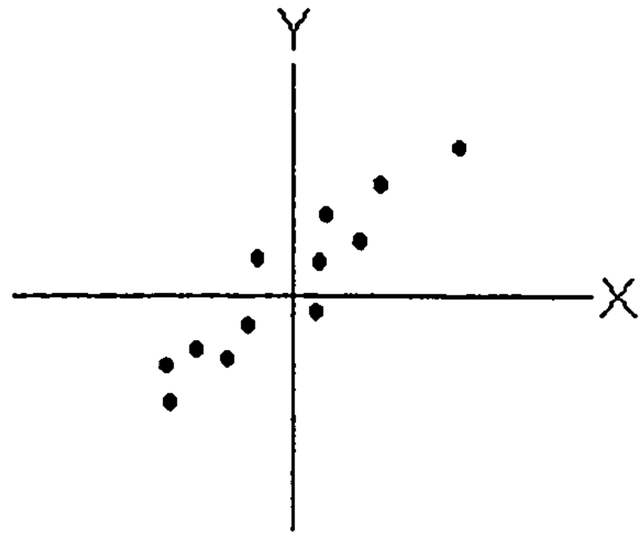


Figure 24: Highly correlated two-dimensional data distribution with orthogonal basis

If the orthogonal lines of best and second-best fit are drawn here,

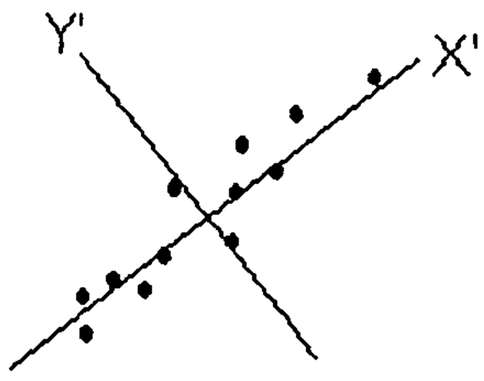


Figure 25: Alternative orthogonal basis for data

then it is clear that X' captures almost all the variation in the data, and Y' only a small amount. If Y' is simply disregarded, then the data can be restated in 1 rather than the original 2 dimensions with minimal loss of information, and the data dimensionality has been reduced.

This idea extends to any dimensionality, though it cannot be shown for dimensionality > 3. In the three-dimensional case (26a), the first two dimensions Z' and Y' are sufficient to represent the data, achieving a dimensionality reduction of 3 to 2, and in case (26b) the dimensionality can be reduced to 1 by using only the Z' dimension.

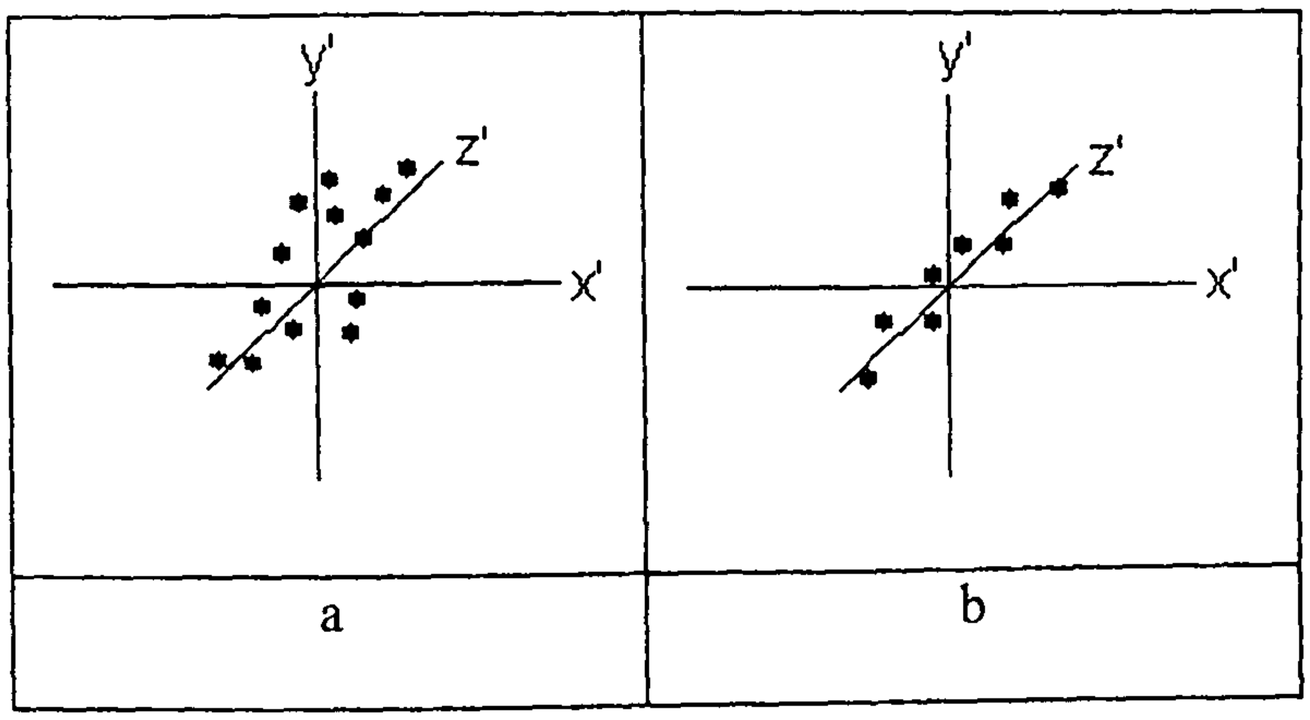


Figure 26: Three-dimensional data distribution with orthogonal basis

Relative to an *n*-dimensional data set D, then, the essence of PCA is this:

- An *n*-dimensional orthogonal basis for D is constructed, such that each axis is the least-squares best fit to one of the *n* directions of variation in D.
- The axes along which there is relatively little variation are eliminated, leaving an *m*-dimensional basis for D, where $m < n$.
- The original *n*-dimensional data D is projected into the reduced *m*-dimensional space, which yields a data set D' that is dimensionality-reduced but still contains most of the variability in D.

The foregoing intuitive sketch has obvious shortcomings that render it inadequate in the general case: the graphical approach is only applicable to data dimensionality 3 or less, and nothing is said about the specifics of how the basis is constructed, the criterion for axis elimination, or projection of the original data into reduced-dimensionality space. What is

required is a numerical approach, and PCA provides it. PCA proceeds in the following stages:

1. Construction of a covariance matrix

Designation of relatedness between and among variables has thus far relied on the colloquial semantics of words like 'redundancy' and 'variability', and on graphical representation of relative proximity in space. PCA provides a precise definition of this relatedness via the statistical concept of covariance, and makes this concept the basis for dimensionality reduction. Specifically, PCA is based not on the given data matrix D, but on a matrix C of covariances between the variables of D. The first step in PCA is to calculate these covariances and to store them in C, where C is an n x n square matrix in which both the rows i and the columns j represent the variables in the original data, and cell C_{ij} represents the covariance between variable i and variable j.

2. Construction of an orthogonal basis for the covariance matrix

An orthogonal basis for the n x n covariance matrix C is constructed incrementally as follows:

- The first axis w_1 of the basis is the vector of best least-squares fit along the direction of greatest covariance in C
- The second axis w_2 is the vector of best fit along the second-greatest direction of covariance in C and orthogonal to w_1.
- The third and subsequent axes $w_3..w_n$ and vectors of best fit along the third to the n'th greatest directions of covariance in C, such that each is orthogonal to all other w_i for i = 1..n.

The usual method for calculating such a basis is to generate the *n* eigenvectors of the correlation matrix C:

$$[E_1\ E_2] = \text{eig}\ (C)$$

where E_1 is a square matrix of the same dimensionality as C whose columns are the eigenvectors of C, E_2 is a square matrix of the same dimensionality as C whose positive diagonal contains the eigenvalues corresponding to the eigenvectors in E_1, and eig is a function that calculates E_1 and E_2 from C; the significance of the eigenvalue matrix E_2 is discussed in the next section.

3. Selection of dimensions

The basis for an *n*-dimensional set of vectors is *n*-dimensional; applied to the *n* x *n* covariance matrix C, there are *n* eigenvectors. To achieve dimensionality reduction, a way has to be found of eliminating the axes that lie along relatively insignificant directions of covariance. The eigenvalue matrix provides the criterion for this: each eigenvector is associated with one and only one eigenvalue, and the magnitude of any given eigenvalue is the length of the corresponding eigenvector. The eigenvalues are therefore sorted, and all the eigenvectors whose eigenvalues are less than some specified threshold are eliminated, yielding an *n* x *m* eigenvector matrix E_1'; selection of an appropriate threshold is discussed below.

4. Projection into *m*-dimensional space

The reduced-dimensionality eigenvector matrix E_1' is now used to project the original *n*-dimensional data set D into the reduced *m*-dimensional space, yielding a new, *n* x *m* dimensional data matrix D_R (the T superscript indicates matrix transposition):

$$D_R^T = E^T D^T$$

Use of PCA for dimensionality reduction raises a variety of issues: Is the original data suitable for analysis? Is there enough data for the extracted components to be reliable? Is there enough correlation among variables to make dimensionality reduction useful or even possible? Are there outliers and missing data? PCA is sensitive to both, and should be eliminated by correction and/or data transformation.

The original data needs to be mean-centred prior to generation of the covariance matrix. That is, for PCA to work properly, the mean must be subtracted from all the data dimensions where the mean subtracted is the average across each dimension. So, all *x* values have the mean for *x* subtracted from them, all the *y* values the mean for *y,* and so on. This produces a data set whose mean is zero.

PCA can be calculated from a covariance or a correlation matrix. A correlation matrix is better where dimensions vary in scale, and has the additional advantage that it is more amenable to interpretation than the covariance matrix. Otherwise, the covariance matrix can be used.

As noted above, dimensionality reduction requires elimination of eigenvectors / orthogonal basis dimensions below some specified threshold. The problem is that there is no known way of calculating an optimal threshold in any given application, and selection of one is therefore subjective. There are, however, some criteria that aid in selection (discussion in Hair *et al.* 1998, 103 ff):

- A priori criterion:

The number *n* of factors to be selected is known in advance, so that the factors with the *n* largest eigenvalues are chosen. If, for example, one wants to represent the data graphically, only the first 2 or 3 factors are usable. The obvious danger here is that too few factors will be selected, with potentially misleading results, but this is a matter of judgement in particular applications.

- Eigenvalue criterion:

  Only eigenvectors having an eigenvalue >= 1 are considered significant and retained on the grounds that significant factors should represent the variance of at least a single variable, and an eigenvalue < 1 drops below that threshold.

- Percentage of variance criterion:

  On the basis of the eigenvectors, it is possible to calculate the cumulative percentage of variance captured by successive factors. Thus, the factor with the largest eigenvector may capture 68% of the total data variance, and the second-largest 17%, giving a cumulative 85%, the third-largest 8% giving a cumulative 93%, and so on. The question, of course, is what percentage is enough. Once again, this is a matter for the researcher relative to a particular application.

- Scree test criterion:

  The scree test is so called by analogy with the erosion debris or scree that collects at the foot of a mountain. The eigenvalues are sorted in descending order of magnitude and plotted; the 'scree' is at the bottom and right of the plot, and represents the eigenvalues that can be disregarded because they are small in

The number *n* of factors to be selected is known in advance, so that the factors with the *n* largest eigenvalues are chosen. If, for example, one wants to represent the data graphically, only the first 2 or 3 factors are usable. The obvious danger here is that too few factors will be selected, with potentially misleading results, but this is a matter of judgement in particular applications.

- Eigenvalue criterion:

  Only eigenvectors having an eigenvalue >= 1 are considered significant and retained on the grounds that significant factors should represent the variance of at least a single variable, and an eigenvalue < 1 drops below that threshold.

- Percentage of variance criterion:

  On the basis of the eigenvectors, it is possible to calculate the cumulative percentage of variance captured by successive factors. Thus, the factor with the largest eigenvector may capture 68% of the total data variance, and the second-largest 17%, giving a cumulative 85%, the third-largest 8% giving a cumulative 93%, and so on. The question, of course, is what percentage is enough. Once again, this is a matter for the researcher relative to a particular application.

- Scree test criterion:

  The scree test is so called by analogy with the erosion debris or scree that collects at the foot of a mountain. The eigenvalues are sorted in descending order of magnitude and plotted; the 'scree' is at the bottom and right of the plot, and represents the eigenvalues that can be disregarded because they are small in

comparison to those that constitute the 'mountain' on the left. In Figure 27, the values to the right of about 100 on the x-axis would be scree.

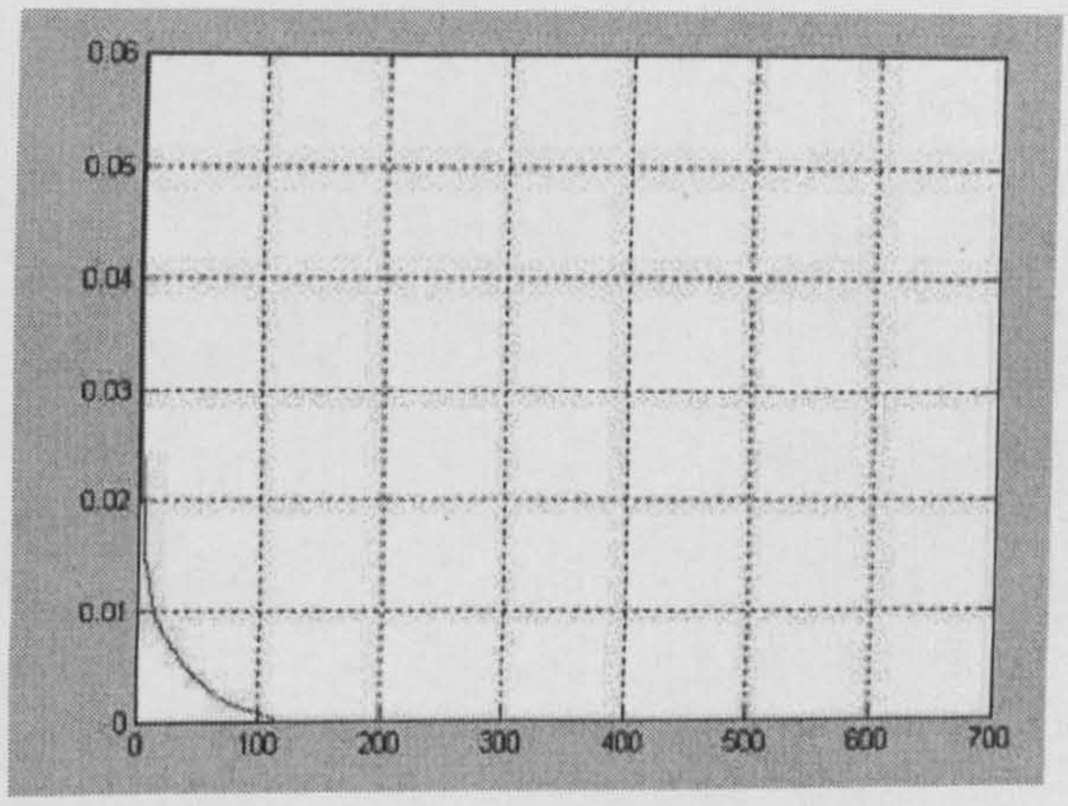


Figure 27: Scree plot

There are formal tests for the adequacy of any particular factor selection, but they are controversial. In practice, it is best to select various numbers of factors on a selection of the above criteria, and to choose the factor solution that, in the view of the researcher, gives the most useful result from an exploratory point of view.

In the original data matrix and in the covariance matrix derived from it, the variables typically have labels that are semantically significant to the researcher in a given project. Because PCA defines a new set of variables, these labels are no longer applicable to the columns of the dimensionality-reduced matrix. In some applications the main interest lies in understanding the meaning of the new set of variables relative to the original ones, and to aid in such interpretation a variety of optimizations are described in the relevant literature, cited below. For present purposes, however, the derived orthogonal basis does not need to be interpreted --PCA is used only for data reduction. These optimizations are consequently not considered here.

Note, however, that the row labels remain valid since the rows are unaffected by PCA. This is important for present purposes, since the aim is to see if the *suras* can be categorized on the basis of their lexical content, and the data matrix rows are labeled by *sura* name.

On PCA see Jolliffe (2002) and Jackson (2003). Briefer accounts are in Bishop (1995, 310ff), Everitt & Dunn (2001, 48-73), Grimm & Yarnold (1995, 99-136), Hair *et al.* (1998, 87-138), Oakes (1998, 96-108), Rietveld & van Hout (1993, 251-95), Tabachnik & Fidell (2001, 582-652), and Woods *et al.* (1986, 273-95).

5.1.6.2.4 Heuristics

The term 'heuristic' is widely taken to mean something like 'a rule of thumb'. This section ends with two such rules of thumb for dimensionality reduction.

a) Stemming

Stemming can, as above, be implemented by merging the morphologically-related columns of the frequency matrix and adding the associated frequencies. It is, however, usual in text processing-related disciplines to stem the document collection in question before creating the matrix. In cases where syntactic information is used in the stemming process, moreover, this is the only possible course.

b) The stopword list

In Information Retrieval and other text processing disciplines, it is standard procedure to compile a list of 'stop-word' types from a given document collection C, and to eliminate all the tokens of those types from C prior to data extraction, thus reducing dimensionality. The list typically comprises 'noise' words: on the one hand, the many very low-frequency words characteristics of lexical distributions in document collections, and on the other the so-

called 'function' words, that is, words like determiners and prepositions that are held to fulfil a grammatical role and carry little or no semantics relevant to text classification in the way that, say, nouns and to a lesser extent adjectives and verbs do (Belew 2000, 47 & 73ff; Ziviani 1999, 167-8). The stop-word list is in direct line of descent from the Luhn algorithm with which this section began, bypassing the keyword selection methods that have just been discussed and, with that, failing to benefit from the advantages they were developed to provide. In fact, these methods subsume the stop-word list in that:

- Low-frequency words have low variance, low IDF, high entropy, and are Poisson, and are therefore eliminated.
- Many high-frequency words have low variance, low IDF, high entropy, and are Poisson, and are therefore eliminated.

but they also provide a more nuanced approach to keyword selection. Specifically:

- Among very high-frequency words, all function words can be safely eliminated, but there are often also very high frequency non-function or 'content' words that may be useful in document classification which the keyword selection methods would retain, but that a purely frequency-based method would eliminate (Belew 2000, ch.3) .
- Where is the threshold below which word types are taken to be too infrequent to be useful? This was the main problem with Luhn's algorithm: too high a threshold, and the baby goes out with the bathwater (Salton & McGill 1983).

The top-word list approach is, in short, an undoubtedly useful but rough and ready approach to keyword selection which the methods discussed in this section supersede. In practice, it is common and justifiable to remove function words and frequency-1 words,

since these would go in any case, and then refine the selection procedure using the methods.

5.1.6.3 Data linearization

In physical systems there is a fundamental distinction between linear and nonlinear behaviour. To get an intuition for what is involved, and why the distinction is important, here is an experiment. Kick a ball and note how far it goes. Kick it again, but this time twice as hard, and once again note how far it goes. The natural expectation is that it will go twice as far, and this expectation is fulfilled. This is linear behaviour: the effect is proportional to the cause. But let's take the experiment further. Because human strength is limited, we will let a machine kick the ball in a series, each time twice as hard as the time before: k, 2k, 4k, 8k and so on. If it goes 10 metres for k, and 20 metres for 2k, will it also go 40 metres for 4k, and 80 metres for 8k? No. As it is kicked harder and harder, it goes faster and further. Air resistance becomes a factor at higher speeds, and so does rolling resistance. The ball might only go 78 metres for an 8k kick, and 150 metres for a 16k kick, etc. Eventually, the kick will be so hard that the ball bursts and goes hardly any distance at all. This is nonlinear behaviour: it is the breakdown of proportionality between cause and effect in physical systems, and it can generate a variety of complex and often unexpected -- including chaotic-- behaviours.

In nature there are few truly linear systems. Nonlinearity pervades the physical world (Hilborn 2000; Porter & Gleick 2001; Sprott 2003), and, because it does, data manifolds that describe the physical world are very likely to contain nonlinearities. The present section considers the implications of this for data analysis.

Mathematical functions are used to model physical systems. Constructing a mathematical model of some aspect P of the physical world is a three-step process:

i. Features of P relevant to a research question are identified, those features are represented as variables, and values derived from measurement of P are assigned to the variables. This is data creation as it has been described thus far.

ii. A function that generates the data created in (i) is proposed

iii. The function is tested to see if its mathematical behaviour both describes the observed behaviour of P as represented in the data, and also predicts previously unobserved behaviour of P acceptably well. If not, P is analyzed using a different set of variables, or a different function is proposed, or both, until the model is judged adequate.

If P's behaviour is nonlinear, then the function that models P will be nonlinear, and analogously in the linear case. Such models are the basis for scientific understanding of the natural world.

What are linear and nonlinear functions? Given two sets of objects A and B, standardly referred to as the 'domain' and the 'range' respectively, *a* function *f* is a rule that associates each object $a \in A$ uniquely with an object $b \in B$ (where $\in$ is read as 'belongs to') so that every object $a \in A$ is associated with only one object $b \in B$. The function *f* is said to map A to B. In standard notation such a mapping is written $b = f(a)$, that is, given any $a \in A$, *f* identifies a unique $b \in B$. The specifics of the mapping --which *a* is associated with which *b*-- are determined by the nature of the function *f*, and there is an infinite number of possible functions. For example, assume that A is the set of integers, standardly designated as N, and that B is also N; there is nothing in the definition of a function that prevents A and B being identical. A function 'double' can be defined on A and B as $b = 2a$, that is,

given some integer a ε A, the associated integer in B is twice a: 2 = double(1), 4 = double(2), and so on. Another function that can be defined on A and B is $b = a$ x a, the 'squared' function: 1 = squared(1), 4 = squared(2), and so on. Functions can become arbitrarily complex -- $b = 4a^3 + 1.6a^2 + 13a + 6$, for example. Given, then, a function $b = f(a)$, f is said to be linear if, for any numbers a and c, the following two equations hold:

1. $f(ca) = cf(a)$

2. $f(a_1+a_2) = f(a_1) + f(a_2)$

If the two equations do not hold for f, then f is nonlinear. To exemplify this definition, let us take the above two functions b = double(a) and b = squared(a) used above:

- The function b = double(a) is linear:

1. $f(ca) = cf(a)$: Taking, say, 4 as the value of a and 1.3 as the value of c, 10.4 = double(1.3 x 4) and 10.4 = 1.3 x double(4), and so on for any values of c and x.

2. $f(a_1+a_2) = f(a_1) + f(a_2)$: Selecting random values for a_1 and a_2 gives 16.96 = double(1.5 + 6.98) and 16.96 = double(1.5) + double(6.98), and so on.

- The function squared(a) is nonlinear:

1. $f(ca) \neq cf(a)$: Using the same values as above, 28.09 = squared(1.3 x 4), whereas 20.8 = 1.3 x squared(4).

2. $f(a_1+a_2) \neq f(a_1) + f(a_2)$: Using the same values as above, 71.9104 = squared(1.5 + 6.98), whereas 50.9704 = squared(1.5) + squared(6.98).

A function generates a data manifold when actual values selected from its domain are mapped to actual values in the range. For example, when plotted for some arbitrary selection of values, say a = 1..100, the above functions b = double(a) and b = squared(a) generate a linear and a nonlinear manifold respectively

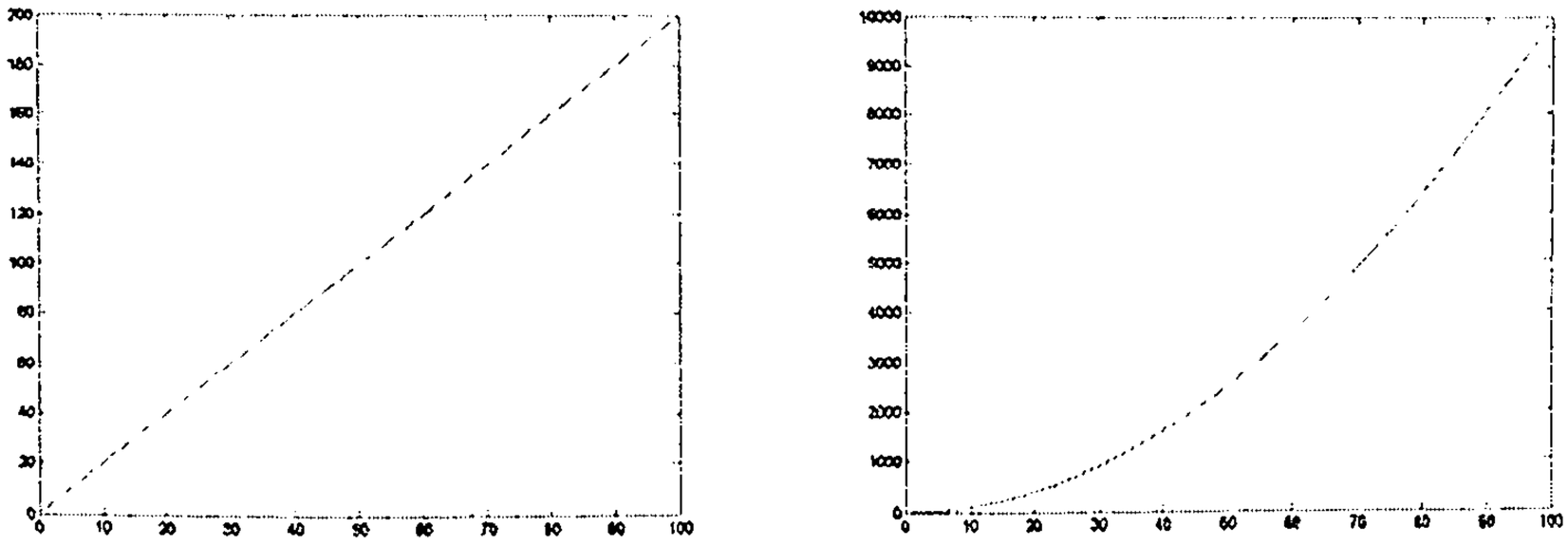


Figure 28: Linear and nonlinear 1-dimensional manifolds

With the linear function there is an invariant proportionality between a and b, and that invariance is represented by a straight line --thus 'linear'. With a nonlinear function, on the other hand, the relationship between a and b varies with different values of a, and that variance of is represented by a curved line --thus 'nonlinear'.

These data manifolds are one-dimensional, but manifolds of any dimensionality can be generated by functions whose domain and range are not scalars but n-dimensional vectors. For n = 3, for example, a linear function might generate a plane, as in Figure 29a, and a nonlinear one a curved surface, as in Figure 29b:

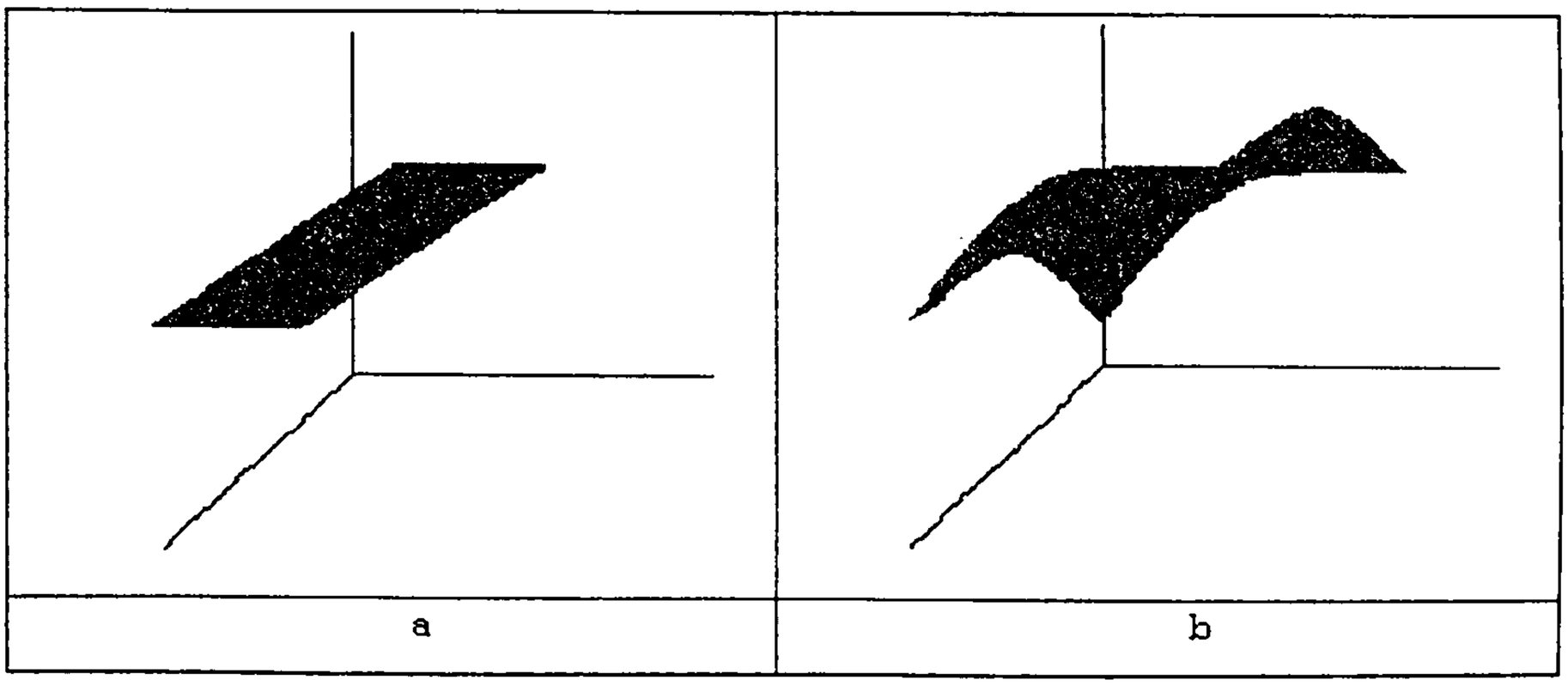


Figure 29: Linear and nonlinear 2-dimensional manifolds

In general, linear manifolds are lines and planes, and nonlinear ones curved and curved surfaces; these cannot be shown graphically for higher dimensionalities, where they are referred to as hyperplanes and hypercurves respectively. Nonlinear manifolds can range from fairly simple curves, as above, to highly complex.

The most frequently used exploratory multivariate methods are designed for analysis of linear data. When applied to nonlinear manifolds, they can give more or less seriously distorted results, depending on the nature of the nonlinearity. Consider, for example, the problem of discovering a classification for the data in Figure 30a: the data space must be partitioned such that all the points in the left-hand cluster fall into one partition, and all the points in the right-hand cluster into another. Linear methods are, by definition, limited to doing this using straight lines or surfaces; in this case, that is sufficient (Figure 30b)

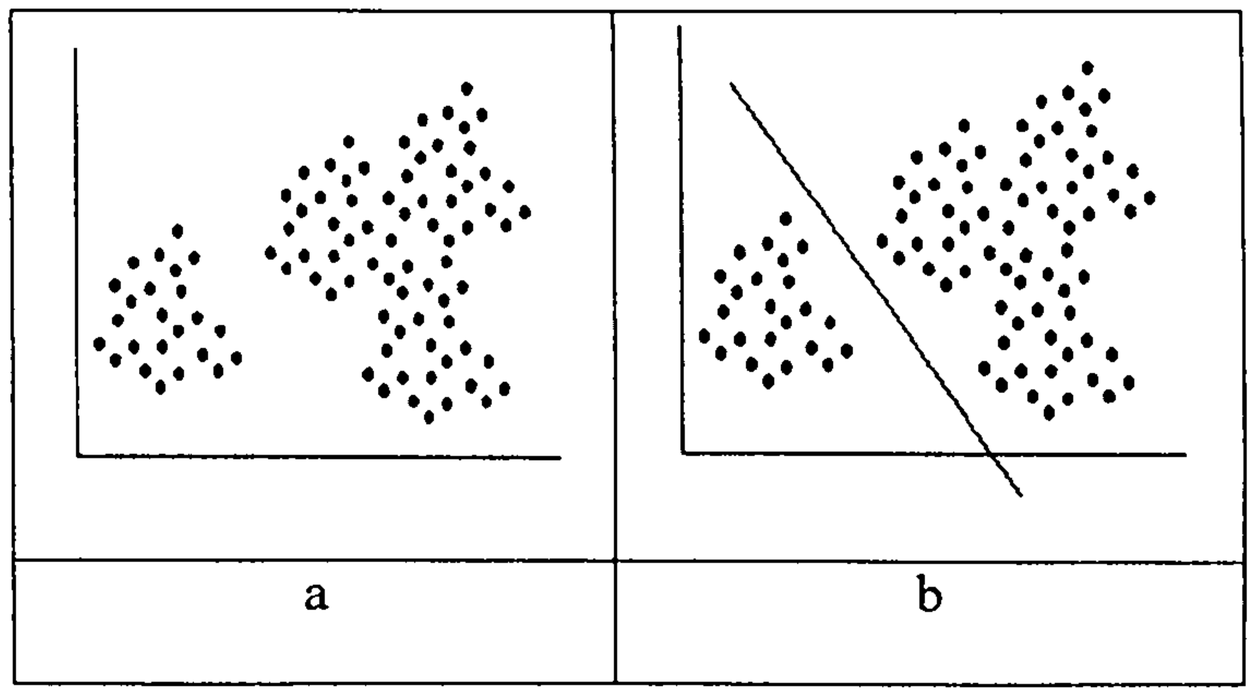


Figure 30: Linearly separable manifolds

For the data in figure 31a, however, there is no straight line that can separate the two clusters without misclassifying some of the points, as in 31b. What is required for correction classification is a method for finding a curved partition, as in 31c.

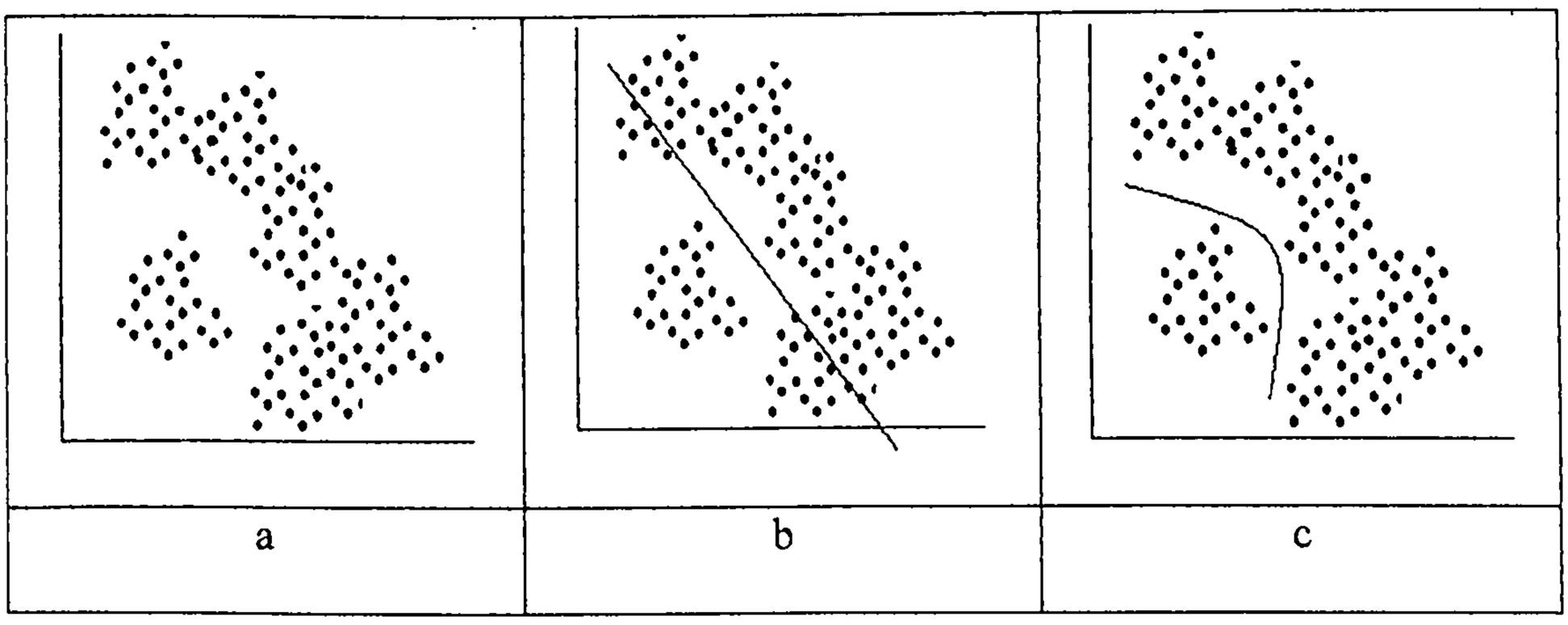


Figure 31: Nonlinearly separable manifolds

The first step in dealing with nonlinearity in a given data matrix is to determine whether or not the matrix in fact contains significant nonlinearity. This seems obvious, but, for high-dimensional data, it is not always or even usually straightforward. In real-world applications, the function that generated the data matrix may not be known. In the light of the foregoing observation that nonlinearity pervades the natural world, the strong suspicion must be that the generating function is nonlinear, but this is not certain. Even if the

generating function is nonlinear, however, there is no guarantee that every data set it generates will contain nonlinearities. This sounds paradoxical, but consider the shape of the familiar nonlinear logistic function:

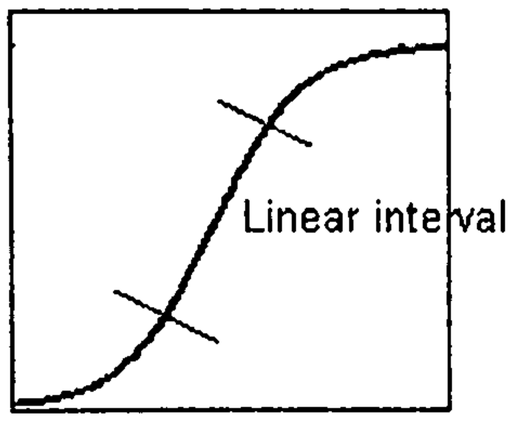


Figure 32: Graph of logistic function

Though it is nonlinear globally, there is a relatively large interval that is linear or near-linear; if the data of interest happens to come from that interval of output values, then it is linear even though it was generated by a nonlinear function. And, of course, this phenomenon generalizes --nothing says that the function underlying the data of interest must generate nonlinear data even if it is itself nonlinear. A priori reasoning cannot, in short, establish whether or not a data set contains significant nonlinearities. Only direct examination of the data will establish this.

The most common ways to assess linearity is to examine scatter-plots of the variables and to identify any nonlinear patterns in the data or to run a simple regression analysis and to examine the residuals. The residuals reflect the unexplained portion of the dependent variable: any nonlinear portion of the relationship will show up there. If significant nonlinearity is found, there are two alternatives in dealing with it. The first is to linearize the matrix using one of the standard linearization methods (Hair *et al.* 1998, 75-83). The nonlinearities may, however, themselves be of interest, and linearization throws the baby

out with the bath water. The second alternative is to use an analytical method that can accommodate nonlinearities, on which more later.

5.2 The Qur'an data

The object of study is the Qur'an, which is written in Classical Arabic[3]. To be amenable to computational analysis the text must be in digital electronic form, and the first part of this section gives an overview of the digital versions of the Qur'an currently available and motivates the selection of the one used here. Thereafter, the abstraction of the data to be analyzed is described in terms of the foregoing discussion of general principles of data creation.

5.2.1 The electronic text of the Qur'an

5.2.1.1 The source text

Electronic versions of the Qur'an are readily available from a variety of sources such as the Internet Sacred Text Archive[4], the IslamiCity Internet Community[5] and Al-Mushaf.com[6]. Some of these consist of pictures of printed editions of the text in graphical formats such as GIF, and so are unusable for textual analysis. The remainder are in text format in the conventional sense, that is, they represent phonological segments of the spoken language symbolically, and fall into two categories determined by how the textual symbols are encoded: those that encode the Arabic orthography of the Qur'an text in Unicode, and those

---

[3] There are mainly three types of written Arabic; a) Classical Arabic, the old form of the language represented mainly in the language of the Qur'an and classical literature, b) Modern Standard Arabic, a simplified form of Classical Arabic used by the Arabic – speaking world in the vast majority of written material, media and academic institutions, and c) dialectal varieties of spoken Arabic in regions such as the Egypt, Morroco, and the Levant; these dialects have no written form.

[4] http://www.islamicity.com/mosque/arabicscript/sindex.htm

[5] http://www.sacred-texts.com/isl/uq/index.htm

[6] http://www.almushaf.com/modules/source/viewSource.php

that transliterate the Arabic orthography into standard Western orthography and then represent the transliterated text in the ASCII subset of Unicode. A transliterated version was used in the present analysis, for the following reasons.

- Arabic orthography is a cursive script written horizontally from right to left. There are 29 consonant symbols and 3 long-vowel symbols; short vowels are indicated by diacritics placed above or below consonant symbols. In addition, there are diacritics to indicate gemination, the indefinite suffix, and various phonetic features. In Modern Arabic text only consonants and long vowels are used. Diacritics are omitted, rendering many orthographic forms ambiguous among several lexical types; disambiguation depends on the reader's knowledge of Arabic and the semantic context provided by the text being read. For example, depending on the context, the word ملك can be read as *mulk* ('reign'), *malik* ('king'), or *malak* ('angel').This ambiguity is a significant problem for computation of Modern Arabic text, since the disambiguating information, and semantic context more particularly, is not easily provided in current computational systems. For this reason, work on Arabic machine translation, morphological analysis, stemming, and part-of-speech tagging (Abduljaleel & Larkey 2003; Beesley 1996) has used Arabic text transliterated into Western orthography: the features denoted by the (omitted) diacritics, such as short vowels, are explicitly represented in the Western orthography, thus resolving the ambiguities.

The foregoing would appear to motivate use of a transliterated version of the Qur'an in the present case, but in fact does not. Because the Qur'an is meant to be read, it includes all the diacritics omitted in Modern Arabic text to ensure correct vocalization, and does so consistently. In other words, lexical ambiguity is not a

problem with respect to the Qur'an. The motivation for using a transliterated text is, instead, purely practical. Unicode versions of the Arabic orthography of the Qur'an represent each diacritic with a different code; because the diacritics can be combined in various ways, this results in a text representation that is often quite complex. A transliterated ASCII representation is much simpler and thus more easily dealt with by the various computational processes involved in the present study, and is thus preferred.

- Word boundaries are not consistently indicated either in Modern Arabic orthography or in that of the Qur'an. They are, however, in standard Western orthography by means of spaces and various forms of punctuation, and the process of transliteration makes the boundaries of the Arabic words explicit in this way. Since the present analysis depends on being able to identify and count words, a Western transliterated text greatly simplifies the analysis.

The Arabic language has a number of phonemes that have no equivalent in English or other European languages, and so transliteration from Arabic to Western orthography is not entirely straightforward. Several transliteration methods have been invented to represent Arabic characters in various applications (see Almisbar[7] and Ajeeb[8]), though there is no standard at present. Since there are many alphabetic symbols in Arabic which have no Western equivalent, these methods typically combine two or more Western symbols to approximate the pronunciation of the corresponding Arabic symbol, or enhance the Western symbols in some way.

Several Western-transliterated electronic versions of the Qur'an are available, such as that at http://transliteration.org/quran/home.htm. The version used in the present study is that

---

[7] Al-Misbar: http://www.almisbar.com/salam_trans.html
[8] Ajeeb: http://tarjim.ajeeb.com/ajeeb/elogin_ET.asp

produced by Muslimnet[9], which uses the orthographic mapping scheme set out in Tables 5 and 6, where the Arabic symbol is on the left and the corresponding Western one on the right in each cell:

آ a	ي ee	ج j	ن n	ص s	ظ th
ع AA	ف f	ك k	ق q	ت t	و w
ب b	غ gh	خ kh	ر r	ط t	ء /
د d	ه h	ل l	ش sh	ث th	ي y
ض d	ح h	م m	س s	ذ th	ز z

Table 5: Arabic consonants and long vowels with Western alphabetic transliterations

Fatiha	ـَ	a
Damma	ـُ	u
Kasra	ـِ	i
Shadda	ـّ	(letter doubling)

Table 6: Arabic short Vowels (diacritics)

This scheme required slight emendation for present purposes, since italicization and underlining, for which there are no ASCII codes, are used in Table 5. These were replaced as in Table 7, which gives the final consonant / long vowel mapping for the text used in this study:

[9] Muslimnet: http://www.usc.edu/dept/MSA/quran/transliteration/

آ A	ي ee	ج j	ن n	ص S	ظ Z*
ع &	ف f	ك k	ق q	ت t	و w
ب b	غ gh	خ kh	ر r	ط T	ء /
د d	ه h	ل l	ش sh	ث th	ي y
ض D	ح H	م m	س s	ذ Z	ز z

Table 7: Arabic - Western orthographic mapping used in this study

The Muslimnet text is in HTML format. This was transformed into untagged plain ACSII text, and edited so as to conform to the mapping set out in Tables 5 and 6.

Here is an example of the transliteration used in this study: the first five verses of *Sura Al-Feel*:

1. alam tarA kayfa faAAala rabbuka biaSHAbi alfeeli
2. alam yajAAal kaydahum fee tadleelin
3. waarsala AAalayhim tayran abAbeela
4. tarmeehim bihijAratin min sijjeelin
5. fajaAAalahum kaAAasfin ma/koolin

Table 8: Transliteration example

5.2.1.2 Preprocessing

Once the Muslimnet-derived text had been prepared as above, it was further processed in two ways prior to abstraction of a data matrix from it.

5.2.1.2.1 Removal of function words

As noted in the foregoing general discussion of data creation, it is standard practice in a range of natural language text processing applications to remove grammatical function words such as prepositions and determiners, and this was done here. Because of the complexity of Arabic morphology, of which more is said in due course, it was easier to compile a list of function words manually than to attempt to define an algorithm for the task. Some were taken from a concordance of the Qur'anic lexicon compiled by Abd Al-Baqi (1987), and the rest were specified by the present author on the basis of her knowledge of Arabic. There is a certain amount of subjectivity about what should count as a function word; the final list can be inspected in Appendix 2.

5.2.1.2.2 Stemming in Arabic

The complex morphology of Arabic creates a range of problems for computational text processing in general (Xu *et al.* 2002), and in particular indicates that considerable dimensionality reduction can be achieved by stemming of Arabic text. This section first gives an outline of stemming-relevant Arabic morphology, then reviews existing work on stemming of Arabic, and finally describes stemming of the Qur'an.

a) Outline of Arabic morphology

Arabic is the official language of many countries and is spoken by over 150 million people. It belongs to the Semitic family of languages, and as such differs from European languages morphologically, syntactically and semantically. In particular, it has a very complex morphology; what follows refers to Classical Arabic, the language of the Qur'an. The grammatical system of Classical Arabic, like other Semitic languages, is based on a root-

pattern scheme and is considered a root-and–pattern language. Most Arabic words are morphologically derived from roots to which many affixes (prefixes, suffixes, and infixes) can be attached to form surface words. This is generated through an application of fixed patterns which are templates to help in deriving inflectional and derivational forms of a word. Thus, the root carries semantic information while the word pattern conveys syntactic information. These roots can be triliteral [10] (three consonants) or quadriliteral (four consonants), but most word forms are triliterally based. For example, the root *KTB* has the broad lexical sense of 'writing' from which the words for 'book' *KiTAb,* 'written' *Maktuub*, 'writer' *KATiB*, and 'office' *MaKTab* (Watson 2002, 3). In addition to the different forms of the Arabic word that results from the derivational and inflectional process, most prepositions, conjunctions, pronouns, and possessives are attached to the Arabic surface form (Aljlayl & Frieder 2002).

i. *Verbs*

The root-and-pattern morphology of Classical Arabic is most commonly described with reference to verbs. Most verbs have triliteral roots, with a lesser number of quadriliteral ones and even a few pentaliteral. For example, the verbs درس *daras* 'he studied'[11] and يدرس *yadrus* 'he is studying' are both generated from the triliteral root درس *drs.* Whereas the verbs ترجم *tarjam* 'he translated' and يترجم *yutarjim* 'he is translating' are derived from the quadriliteral root ترجم *trjm*. There are three main tenses for Arabic verbs; the perfect tense to denote completed events as in كتبت *katab* 'he wrote', the imperfect tense to denote uncompleted actions as in يكتب *yaktub* 'he writes', and the imperative for commands or orders for something to be done in the future as in اكتب *uktub* 'write'. Particles can be added to these forms to create a wider range of tenses. For example, the future tense can be

[10] Also known as triconsonantal

[11] For any given example, the Arabic word will be followed by a transliteration in italics and the English translation will be placed within quotation marks next to the transliteration.

formed by adding the particle سـ *sa* or سوف *sawf* to the beginning of the imperfect verb as in سيكتب *sayaktub* 'he will write'. Verbs also have active and passive participles.

ii. *Nouns*

Nouns in Classical Arabic are subcategorised by gender, number and case. There are two genders, feminine and masculine, and the number system has singular, dual and plural forms; the plural forms can be classified into two categories; regular and irregular. The regular plurals are formed as in English by adding appropriate suffixes:

- The masculine plural is formed by adding the suffix 'ون' *oon* to the singular in the nominative case. For example: مهندس *muhandis* (engineer) - مهندسون *muhandisoon* (engineers) and 'ين' *een* to the singular in the genitive and accusative case as in مهندس *muhandis* - مهندسين *muhandiseen*.
- The feminine plural is formed by adding the suffix 'ات' *At*[12] to the singular form in all syntax cases as in كاتبة *kAtibat* (female writer) - كاتبات *kAtibAt* (female writers).

The irregular/broken plurals are formed by undergoing root changes to the singular form such as in 'كتاب' *KitAb* (book), 'كتب' *kutub* 'books'. Grammatical cases in Classical Arabic are nominative, genitive or accusative. These different cases help generate a variety of morphological variants of words.

These gender and number systems combine in an agreement system with verb and adjective forms for, say, a feminine dual noun and a feminine plural noun (see table 9).

Nouns may also be definite or indefinite by adding the definite prefix ال (the) to the beginning of the noun. For example: البنت (the girl) and بنت (girl).

[12] The A represents the prolonged vowel a.

iii. *Adjectives*

Adjectives in Classical Arabic agree with nouns in gender, number and case, and thus have commensurately many variants; see table 9 below

Arabic words	Grammatical case
تلميذ ذكي	clever student (male)
تلميذ ذكية	clever student (female)
تلميذات ذكيات	clever students (plural female)
تلاميذ أذكياء	clever students (plural masculine)
تلميذتان ذكيتان	clever students (dual feminine)
تلميذان ذكيان	clever students (dual masculine)

Table 9: Adjective agreement in Arabic

iv. *Particles*

Examples of particles are the definite article, the coordinating conjunction, many of the prepositions, and negatives. Sequences of two, three and even four particles can combine with a single word, and the number of different prefix sequences is in the hundreds. For example, a noun like والدين *wAlidayn* (parents) can be preceded by the definite article *al*, the conjunction *wa* 'and', and prepositions like *bi* 'in' or *li* 'to' as in وبالوالدين *wabialwAlidayn* (and for the parents). Verbs, too, can be preceded by conjunctions like *wa*, *fa*, and *fal*, conjunctions, the affirmative particle *la*, and prepositions like *bi*, *li*, or *ka*.

The abundance and complexity of morphological rules in Classical Arabic explains why an Arabic word can potentially have a large number of variants which are formed by different

prefixes, suffixes and infixes. Table 10 gives an impression of this variety by showing how suffixes are added to the end of a verb stem to indicate gender, number, tense and person.

Suffix	**Arabic word**	**Transliteration**	**English Meaning**
َ / a	كتبَ	Kataba	He wrote
ت / at	كتبتْ	katabat	She wrote
تُ / tu	كتبتُ	katabtu	I wrote
تما / tuma	كتبتما	KatabtumA	You wrote (masc/dual)
تا / ata	كتبتا	katabatA	They wrote (fem/ dual)
نا / na	كتبنا	katabnA	We wrote
وا /u	كتبوا	katabu	They wrote (masc/plural)
تم /tum	كتبتم	katabtum	You wrote(masc/plural)
تن /tunna	كتبتن	katabtunna	You wrote (fem/plural)

Table 10: Verb stem suffixes (variations derived from the root *ktb*)

b) Overview of Arabic stemming

There has been a fair amount of work on Arabic stemming, but no standard approach has yet emerged. Larkey *et al.* (2002) identify four general approaches: manually constructed dictionaries, algorithmic light stemmers which remove prefixes and suffixes, stemmers based on morphological analysis which attempt to find roots, and statistical stemmers which group word variants using clustering techniques. Xu *et al.* (2002) offer an alternative classification into stem-based and root-based approaches; Darwish (2002b) suggests yet another classification into the symbolic, statistical, and hybrid approaches. Al-Sughaiyer & Al-Kharashi (2004) provide a detailed review of the proposed classifications of Arabic morphological analysis techniques with reference to stemming, themselves suggesting a

four-way classification: the table lookup approach where a given word is checked against all valid natural Arabic words along with their morphological variants stored in a huge table, the linguistic approach, which utilizes linguistic rules that have been derived through analysis of the Arabic morphological system, the combinatorial approach, which tests and compares all combinations of letters of a given word against a list of roots, and the pattern-based approach, which uses the apparent symmetry of natural Arabic words.

These various approaches generate more or less different results, and, while researchers have provided general descriptions of the methods they propose, they have not in general given measures of accuracy (Al-Sughaiyer & Al-Kharashi 2004), though Al-Kharashi & Al-Sughaiyer (2002) have recently proposed a framework for testing and evaluating Arabic morphological analysis and stemming techniques. As such, it is difficult to make a principled choice for the present application. Light and root-based stemmers are most commonly used, and, however weak the criterion, were for that reason the ones considered for use here.

i. *Root-based stemming*

Given an orthographic word, root-based stemmers use morphological analysis to identify its root, and then strip off anything that is not part of the root; several morphological analyzers have been developed. Al-Fedagi & Al-Anzi (1989) developed a morphological analyser which compares words against a list of patterns to extract the root. The list they constructed is restricted to triliteral roots only. Al-Shalabi (1996) proposed a slightly improved version of Al- Fedaghi & Al-Anzi's morphological analyser which deals with words with quadriliteral roots. Beesley (1996) describes a finite-state system that performs morphological analysis of online vocalized or non-vocalized Arabic texts. Their system provides the root, pattern, attached affixes, part of speech tags, person, number, mood,

voice and aspect of any given word. Hegazi & Elsharkawi (1985, 1986) developed morphological algorithm for Arabic words. The system gives the root of the word, its morphological pattern, and morphological attributes. Darwish (2002b) presented a quick method of developing a shallow Arabic morphological analyzer that generates the possible roots of any given word. It performs automatically and thus does not require a manually constructed list of rules and affixes. Khoja (1999) developed a root-based stemmer to extract the root of a given Arabic surface word. Khoja's algorithm starts to remove the suffixes, prefixes, and infixes of a given Arabic surface word. After every elimination process, the root extraction is performed. Buckwalter (2002) designed an Arabic morphological analyser which performs the following functions: tokenization, word segmentation, dictionary lookup, compatibility check, analysis report and second lookup.

ii. *Light Stemming*

Light stemmers do not attempt to identify roots, but identify and remove known prefixes and suffixes from given orthographic word form. All Arabic light stemmers adhere to the same steps of normalization and stemming. Normalization consists of the following steps: removal of punctuation and diacritics, replacement of أ ! آ with plain ا, replacement of ىء with ئ , replacement of final ى with ي, and replacement of final ة with ه .The main difference among the various light stemmers is the number of prefixes and suffixes removed. Each has different stopword lists consisting of Arabic pronouns, particles and the like, which are removed after minimal normalization. Examples of light stemmers are Aljlayl & Frieder (2002), Chen & Gey (2002), Darwish (2002a), Larkey (2001), Larkey *et al.* (2002).

iii. *Problems with root-based and light stemmers*

It may appear that root-based and light stemming are just two different ways of doing the same thing, but this is not the case. To see why, it is necessary to understand that, in Arabic, the stem of a word is different from its root. The stem of a word is what remains after prefixes and suffixes have been removed, but it may contain one or more infixes, whereas as the root is the form of the word with all affixes removed, including infixes. For example, the word *katabat,* 'she wrote', consists of the stem *katab* and the subject marker suffix *–at*, but the root of that word is *ktb.* Root-based stemmers generate roots, and light stemmers generate stems.

Root 'stemmers' and light stemmers have their characteristic problems. The root-based approach is the more aggressive in that it tends to conflate morphologically-distinct words into stem classes, a tendency that is exacerbated in non-Classical texts in which, as noted earlier, the orthography does not include vowel diacritics and therefore provides insufficient morphologically-useful information for root identification. Words derived from the same root do not necessarily have closely related semantics, and assigning morphologically variant but semantically disparate words to the same equivalence class would distort any data abstracted from the text in question. The example used earlier was 'neutron' and 'neutralize'; an Arabic example would be the words جامع (mosque), جامعة (university), مجموعة (group), and اجتماع (meeting), all of which belong to the same root جمع.

Al-Suwaynea (1994) argues that using root-based algorithms might be useful for dictionaries and other natural language applications but not for information retrieval tasks. Using root-based algorithms can cause conflation of words of distinct classes, which degrades the performance by introducing noise. To some extent, reducing morphological variants to a single root might be useful in some situations and for specific tasks when high

recall is required but without high precision. However, the stem-based methods are more effective when high precision is needed.

Light stemming may on the other hand be too unaggressive in that it may fail to classify certain kinds of orthographic form correctly. For example, the so-called broken plurals for nouns and adjectives are not classified with their singular forms and past tense verbs are not classified with their present tense forms, because, in both cases, different infixes yield different stems. Light stemmers also fail for words having start or end character strings that are similar to affixes. For example, the letter 'و' in Arabic could be a prefix meaning 'and' as in 'والكتاب' (and the book) or part of the word itself as in 'والد' (father). Removing the prefix 'و' throughout the text would probably generate at least some errors.

Handling broken plurals still remains a problem for both types of stemming. Existing stemming algorithms fail to reduce these plurals to their singular and therefore, broken plurals are not handled by current Arabic stemmers. Goweder (2004) and Goweder *et al.* (2004) developed a light stemmer with broken plural identification. Their results improved the performance of information retrieval and outperformed standard light stemming and root stemming. For other studies on broken plurals in Arabic see McCarthy & Prince (1990) and Kiraz (1996).

The conceptual and method-specific problems notwithstanding, stemming remains a fundamental tool in computational processing of Arabic text. Studies by Abu-Salem *et al* (1999), Aljlayl & Frieder (2002), Al-Kharashi & Evans (1994), Al-Sughaiyer & Al-Kharashi (2004), Larkey & Connell (2001), Larkey *et al.* (2002), and Xu *et al.* (2002) have all shown that stemming in highly inflected languages like Arabic can significantly improve the information retrieval performance. Most of these studies have, moreover, argued that light stemmers tend perform better than root-based ones (Larkey & Connell

(2001); Larkey *et al.* (2002), and Al-Jlayl & Frieder (2002), though Al-Kharashi & Evans (1994) and Abu-Salem *et al.* (1999) argue the opposite.

c) Stemming the Qur'an

The proposed analysis of the Qur'an involves counting the occurrences of lexical items in the Qur'an. Such a task cannot be done accurately without some sort of stemming of words in the text. The main problem with using any of the stemmers described in the above review is that they are designed for use with Modern Standard Arabic. The language of the Qur'an, however, is Classical Arabic, which differs from the Modern standard in morphology, lexicon and orthography. Orthographic variations and the widespread use of diacritics and glyphs in the representation of the language of Classical Arabic increase the difficulty of stemming. In many respects, the Qur'an, with its unique lexicon and orthography, requires a text-specific approach (Thabet 2004).

It would probably have been possible to adapt an existing stemmer to accommodate the Qur'an --assuming of course that the source code for the implementation could be obtained-- but it seemed more straightforward to design a Qur'an-specific stemmer using ideas from the existing literature, and then to implement it. This was done; the result, program *StemQuran*, is described in Appendix 1.

StemQuran is a light stemmer that takes the transliterated text of the Qur'an described above as input. The algorithm for the stemmer consists of two main steps:

Step 1: Prefix stemming

Nine prefixes are deleted: *wa, fa, la, li, lil, bi, al, ka, sa*. Starting with the first word in the Qur'an and reading each word in the text to the last, when a word w beginning with one of the above prefix letter strings str_w is found:

- Associated with each of the first seven prefixes *wa, fa, la, li, lil, bi, al* is a 'stopword' list containing words for which the str_w is not in fact a prefix but part of the stem. If *w* is in the stopword list, then str_w is not a prefix and no action is taken, that is, *w* remains unaltered in the Qur'an text. If, however, it is not in the stopword list, then str_w is a prefix: it is deleted from *w* and what remains of *w* after deletion is inserted in place of the original *w* in the text. The stopword list for each of these seven prefixes was compiled manually, and they are given below.

wa&ada	wa&adaha	wa&adahA	wa&adakum	wa&adakumu
wa&adanA	wa&adnA	wa&adnAhu	wa&adnAhum	wa&adoohu
wa&adtahum	wa&adtanA	wa&da	wa&dahu	wa&daka
wa&dan	wa&dihi	wa&du	wa&duhu	wa&dun
wa&eedi	wa&iyatun	wabAla	wabeelAn	waDa&ahA
waDa&at	waDa&at-hA	waDa&tuhA	wadda	wadda&aka
waddan	waddat	waddoo	wadoodun	wafdAn
waffA	wahaba	wahabat	wahabnA	wahana
wahanoo	waHdahu	waHeedAn	wahhAjAn	Wahnan
wahnin	waHyan	waHyuhu	waHyun	Wajabat
wajada	wajadahA	wajadnA	wajadnAhu	Wajadoo
wajadtu	wajadtuhA	wajadtum	wajadtumoohum	wajeehan
wajeehAn	wajha	wajhahA	wajhahu	Wajhaka
wajhi	wajhihA	wajhihi	wajhika	Wajhiya
wajhu	wajhuhu	wajilat	wajilatun	wajiloona
wajjahtu	wakeelAn	wakeelun	wakkalnA	walada
waladan	waladAn	waladihi	waladin	waladnahum
waladun	walAyatihim	waleedan	waleejatan	wAlidayya
waliyyan	waliyyAn	waliyyee	waliyyin	waliyyiya
waliyyu	waliyyuhu	waliyyuhum	waliyyuhumA	waliyyuhumu
waliyyukumu	waliyyun	waliyyunA	wallA	wallAhum
wallaw	wallaytum	waqa&a	waqa&ati	waqaba
waqArAn	waqoodu	waqooduhA	waqran	waqrun
warA-ee	warA-i	warA-ihi	warA-ihim	warA-ikum
warAa	warAahu	warAahum	warAakum	warada
waradoohA	waraqatin	waraqi	waratlıati	wardatan
warithoo	wasaqa	wasaTan	waSeelatin	waSfahum
wasi&a	wasi&at	wasi&atun	wasi&ta	waSiyyatan
waSiyyatin	waSSA	waSSAkum	waSSAkumu	waSSalnA
waSSaynA	waT-an	waTaran	wathAqahu	waykaanna
waykaannahu	waylaka	waylakum	waylanA	waylatA
waylatanA	waylun	wazanoohum	wazara	wazeeran
waznAn				

Table 11: stopword list for *wa*

fa&&Alun	fa&ala	fa&alahu	fa&alna	fa&alnA
fa&aloo	fa&aloohu	fa&alta	fa&altuha	fa&altuhu
fa&altum	fa&lataka	fadaynAhu	faDDala	faDDalakum
faDDalanA	faDDalnA	faDDalnAhum	faDDaltukum	faDlahu
faDlan	faDli	faDlihi	faDlin	faDlu
faDlun	fajjarnA	fajjin	fajwatin	fakhoorAn
fakhoorin	fakhoorun	fakiheena	fakkara	Fakku
falakin	faqeeran	faqeerun	far&uhA	faraDa
faraDnA	faraDnAhA	faraDtum	faraghta	faraqnA
faraqnAhu	farartum	farashnAhA	fardAn	Fardan
fareeDatan	fareeqan	fareeqAn	fareeqAni	Fareeqin
fareequn	fariHa	fariHeena	fariHoo	fariHoona
fariyyAn	farjahA	farqAn	farraqoo	farraqta
farrat	farraTnA	farraTtu	farraTtum	farshan
farthin	fasAdan	fasAdin	fasAdun	faSala
faSalati	fasaqoo	faSeelatihi	fashiltum	faSla
faSlun	faSSala	faSSalnA	faSSalnAhu	fataHa
fatAhA	fataHnA	fataHoo	fatan	fatannA
fatannAhu	fatannAka	fatanoo	fatantum	faTara
faTarahunna	faTarakum	faTaranA	faTaranee	fatayAni
fatayAtikum	fatayAtikumu	fateelAn	fatHan	faZ*Z*an
fatHun	fatratin	fawAkiha	fawAkihu	fawAqin
fawjan	fawjun	fawqa	fawqahA	fawqahum
fawqahumu	fawqakum	fawqakumu	fawqi	fawqihA
fawqihi	fawqihim	fawqihinna	fawqikum	Fawrihim
fawta	fawzan	faza&in	fazi&oo	

Table 12: stopword list for *fa*

la&alla	la&allahu	la&allahum	la&allaka	la&allakum
la&allanA	la&allee	la&ana	la&anahu	la&anahumu
la&anat	la&annA	la&annAhum	la&iban	la&ibeena
la&ibun	la&nan	la&nata	la&natakum	la&natan
la&natee	la&natu	labanan	labanin	Labitha
labithnA	labithoo	labithta	labithtu	labithtum
laboosin	labsin	ladA	lada	Ladayhi
ladayhim	ladaynA	ladayya	ladun	Ladunhu
ladunka	ladunnA	ladunnee	lafeefAn	Laghwan
laghwun	lahabin	laHma	laHman	laHmi
laHmin	laHmu	laHni	lahwa	Lahwan
lahwun	lajjoo	lamasnA	lamasoohu	lammAn
laqeenA	laqeetum	laqeetumu	laqiyA	Laqoo
laqookum	laqqAhum	lasta	lastu	Lastum
lastunna	lAta	laTeefan	laTeefun	laZ*A
laZZatin	lawAqiHa	lawHin	lawmata	lawnuhA
lawwAHatun	lawwaw	layAlin	layAliya	laylahA
laylan	laylata	laylatan	laylati	Laylatin
laylatu	laysa	laysati	laysoo	Layta
laytahA	laytanA	laytanee	layyan	Layyinan

lAzibin	libadAn	libAsa	libAsahumA	libAsan
libAsAn	libAsu	libAsuhum	libAsun	Linta
liqA-i	liqA-ihi	liqAa	liqAanA	lisAna
lisAnaka	lisAnan	lisAnee	lisAni	lisAnika
lisAnin	lisAnu	lisAnun	liwAZan	lizAman
lizAmAn				

Table 13: stopword list for *la / li / lil*

biDA&atahum	biDA&atan	biDA&atunA	bid&an	bidAran
bi/rin	bi/sa	bi/samA	bismi	biD&a
biD&i	bikrun	binAan	bisATAn	biTAnatan
biya&un				

Table 14: Stopword list for *bi*

alA-i	alAfin	alam	alatnAhum	Aleemin
aleemun	alfa	alfAn	alfaynA	Alfayni
alfi	alfin	alfun	alif-lAm-meem	alif-lAm-meem-rA
alif-lAm-meem-sAd	alif-lAm-rA	alihatakum	alihatan	Alihatee
alihatihim	alihatikum	alihatinA	alihatuhumu	Alihatun
allA-ee	allafa	allafta	allAha	allAhi
allAhu	allAhumma	allAtee	allaZayni	allaZAni
alihatika	alihatunA	alwAhin	allatee	allaZee
allaZeena	aleemAn	aleeman	alsinatihim	alsinatahum
alsinatuhum	alsinatuhumu	al&anhum	alfAfAn	Alfaw
alfayA	alqA	alqAhu	alqAhA	alsinatikum
alsinatukumu	alsinatin	aladdu	alHaqtum	alHaqnA
alHiqnee	alzamnAhu	alzamahum	alqat	Alqaw
alqoo	alqaytu	alqaynA	alqA	Alqi
alqih	alqihA	alqawoo	alqoohu	Alqiya
alqiyA	alqiyAhu	alqeehi	alhamahA	alhAkumu
alwAHin	alwAnikum	alwAnuhu	alwAnuhA	alannA

Table 15: Stopword list for *al*

- For the remaining two prefixes *ka, sa* the above procedure is reversed, in that the associated stopword lists contain the words that are stemmed rather than, as above, those that are not. The lists are searched and, if w is found in them, str_w is removed and that remains of w is inserted into the text; if w is not found, then no action is taken. The reason for this special case is that words whose stems begin with *ka* and

sa are much more frequent than words in which they are prefixes, and so it was easier to compile lists of words to stem rather than lists of words not to stem.ka:

ka&arDi	ka&aSfin	ka&aZAbi	kaaHadin	kaal-a&lAmi
kaal-a&mA	kaal-an&Ami	kaal&ihni	kaal&urjooni	kaalddihAni
kaalfakhkhAri	kaalfarAshi	kaalfi	kaalfujjAri	kaalHijArati
kaaljawAbi	kaaljibAli	kaallatee	kaallathee	kaallaZee
kaallatheena	kaallaZeena	kaalmu&allaqati	kaalmufsideena	kaalmuhli
kaalmujrimeena	kaalonthA	kaalqaSri	kaalrrameemi	kaalSSareemi
kaalZ*Z*ulali	kaalTTawdi	kaamthAli	kabASiTi	kada/bi
kadu&A-i	kaghalyi	kahasheemi	kahay-ati	kaHubbi
kajahri	kakhalqihi	kakhashyati	kakheefatikum	kalamHi
kalamHin	kamA-in	kaman	kamathali	kamishkAtin
kamithlihi	kanafasin	kanafsin	karamAdin	kaSAHibi
kasarAbin	kaSayyibin	kashajaratin	kaTayyi	kaZ*ulumAtin
kazar&in				

Table 16: Stopword list for *ka*

saaktubuhA	saaSrifu	saastaghfiru	saateekum	saAteekum
saatloo	saawee	saAwee	sanad&u	sanafrughu
sanajzee	sanaktubu	sananZ*uru	sanashuddu	sanasimuhu
sanastadrijuhum	sanazeedu	sanu/teehim	sanu&aZZibuhum	sanu&eeduhA
sanudkhiluhum	sanulqee	sanumatti&uhum	sanuqAtilu	sanuqattilu
sanuqri-oka	sanurAwidu	sanureehim	sanuTee&ukum	sanuyassiruhu
saolqee	saonabbi-oka	saonzilu	saoreekum	saorhiquhu
saoSleehi	sata&lamoona	satajidoona	satajidunee	sataZkuroona
sataZkuroonahunna	satubSiru	satud&awna	satughlaboona	satuktabu
saturDi&u	saturaddoona	saya&lamoona	saya&lamu	sayadkhuloona
sayaghliboona	sayahdeehim	sayahdeeni	sayaHlifoona	sayaj&alu
sayajzeehim	sayakfeekahumu	sayakfuroona	sayakoonu	sayanAluhum
sayaqooloona	sayaqoolu	sayarA	sayarHamuhumu	sayaHshuruhum
sayaSlA	sayaSlawna	sayaZZakkaru	saya/teehim	sayu/teehi
sayu/teena	sayu/teenA	sayubTiluhu	sayudkhiluhumu	sayudkhiluhum
sayughfaru	sayuhzamu	sayuHbiTu	sayujannabuhA	sayujzawna
sayunfiqoonahA	sayunghiDoona	sayureekum	sayuSeebu	sayuSeebuhum
sayuTawwaqoona				

Table 17: Stopword list for *sa*

Step 2: suffix stemming

This step involves the removal of suffixes, which include pronouns, articles, prepositions, etc. The following suffixes, ranging from length-1 to length-10, were removed:

Length-1	Length-2	Length-3	Length-4	Length-5	Length-6	Length-7	Length-8	Length-9	Length-10
a	An	ani	ayni	hinna	tukumA	atahumA	tumoohum		tumoohunna
i	an	him	eehi	Hunna					
o	ee	Hum	Eena	nAhum					
u	hA	iya	himA	nahum					
	ha	kum	hima	tahum					
	hi	nee	himu						
	hu	tum	humA						
	in	uki	huma						
	ka	uma	humu						
	nA	wka	Kuma						
	na	woo	kumA						
	on		kumu						
	oo		nAhA						
	um		naha						
	un		nAhu						
			nahu						
			nAka						
			oohA						
			ooha						
			oohu						
			oona						
			ooni						
			oom						
			tana						
			tuhu						
			tumA						
			tumu						
			wnee						

Table 18: Suffixes removed by stemming

The procedure is iterative, removing suffixes in decreasing order of suffix length. For each of the suffix lengths from 10 down to 3 (except, obviously, for length-9), the text of the Qur'an is read from beginning to end, one word at a time. Each time a word w is found containing a suffix str_w in the relevant column of the above table, str_w is deleted from w, and what remains of w replaces the original w in the text. This worked well for suffixes of length-3 to length-10, but length-1 and length-2 suffixes are pervasively ambiguous in the sense that they could be part of word stems rather than suffixes. There was no obvious algorithmic way of resolving the ambiguity, and so the stemming for these two suffixes was carried out manually.

5.2.2 Qur'an variable selection

Given the stated research aim, the variables selected to describe the Qur'an are the lexical types it contains, where 'lexical type' is understood as an abstraction over a set of identical lexical tokens, 'lexical token' as a string of alphanumeric symbols, and 'abstraction' as a set label, as noted earlier. The total number of lexical types in the text of the Qur'an described above, that is, the transliterated source text from which function words had been removed and the remainder of the text stemmed, was 6696.

5.2.3 Qur'an data representation

The section on data creation in the abstract described vector space representation. This is adopted here. Specifically, there are 114 'documents' in the collection, that is, 114 *suras*. Each *sura* s_i, for $i = 1..114$, is represented by a lexical profile vector p_i of length n, where n is the number of lexical types in the Qur'an; as just noted, $n = 6696$. Each element p_{ij}, for $j = 1..n$, is associated with a unique lexical type, and contains the frequency of lexical type j in *sura* i. The 114 profile vectors are collected in a matrix Q0, in which each row is a lexical profile vector, each column is associated with a unique lexical type, and the numerical quantity in any given element Q_{ij} is the number of times lexical type j occurs in *sura* i.

5.2.4 Qur'an variable value assignment

The frequency values for Q0 are obtained by counting, for each *sura*, the number of times each of the lexical variables occurs. This is done using the program *CreateFreqMatrix*, which is described in Appendix 1.

5.2.5 Qur'an data validation

Validation of Q0's accuracy is not an issue, since calculation of lexical type frequencies in the Qur'an can be and was determined without error.

5.2.6 Qur'an data transformation

Q0 was transformed in three main steps: (i) application of heuristics, (ii) normalization for variation in *sura* length, and (iii) sparsity minimization.

5.2.6.1 Heuristics

All the columns of Q0 with a cumulative frequency of 1 across all *suras* were removed. This reduced the column dimensionality of Q0 from 6696 to 3658.

5.2.6.2 Normalization for variation in *sura* length

There is no obvious reason to choose cosine normalization over normalization by average document length or vice versa, since both are theoretically well motivated. It would have been possible to apply both to Q0 and then to compare their effects on the analytical results that follow, but, if those results turned out to be different, it would not be obvious which to prefer. Normalization by average document length was therefore selected at random and applied to Q0.

5.2.6.3 Sparsity minimization

At 3658 matrix columns after elimination of the frequency-1 lexical type variables, the dimensionality of Q0 is very high relative to the available 114 data points, and the data manifold correspondingly poorly defined. Dimensionality reduction is therefore necessary. This was done in two stages: keyword selection and variable redefinition.

5.2.6.3.1 Keyword selection

Earlier, the decision was taken to use the various available methods for identification of significant keywords in document classification not for weighting, as is usual in Information Retrieval, but only for selection, and a justification was promised. This section begins by presenting that justification.

At the root of the decision is the considerable variation in *sura* length which was pointed out as the beginning of this discussion. To understand the significance of this, a brief background account of document length variation in Information Retrieval is required.

As IR has increasingly come to work with very large full-text document collections, variation in document length has emerged as a major issue. It is a commonplace in the IR literature that longer documents stand a better chance of retrieval than shorter ones (Salton & Buckley 1988) for the simple reason that, with respect to a document collection D and a set K of *n* keywords, a given keyword $k \varepsilon$ K with a known probability of occurrence across all the texts in D has a greater chance of appearing in a longer document than a shorter one. Assuming a vector space model of D, this implies that

- the *n*-dimensional vector for a longer document will in general be of greater magnitude in the space than the vector for a short one in that the keyword frequencies in the longer document vector will be higher.
- the vector for a shorter document will in general be sparser than the vector for a longer one in that there will be more zero-entries in the shorter-document vector.

IR aims to retrieve documents relevant to a given query irrespective of its length, and as such this preference for longer documents is undesirable. The problem has been addressed by eliminating document length as a factor in retrieval as much as possible using a variety

of methods, two of which were described earlier. In the present context it is not the details of the methods that are of interest, but the assumption that underlies them: that it is possible to estimate what a document would have been like in terms of keyword frequency if it had been longer or shorter. This assumption is stated in the IR literature in terms of 'scope' and 'verbosity', where scope is the conceptual content of a document *d* ε D, and verbosity is the number of words that *d* uses to articulate its scope: longer documents are longer because they are more verbose in articulating their scope, shorter documents are shorter because they are more economical, and it is possible to compensate for verbosity and economy by adjusting their respective frequency vectors. As a result, 'once we have decided that *about-ness is conserved* across documents, all document vectors will have constant length' (Belew 2000, 89).

It is not difficult to find fault with this. To assume, for example, that the frequency of some specific keyword automatically increases as a function of document length is to presume to know the author's intentions and is analogous to extrapolation in statistics, whose dangers are well known. However, nothing succeeds like success. Document length normalization gives improved results in IR, and is therefore used whatever the objections in principle. The same applies in the present instance. The unnormalized 114 *sura* vectors were hierarchically cluster analyzed with the result that they clustered primarily on the basis of relative vector length, which in terms of the present discussion is without interest; when they were first length-normalized the effect of relative length was substantially diminished and the results were conceptually interesting, as we shall see. There is, in short, no alternative to length-normalizing the *sura* vectors.

What has all this to do with the decision to use keyword selection rather than weighting? As noted, many of the *suras* are very short and the associated frequency vectors are

correspondingly sparse, consisting mainly of zero-frequency variables. Given their shortness relative to the longer *suras*, document length normalization substantially increases the actually-attested frequencies of the non-zero variables on the assumption that the length-adjusted values approximate what the frequencies would have been had the *suras* been longer, but leaves the zero-variables unaltered. The conceptual content of these *suras* is therefore represented by relatively very few variables containing conjectural values. If these values are now further adjusted by keyword weighting, they become even more conjectural. There is no alternative but to use document length normalization. Keyword weighting, in the other hand, is only an optimization. To minimize conjecture, therefore, keyword weighting is not used.

The aim of dimensionality reduction by keyword selection is to select m variables / columns from Q0 such that $m <$ the column dimensionality of Q0. Four keyword selection techniques have been described, one based on variance, one on term frequency/inverse inverse document frequency, one on fitting to a standard Poisson distribution, and one on entropy; for brevity, these are henceforth referred to as 'variance', 'TF/IDF', 'Poisson', and 'entropy' methods respectively. Each of these methods was applied to each of the columns of Q0:

- The variance for each each column was calculated and saved as vector $v_{variance}$.
- The TF/IDF for each column was calculated and saved as vector $v_{tf/idf}$.
- The fit of each column to the theoretical Poisson distribution was calculated, and the values representing each fit were saved in vector $v_{Poisson}$.
- The signal for each column was calculated and saved as vector $v_{entropy}$.

These four vectors were then sorted in descending order of magnitude and plotted:

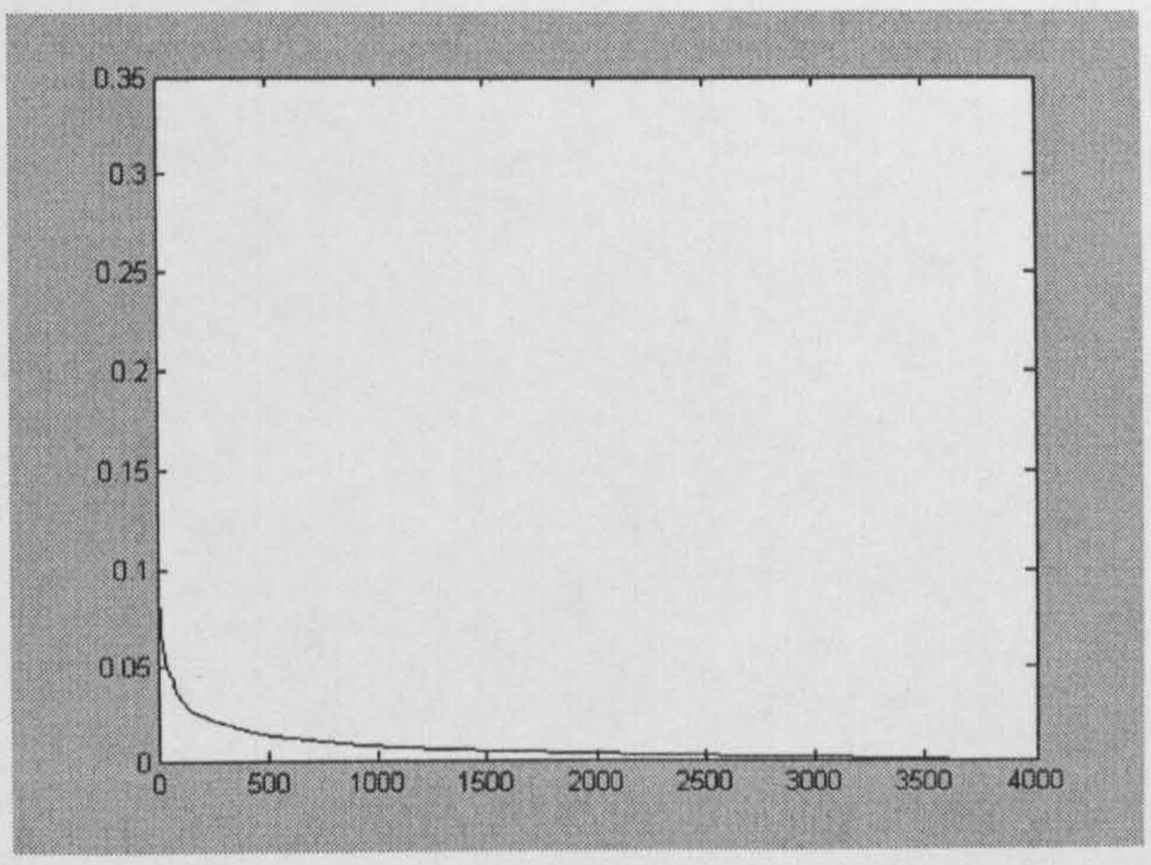


Figure 33: $v_{variance}$

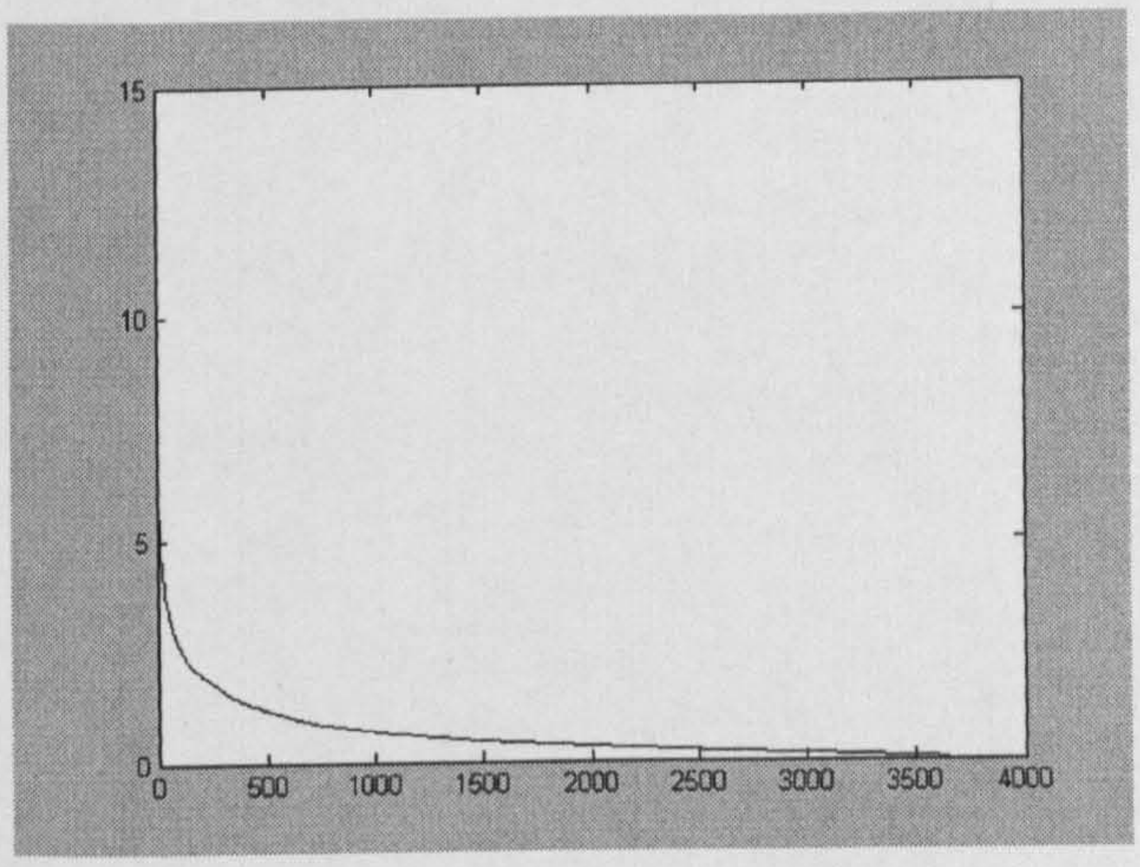


Figure 34: $v_{tf/idf}$

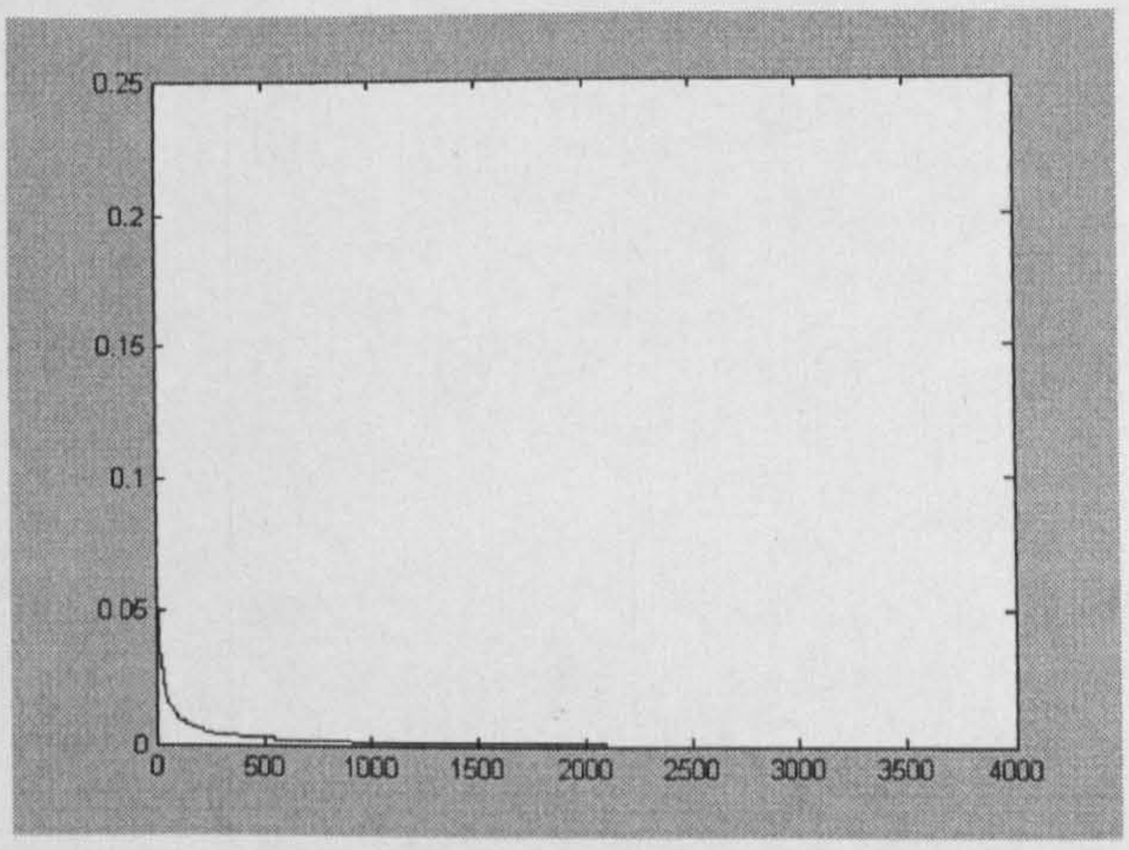


Figure 35: $v_{Poisson}$

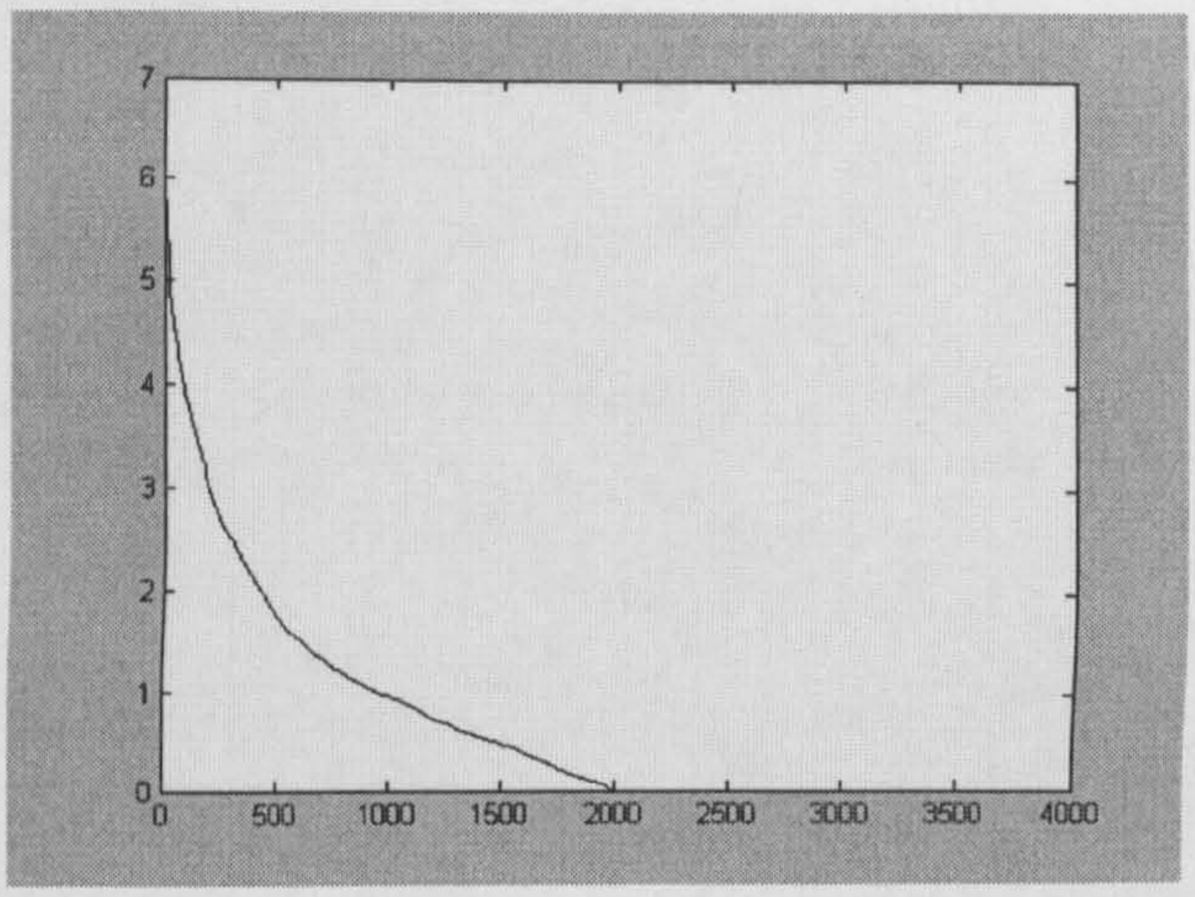


Figure 36: v_{signal}

The plots show that, in all four cases, the most significant variables --and thus the ones definitely to be retained-- are the first 500 or so. To the right of that the methods differ on the significance of the remaining variables, though there is a downward trend in all of them. Where to draw the line is a subjective matter, but to judge from the shape of the plots 1000 seems a reasonable threshold.

Though they pretty much agree on the distribution of variable significance, however, the four methods differ to some extent on which lexical items they include in the set of 1000 most significant: there is a core of lexical types on which all four methods agree, but surrounding this core is a penumbra of types on which they do not. Specifically:

Selected by *n* methods	**Nr of lexical types**
n = 4	761
n = 3	140
n = 2	201
n = 1	134
	Total = 1236

Table 19: Lexical types selected by keyword selection methods

Thus 761 lexical types are selected by all four methods for inclusion in the set of the 1000 most significant variables, 140 are selected by three of the methods, and so on.

Given this failure of unanimity on which lexical items are most significant, the question of which method's selection to use in generating a dimensionality-reduced matrix for subsequent analysis arises. There is no obvious basis for a choice among them, however, and as such a different approach is adopted. Instead of choosing the keyword selection generated by one of the four methods at random, two keyword sets are defined:

- The keyword set consisting of the intersection of the sets generated by each of the four methods, that is, of the 761 types in the above table that all the methods agree on, yielding a 114 x 761 matrix Q1.
- The keyword set consisting of the union of the sets generated by each of the four methods, that is, of the 1236 types in the above table, yielding a 114 x 1236 matrix Q2.

Q1 and Q2 are both analyzed and the results of analysis compared. Transformation of the original data matrix Q0 into Q1 and Q2, is implemented in program *EditMatrix*, for which see Appendix 1.

5.2.6.3.2 Variable redefinition

Dimensionality reduction by variable redefinition in general, and PCA in particular, make sense only if there is a significant level of correlation among variables in the data matrix to be reduced. The standard way of doing this is to apply Bartlett's sphericity test, which tests the null hypothesis that the correlation matrix abstracted from the data matrix is an identity matrix. The statistical package SPSS provides the test as part of its data reduction procedure; the results for Q1 and Q2 are:

Bartlett's test of sphericity	Approx. Chi-Square	963.855	Bartlett's test of sphericity	Approx. Chi-Square	934.650
	df	435		df	45
	Sig.	.000		Sig.	.000
Q1			Q2		

Table 20: Bartlett's sphericity test for Q1 and Q2

The probability that both the Q1 and Q2 correlation matrices are identity matrices is 0 to three decimal places. There is, therefore, sufficient correlation between variables in both Q1 and Q2 to make dimensionality reduction by variable redefinition worthwhile. This is readily confirmed by direct observation. Correlations between the variables of Q1 and Q2 were calculated and, in both cases, there are 220 correlations that range from 0.3 --the textbook threshold for significance (Hair *et al.* 1998, 99)-- and 0.87; the distribution of correlations can be seen by sorting them in descending order of magnitude and plotting:

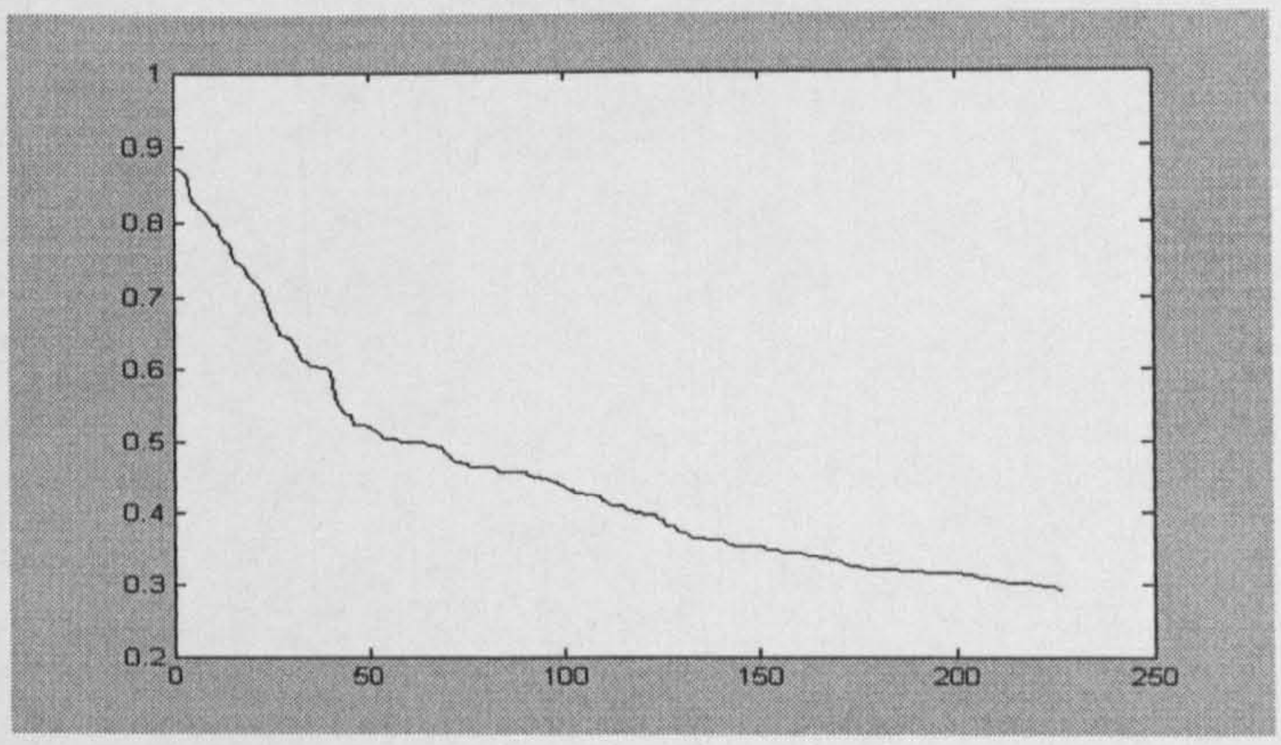


Figure 37: Correlations between 220 most-correlated variables in Q1

For other tests to determine the appropriateness of this type of dimensionality reduction to data, see Hair *et al.* (1998, 99-100);

a) Q3

The covariance matrix COV_{Q1} was calculated and its eigenvalue and eigenvector matrices $EVAL_{COVQ1}$ and $EVECT_{COVQ1}$ derived. The eigenvalues on the main diagonal of $EVAL_{COVQ1}$ were ordered in descending order of magnitude and then plotted to determine which of the eigenvectors were to be retained and which discarded:

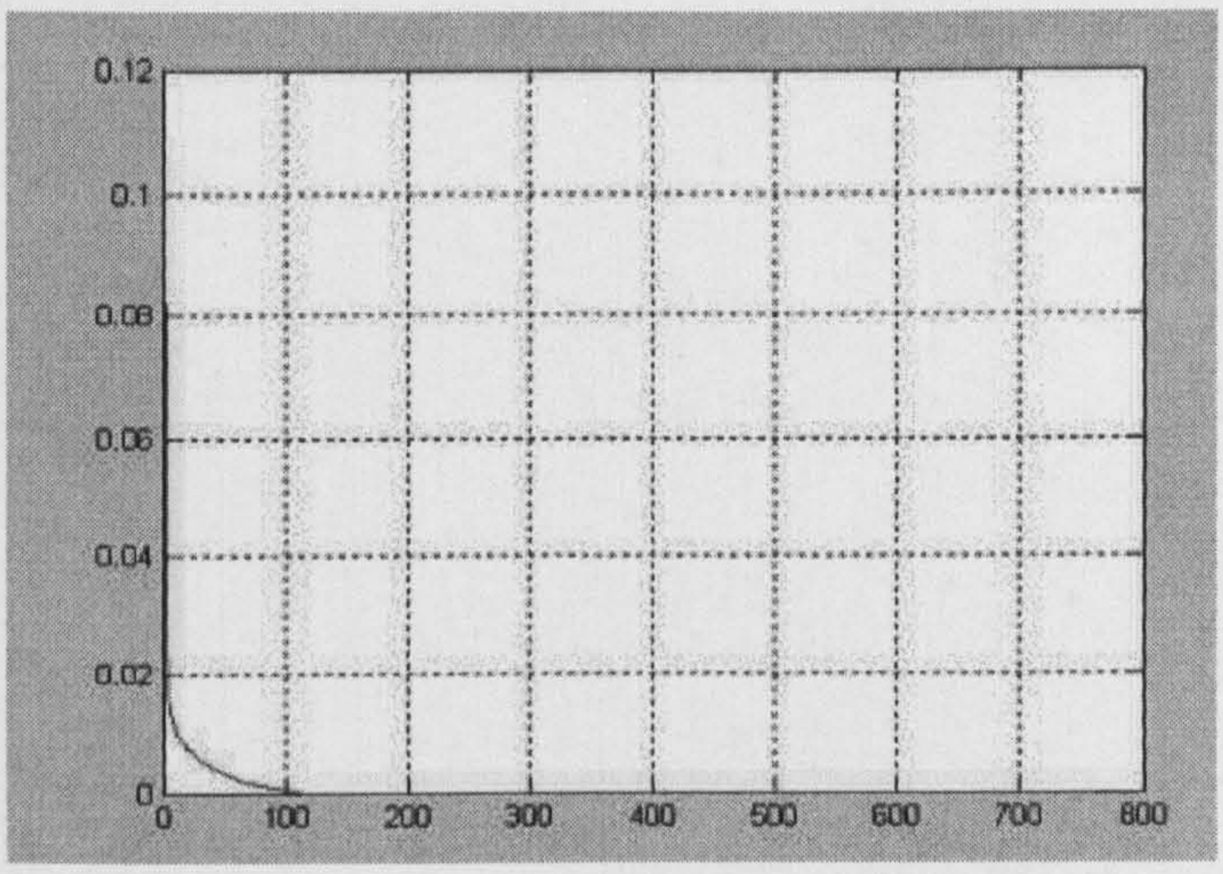


Figure 38: Eigenvalues of COV_{Q1} in descending order of magnitude

This graph shows that the variance in the original 761 variables can be represented with minimal loss of information in the 100-dimensional space defined by the 100 eigenvectors corresponding to the above 100 largest eigenvalues. The 761-dimensional matrix Q1 was then projected into this 100-dimensional space using the formula presented earlier, resulting on the 114 x 100 matrix Q3:

$$D_R{}^T = E^T D^T$$

where D_R^T is the transpose of the reduced data matrix, Q3, E^T is the transpose of the matrix consisting of the first 100 columns of the eigenvector matrix $EVECT_{COVQ1}$, and D^T is the transpose of the original matrix Q1.

b) Q4

Q4 was created from Q2 in exactly the same way as Q3 from Q1, and the procedure does not need to be described in detail again. The eigenvalue plot for the 1236-dimensional matrix Q2 looks like Figure 39a:

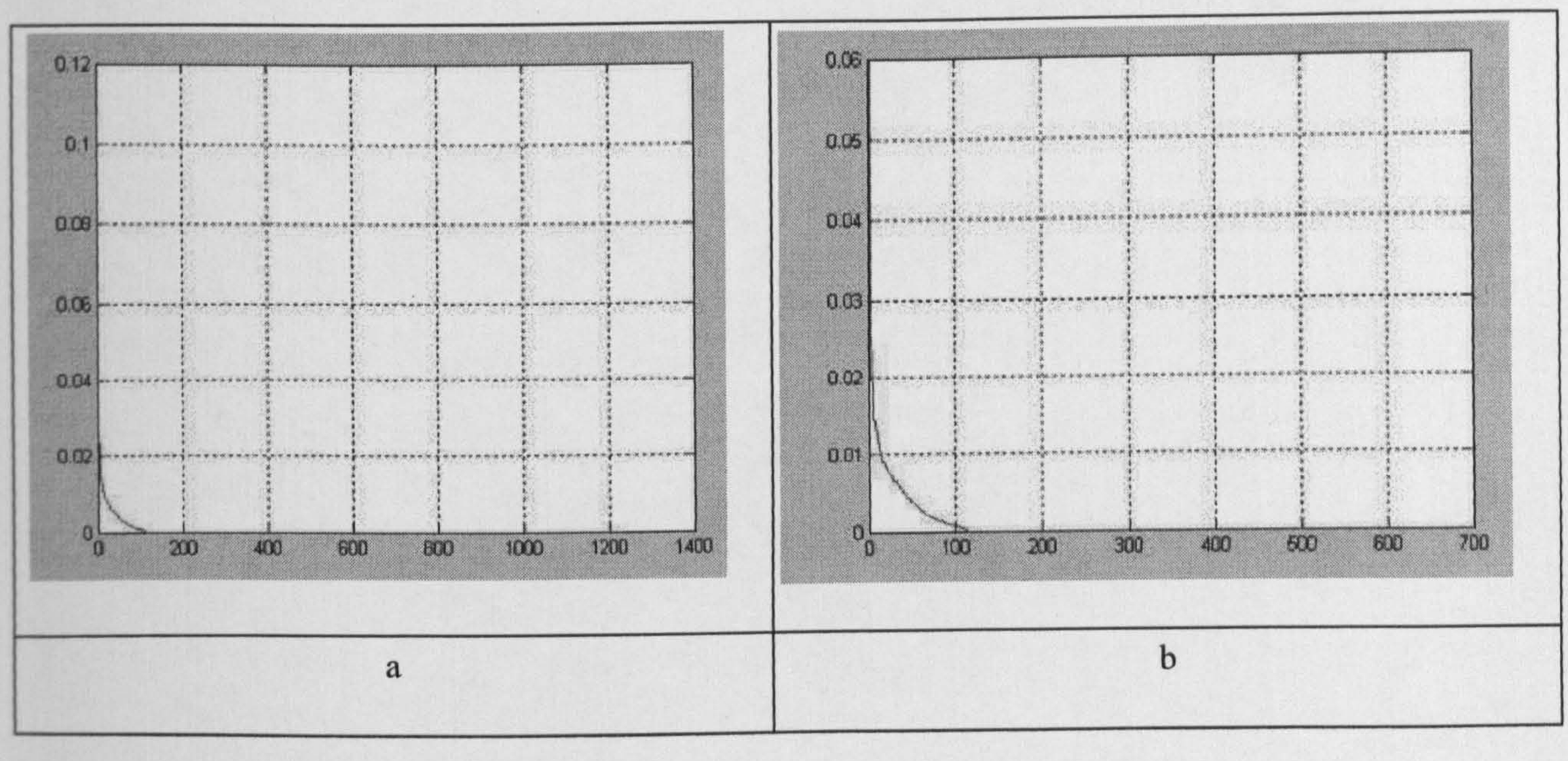


Figure 39: Eigenvalues of COV_{Q2} in descending order of magnitude

Figure 39b is a slightly higher-resolution version of 39a, and shows that, like Q1, Q2 can be dimensionality-reduced to 100, resulting in the 114 x 100 matrix Q4. Inclusion of the additional lexical types in Q2 appears to have added little if any variability to the data, and thus to have made a commensurately minimal contribution to distinguishing the *suras* from one another.

c) Implementation

PCA is provided by all the standard statistical packages like SPSS and SAS, and any of these could have been used to derive Q3 and Q4. For simplicity and conceptual clarity, however, relevant Matlab functions were used to calculate the covariance, eigenvalue, and eigenvector matrices, and *EditMatrix* (Appendix 1) was used to project Q1 and Q2 into Q3 and Q4 respectively using the Matlab-derived eigenvector matrices. Specifically, for Q3:

- Q1 was imported into the Matlab workspace using the *import* function.
- The covariance matrix for Q1 was calculated using the Matlab function *cov*: COV = *cov* (Q1)
- Eigenvalue and eigenvector matrices for COV were derived using the Matlab *eig* function: [EVAL EVECT] = *eig* (COV)
- EVAL and EVECT were exported from Matlab using the *fprintf* function
- EVAL and EVECT were imported into *Editmatrix*
- *Editmatrix* was used to abstract and output the vector of eigenvalues on the main diagonal of EVAL. This vector was then re-imported into Matlab using the *import* function.
- On the basis of the information provided by the plot, *Editmatrix* was used to select the 100 eigenvectors corresponding to the 100 largest eigenvalues, and then projected the original 761-dimensional matrix Q1 into the 100-dimensional space defined by those eigenvectors, yielding Q3.
- *Editmatrix* exported Q3 for subsequent processing.

Q4 was generated in the same way. Details of the Matlab functions can be found in the Matlab online Help system; for the relevant *EditMatrix* facilities see Appendix 1.

5.2.7 Dealing with nonlinearity

Given the difficulty of determining the presence of nonlinearity in high-dimensional data, no attempt is made to do so with respect to the Qur'an data. Given also the implications of nonlinearity for analysis, however, the issue cannot be ignored, and as such some way of accommodating the possibility that the Qur'an data contains nonlinearities must be found. The foregoing discussion provides two possibilities: remove the problem by linearizing the data, or use an analytical method that can take account of nonlinearity. The present discussion adopts the latter alternative on the grounds that such nonlinearities as may exist in the data might be of interest in relation to the research question, and simply removing them limits the analysis a priori. The issue of linearity is, therefore, deferred for the time being, and will be taken up again when selecting an analytical method or methods for the Qur'an data.

Chapter Six

Data Analysis

6.0 Introduction

This chapter analyzes the data created in the preceding one using exploratory multivariate analytical methods. It is in two main parts. The first part explains what is meant by exploratory multivariate analysis, gives an overview of available methods, and presents a detailed account of the methods selected for analysis of the Qur'an data. The second then applies these methods to the data and compares the results.

6.1. General principles

6.1.1 Exploratory multivariate analysis

Observation of nature plays a fundamental role in science. In current scientific method (Chalmers 1999; Ladyman 2002), a hypothesis about some natural phenomenon is proposed and its adequacy is assessed using data obtained from observation of the domain of inquiry. But nature is dauntingly complex, and there is no practical or indeed theoretical hope of being able to observe even a small part of it exhaustively. Instead, the researcher selects particular aspects of the domain for observation, basing the selection on some combination of acquaintance with existing theory in the relevant scientific discipline, personal intuition derived from experience of the domain, and specific research aims. Each selected aspect is represented as a variable, and a series of observations is conducted in which, at each observation, the values for each variable are recorded. A body of data is thereby built up on the basis of which a hypothesis can be assessed. One might choose to

observe only a single aspect --height among humans, say-- in which case the data set consists of more or less numerous values assigned to one variable; such a data set is said to be univariate. If two variables are observed, say height and weight, then the data set is said to be bivariate, if three trivariate, and so on up to some arbitrary number *n*. Any data set in which $n > 1$ is said to be multivariate.

As the number of variables grows, so does the difficulty of understanding the data, that is, of conceptualizing the interrelationships of variables within a single data item on the one hand, and those between and among multiple data items on the basis of their variable values on the other. Multivariate analysis is the computational use of mathematical and statistical tools for understanding such interrelationships in data.

Numerous techniques for multivariate analysis exist. They can be divided into two main categories (Hair *et al.* 1998, 18-22) which are usually referred to as 'exploratory' and 'confirmatory'. Exploratory analysis aims to discover regularities in data which can serve as the basis for formulation of hypotheses about the domain from which the data comes. Such techniques emphasize intuitively accessible, usually graphical representations of data structure. Confirmatory multivariate analysis attempts to determine whether or not there are significant relationships between some number of selected independent variables and one or more dependent ones. These two types are complementary in that the first generates hypotheses about data, and the second tries to determine whether or not such hypotheses are valid. The research question directing the present discussion is by nature exploratory, since it aims to generate hypotheses about the conceptual structure of the Qur'an. As such, the remainder of the discussion is concerned only with exploratory methods.

For introductory, conceptually-oriented accounts of multivariate analysis in general and exploratory multivariate analysis in particular see Kachigan (1991), Grimm & Yarnold

(1995), and Grimm & Yarnold (2000). More mathematically and statistically technical are Tukey (1977), Hair *et al.* (1998), Everitt & Dunn (2001), Tinsley & Brown (2000), and Tabachnik & Fidell (2001).

6.1.2 Method selection

Exploratory analytical methods are all variations on a theme: cluster analysis. Cluster analysis aims to identify and --usually-- graphically to represent nonrandomness in the distribution of vectors in *n*-dimensional space. Spatial regularities in the graphical representations are interpreted as reflecting regularities in the process that generated the data, and support hypotheses about the characteristics of the process. In the plot on the left in Figure 40, for example, the vectors are spread more or less uniformly in two-dimensional space; there are some local concentrations, but these are not clearly defined and it is difficult to infer anything about the process that generated the data other than that it appears to be broadly random. In the plot on the right, on the other hand, there are clearly defined concentrations of vectors such that two groups of points are spatially relatively close in each group, and spatially relatively far from each other, which suggests that the generating process is strongly nonrandom. In fact, the data on the left was produced by a random number generator, and that on the right by a known bipolar process.

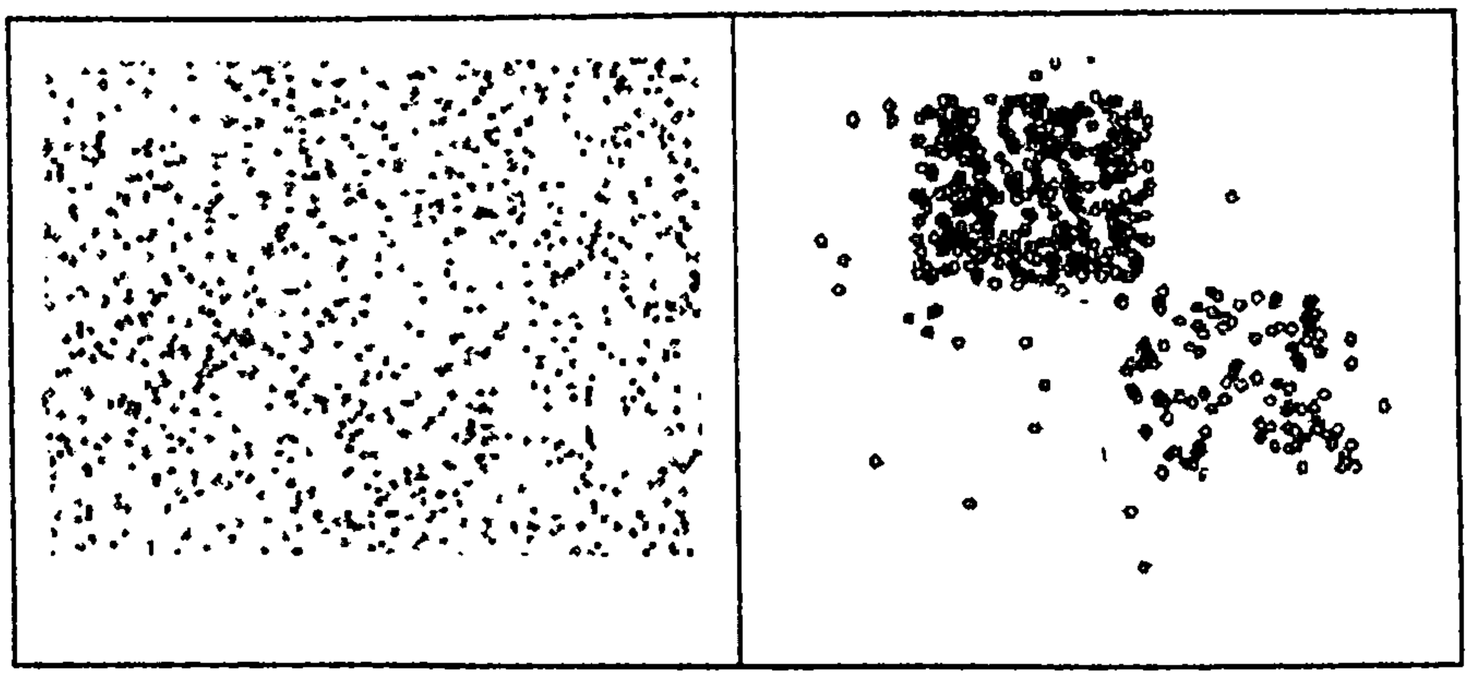


Figure 40: Random and nonrandom data

In two or three dimensions, such distributions can be plotted directly and interpreted by eye. In higher dimensions this is no longer possible, however; the various cluster analysis methods are just different ways of representing nonrandom structure in higher-dimensional data graphically in two or three dimensional space.

There is an extensive and at times bewildering range of cluster analysis methods together with a vast associated literature. These are covered in the various multivariate analysis textbooks cited above (Grimm & Yarnold 1995; Kachigan 1991; Hair *et al.* 1998; Grimm & Yarnold 2000; Tinsley & Brown 2000; Everitt & Dunn 2001; Tabachnik & Fidell 2001), as well as in more specialized accounts such as Gordon (1987), Jain & Dubes (1988), Gordon (1992), Arabie *et al.* (1992), Gordon (1999), Jain *et al.* (1999), Manning & Schütze (1999, ch. 14), Gore (2000), Duda *et al.* (2001), Everitt *et al.* (2001), and Webb (2002). There is no hope of being able to use all, or most, or even many of these methods within the scope of a study like this one. As such, the range of choice must be narrowed down in a principled way.

The first step in doing this is to make a distinction between data classification and data categorization. Classification as understood here makes no a priori assumptions about the structure of data: it tries to discover structured interrelationships among data items that might be interesting in relation to a research question. Categorization, on the other hand, does make an important a priori assumption: it presupposes a set of categories to which data items are to be assigned. The research question that the present study addresses is intrinsically classificatory in that the structural interrelationships among the *suras* is what is to be discovered, not prespecified or hypothesized. As such, categorization methods are not further considered. These include *k*-means algorithms and kernel-based adaptive

algorithms such as support vector machines and generalized topographic maps (see survey in Webb 2002).

Also eliminated from consideration are classification methods based on finite mixture density modelling (Everitt *et al.* 2001, ch. 6). These methods provide a firm statistical basis for data analysis in attempting to partition data items into subsets generated by processes with different probability density functions, and thus seem an attractive option. To work reliably, however, they require reasonably large sample sizes, and the 114 data items available to this study do not qualify as 'reasonably large'.

Beyond this, the criteria for choosing among the still-extensive range of methods become less clear-cut. All the textbooks agree that there is no 'best' method or class of methods (see for example Everitt *et al.* 2001, ch.8), and that the choice must be made in the light of various considerations, the most important ones being:

1. The selected method or methods must generate results that are useful in relation to the research question. The fundamental aim of exploratory analysis is to generate hypotheses about some domain of inquiry, and it may be that, in any particular case, some methods provide representations of structure that do this more usefully than others. The main distinction among methods in this regard is between those that generate hierarchically ordered clusters, and those that do not and are therefore described as nonhierarchical. Nonhierarchical methods generate graphical representations in two or three dimensional space such that, given a suitable measure of proximity, vectors which are spatially or topologically relatively close to one another in high-dimensional space are spatially or topologically close to one another in their two or three dimensional representation, and vectors which are relatively far from one another in high-dimensional space are clearly separated, either by relative spatial

distance or by some other graphical means, resulting --in the case of nonrandom data-- in a configuration of well defined clusters. Figure 40 above is a two-dimensional example; a three-dimensional one might look like Figure 41:

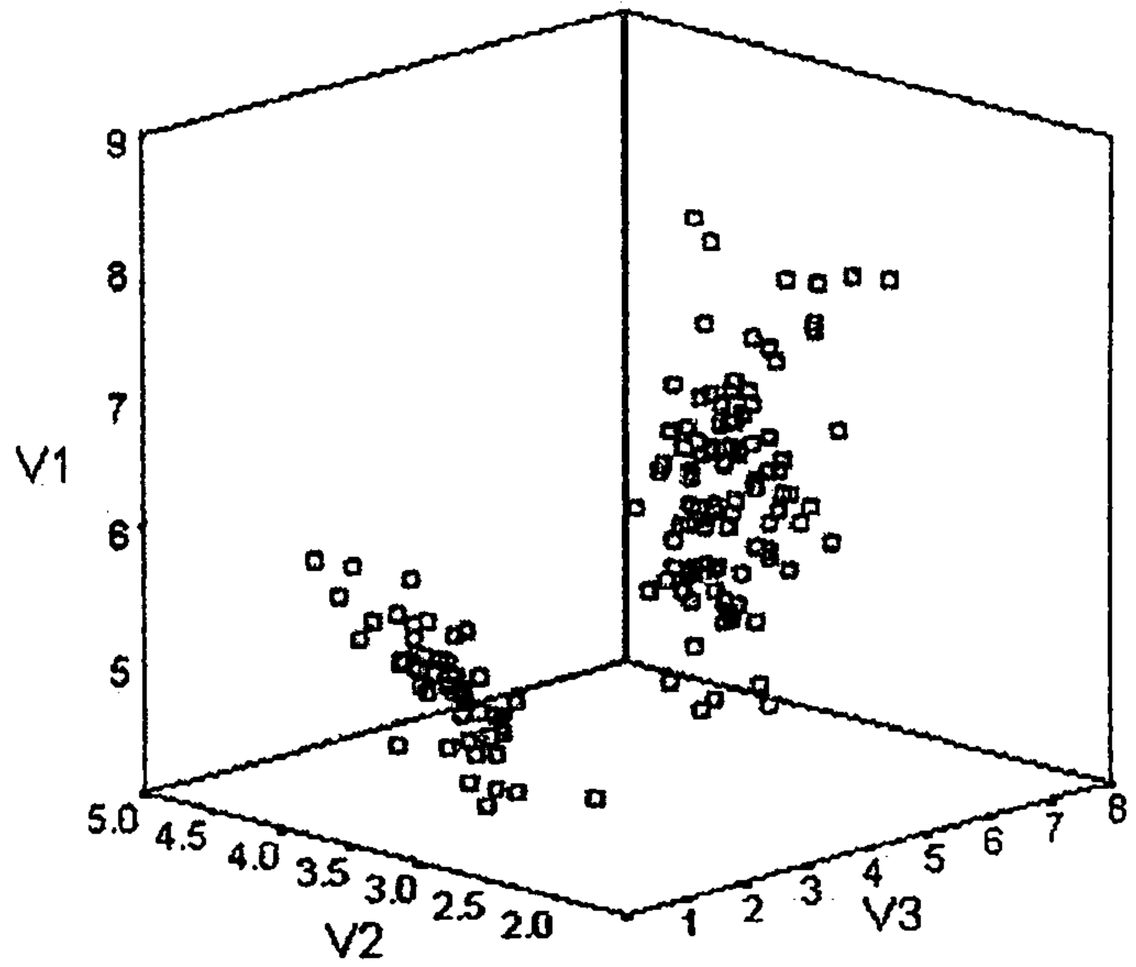


Figure 41: Clusters in 3-dimensional space

Hierarchical methods, on the other hand, represent proximity structure in high-dimensional data not as spatial clusters but as 'dendrograms':

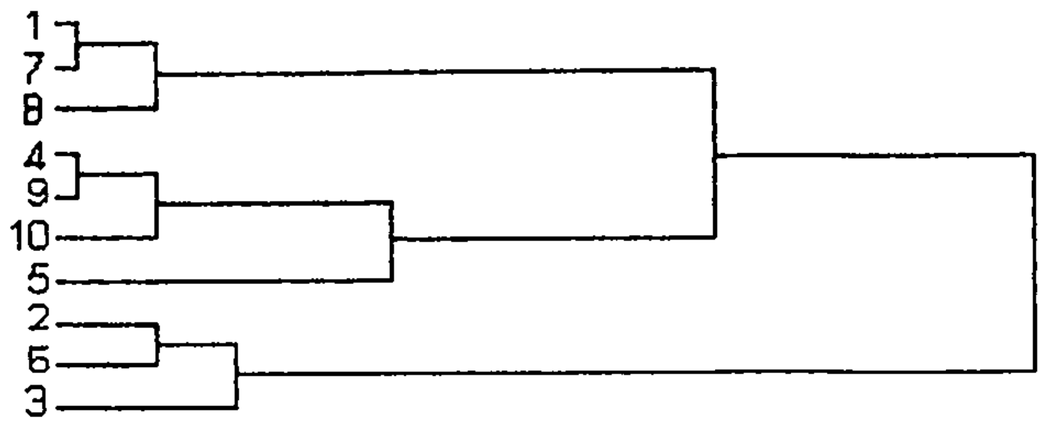


Figure 42: A cluster dendrogram

A dendrogram is simply a tree of the kind linguists are familiar with from sentence structure analysis. It is shown horizontally rather in the vertical orientation that is more

usual in linguistics in order to make it more readily representable on a page, and the labels at the 'leaves' are not lexical tokens but labels for the vectors in the data set --'1' is the first vector, '2' the second, and so on. Like a linguistic phrase structure tree, a dendrogram shows constituency structure: in this tree, vectors 1 and 7 constitute a 'phrase' that combined with vector 8 to form a superordinate phrase, which itself combines with the (4,9,10,5) 'phrase' to form an even higher-level 'phrase', and so on. Unlike a linguistic phrase structure tree, however, this one represents not grammatical constituency but vector proximity in *n*-dimensional space: vectors 1 and 7 are relatively very close and both of them are quite close to 8; vectors 4 and 9 are relatively close and both are quite close to 10; and so on. And, again unlike grammatical phrase structure trees, the lengths of the branches linking 'phrases' represent relative degrees of proximity: that the lines for linking 4 and 9 are relatively very short indicates that the corresponding vectors are close in n-dimensional space, but the relatively long lines between (1,7,8) and (4,9,10,5) indicate considerable distance in n-dimensional space. In the light of this, the cluster interpretation of the above tree is straightforward: there are two main clusters: (1,7,8,4,9,10,5) and (2,6,3); within each of the two main clusters there are subclusters (1,7,8) and (4,9,10,5) on the one hand, and (2,6) and (3) on the other; and so on.

2. The selected method or methods cannot be expected to provide the 'true' cluster structure of the data manifold. There are two main reasons for this. On the one hand, each method either implicitly or explicitly makes assumptions about what constitutes a cluster and how clusters so defined can be algorithmically identified; different methods can and often do generate different cluster results for the same data. On the other, all the methods depend to greater or lesser degrees on parameter values that are user-supplied; again, different parameter values can and do yield different results for

constant data. In such a situation, it is not obvious which method and/or combination of parameter values is to be preferred in any given application, or why. This leads to an obvious question: what are these clustering algorithms really telling us about the structure of the data they describe --how reliable, in other words, are these methods, and are they in fact of any use at all if they cannot be relied on to reveal the true vector density structure of the data?

In the literature there are two main approaches to an answer. One is to attempt to establish the validity of cluster results using numerical measures: there are measures of the degree of structure in data, of the quality and robustness of individual clusters, and of the relative goodness of results generated by different clustering methods. The other approach is to apply a variety of different clustering methods to the same data and to compare the results: a clear convergence on one particular cluster structure is held to support the validity of that structure with respect to the data. And, of course, the two approaches can be used in combination (for a summary of cluster validation approaches and methods see Everitt *et al.* 2001, ch.8; Duda *et al.* 2001, 557-9).

3. The selected method or methods must be compatible with the data being analyzed. For continuous-valued data such as that being discussed here, the main criterion for compatibility is whether the data manifold is linear or not. Data that contains significant nonlinearity must be analyzed using a nonlinear clustering method; use of a linear method in such a case misrepresents the structure of the data to greater or lesser degrees, depending on the nature of the nonlinearity. What does it mean for a method to be linear or nonlinear? Assume a curved manifold in n-dimensional space, as discussed earlier. What is the distance d_{ij} between any two points i and j on that manifold? A linear method measures that distance as a straight line joining the points,

ignoring the manifold's curvature, whereas a nonlinear method measures the distance along the surface of the manifold, thereby taking account of the curvature. Depending on the amount of curvature, the difference between the two measures can be significant, and can therefore significantly affect analysis based on it. An example is the distance between two points A and B on the perimeter of the circle in Figure 43: the linear distance between them is a chord drawn through the interior, and the nonlinear one is the length of the perimeter segment between the points as indicated by the arc in the figure:

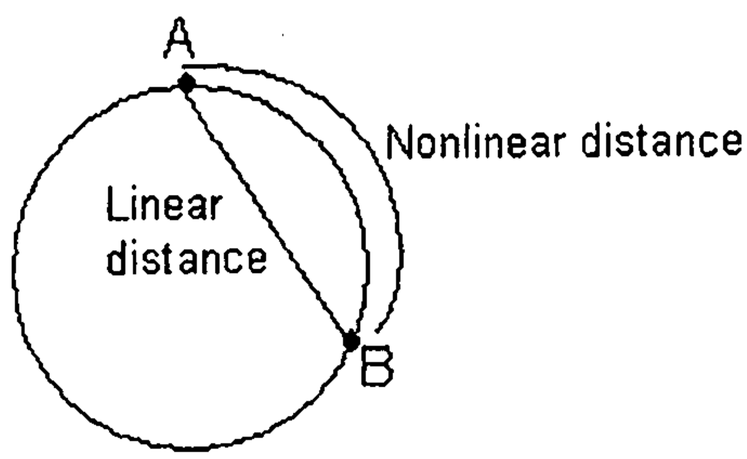


Figure 43: Linear and nonlinear distance

In the light of (1) above, hierarchical methods provide more information than nonhierarchical ones in that they not only identify the main clusters, but also their constituency relations relative to one another as well as their internal structures. For this reason, hierarchical methods will be used in the present analysis. In the light of (2), however, it is recognized that a single class of methods cannot safely be relied on, and that at least one additional method or class of methods must be used to corroborate the results from hierarchical analysis. And, in the light of (3), because hierarchical methods are linear (on which more later), the additional method or methods used must be nonlinear to provide for the possibility that the Qur'an data contains significant nonlinearities. A good range of methods satisfies these requirements --for example, multidimensional scaling (Borg &

Groenen 2005), Sammon's mapping (Sammon 1969), principal curves (Hastie & Stuetzle 1989), Isomap (Tenenbaum *et al.* 2000), locally linear embedding (Roweis & Saul 2000), self-organizing maps (Kohonen 2001), together with their fairly numerous variants. The self-organizing map, or SOM, was selected partly on pragmatic grounds, and partly on theoretical ones. The pragmatic grounds are that the SOM is well established: it is probably the most widely used of the available nonlinear clustering methods (Kaski *et al.* 1998; Oja *et al.* 2001), and as such its characteristics are well understood and documented in an extensive literature. It is, moreover, the basis for WEBSOM, a document classification system that has classified millions of Web documents in a way that is very close to what is required here, and thus provides an exemplary case study --see for example Kohonen *et al.* (2000), Lagus (2002), Lagus *et al.* (2004), as well as the WEBSOM website. The theoretical grounds have to do with the need to corroborate the hierarchical analytical results. The hierarchical methods use linear measures of spatial distance among data vectors as the basis for clustering, whereas the SOM uses projection of the data manifold topology into low-dimensional space. More is said about both of these in due course; the important thing to realize at this stage is that they are fundamentally different clustering algorithms, and that substantial coincidence of results from them would therefore provide strong corroborative justification for asserting the objective reality of those results in the data.

In what follows, therefore, the Qur'an data is first analyzed using hierarchical methods, and then using a SOM. Finally, the results are compared with the aim of determining the degree, if any, of corroboration between them.

Before going on to the analyses, a brief note is required on the computer software used to carry them out. The main cluster analysis software up to c.2000 is reviewed in Everitt *et al.*

(2001, 197ff). As every researcher is aware, however, the World Wide Web has become the prime source of up-to-date information on a wide range of topics, including newly-developed software applications and updates of existing ones. Specific URLs are not given here because, in the present state of the Web, these can be ephemeral. Instead, it is assumed that the reader will use a Web search engine to locate the relevant sites.

- The standard statistical packages such as SPSS, SAS, and Statistica provide good implementations of hierarchical cluster analysis, but the current version of Clustan is used here partly because of the breadth of its coverage of hierarchical methods, partly on account of its convenience of use and attractive graphics, and partly because it was readily available in the IT environment in which the present research was conducted.
- SOM implementations are listed by the SOM Toolbox Web page at the Adaptive Informatics Research Centre in the Helsinki University of Technology. The one used for the present research is a modified version of the SOM Toolbox, the Matlab source code of which is freely available from this site. It was selected because it could readily be modified according to need.

6.1.3 Hierarchical cluster analysis

The aim of hierarchical cluster analysis has been said to be identification and graphical representation of nonrandomness in the distribution of data vectors in *n*-dimensional space as nested constituency trees (detailed accounts of hierarchical cluster analysis are in Everitt *et al.* 2001, Gordon 1999, and Gore 2000. For briefer discussions see for example Everitt & Dunn 2001, Hair *et al.* 1998, Jain *et al.* 1999, Kachigan 1991, Oakes 1998, Gore 2000, and Duda *et al.* 2001, ch.10). The remainder of this section describes these hierarchical

methods in three subsections: proximity measurement, clustering algorithms, and interpretation of results.

a) Proximity measurement

The first step in a hierarchical cluster analysis is to create a matrix whose cells contain a measure of the similarities or, alternatively, the dissimilarities in *n*-dimensional space of all pairs of data vectors; to avoid terminological confusion, the generic term 'proximity' is used to cover both similarity and dissimilarity. Proximity between vectors can be measured in terms of their correlation, of the angle between them, or of distance in Euclidean space (Everitt *et al.* 2001, ch. 3). These are closely related, and if all the variables are measured on the same scale or have been standardized, there is no particular reason to prefer one over another. Distance is the most common measure and is best provided for in software implementations, and so is used here.

Seen geometrically, the coordinates of data points in an *n*-dimensional space define the positions of the points relative to one another in that space. Various measures can be used to calculate the distance from one point to another.

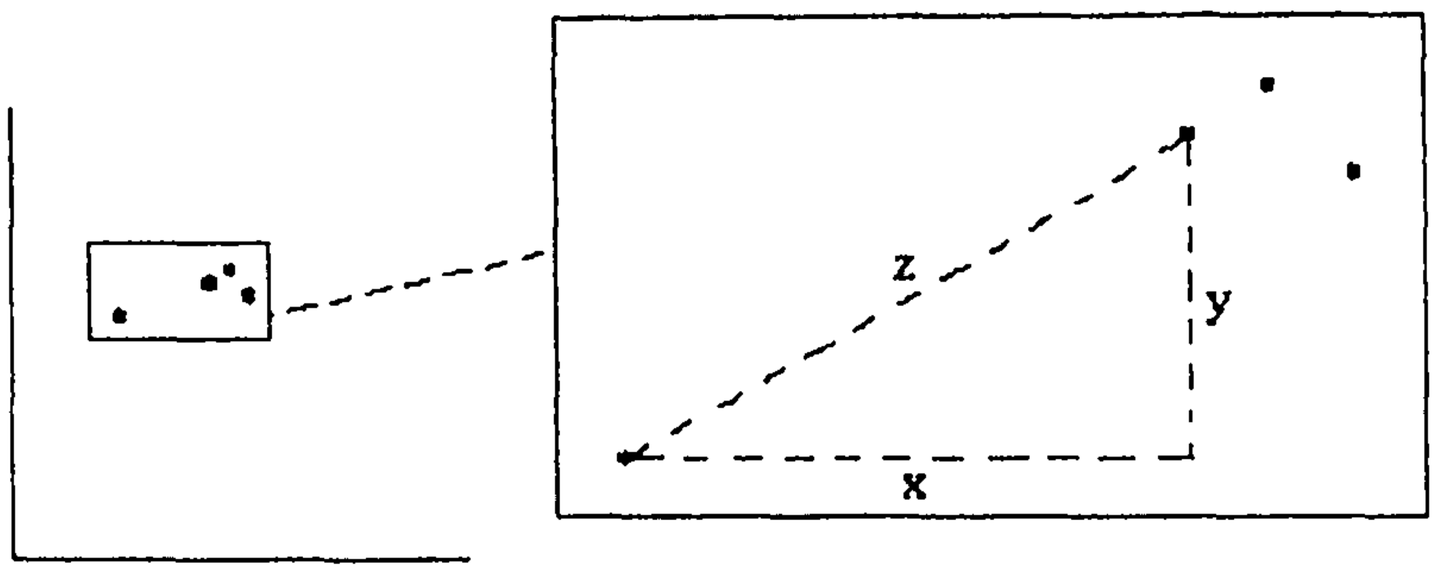


Figure 44: Euclidean distance measure

In Figure 44 the distance between the two points at the vertices of the triangle is

$$length(z) = \sqrt{(length(x))^2 + (length(y))^2}$$

This is the Euclidean distance, the simplest and most widely used of the various available distance measures. Others (Gordon 1999, 15-34; Jain *et al.* 1999, 271-4; Duda *et al.* 2001, ch. 10; Everitt *et al.* 2001, ch.3) have particular characteristics that may be useful in certain applications. Squared Euclidean distance, for example, weights the distance among vectors in a nonlinear way by squaring the Euclidean distance, as its name indicates; by putting progressively greater weight on vectors that are further apart, it accentuates the degree of separation among them and may help in delineating clusters more clearly. Another measure, Mahalanobis distance, is appropriate when variables are measured on different scales --Euclidean or squared Euclidean distance calculation is adversely affected by such disparities of scale, but the Mahalanobis measure contains a standardization procedure which compensates for this. As Everitt *et al.* 2001, 52-4 point out, different proximity measures can and do lead to different cluster solutions, and as such it would be extremely useful to be able to select a measure that is in some sense optimal. No reliable selection procedure exists, however. The choice of measure in any given application is governed by the nature of the data and by the clustering algorithms that will use it. For example, in the present instance, the Mahalanobis measure is unnecessary because all variables are frequency counts and thus measured on the same scale. In fact, squared Euclidean distance is used in the analyses that follow because one of the clustering algorithms, Ward's method, requires it.

A note on linearity is appropriate at this stage. The distance measures generally discussed in the literature with reference to hierarchical methods, as well as those standardly provided in software implementations in general and in Clustan more particularly, are linear. Even measures like the squared Euclidean and increase in sum of squares, which

include an exponential term, must be considered linear for present purposes because both are based on a linear, that is, Euclidean, measure of distance between vectors. To be considered nonlinear, distance would have to be measured along the surface of the manifold, and none of these do that.

b) Cluster definition

The second step in hierarchical analysis is to build a cluster tree using the proximity matrix. There are two main approaches for doing this: top-down or divisive, where a set of *n* vectors is successively subdivided into ever-finer subgroups, and bottom-up or agglomerative, where the *n* initially-separate vectors are successively merged into ever-larger groups. Agglomerative methods are the more often used and also better covered in the relevant literature, and for that reason the remainder of the discussion deals only with them.

The generic agglomerative algorithm is as follows:

1. Initially, each vector is defined as a cluster
2. Using as many steps as necessary, at each step combine the two nearest clusters to form a new, composite cluster, thus reducing the number of clusters by 1
3. When only one cluster containing all the vectors remains, stop.

At the first step in (2), determination of which clusters to combine is simply a matter of finding the smallest value in the proximity matrix and combining the relevant two vectors into a new, composite cluster. Thereafter, however, it becomes necessary to know not only the distances between single-vector clusters, but also between single-vector and composite clusters and, increasingly as the steps proceed, between two composite clusters. These distances are not found in the original proximity matrix and have to be computed at each

step, that is, each time a new cluster is created. To do that, it is necessary to define the notion of distance between clusters one or both of which might be composite. Various definitions exist; four of the most commonly discussed in the literature and most often used are:

- Single link, which defines the distance between two clusters A and B to be that between the two closest vectors, or 'nearest neighbours', in the respective clusters.

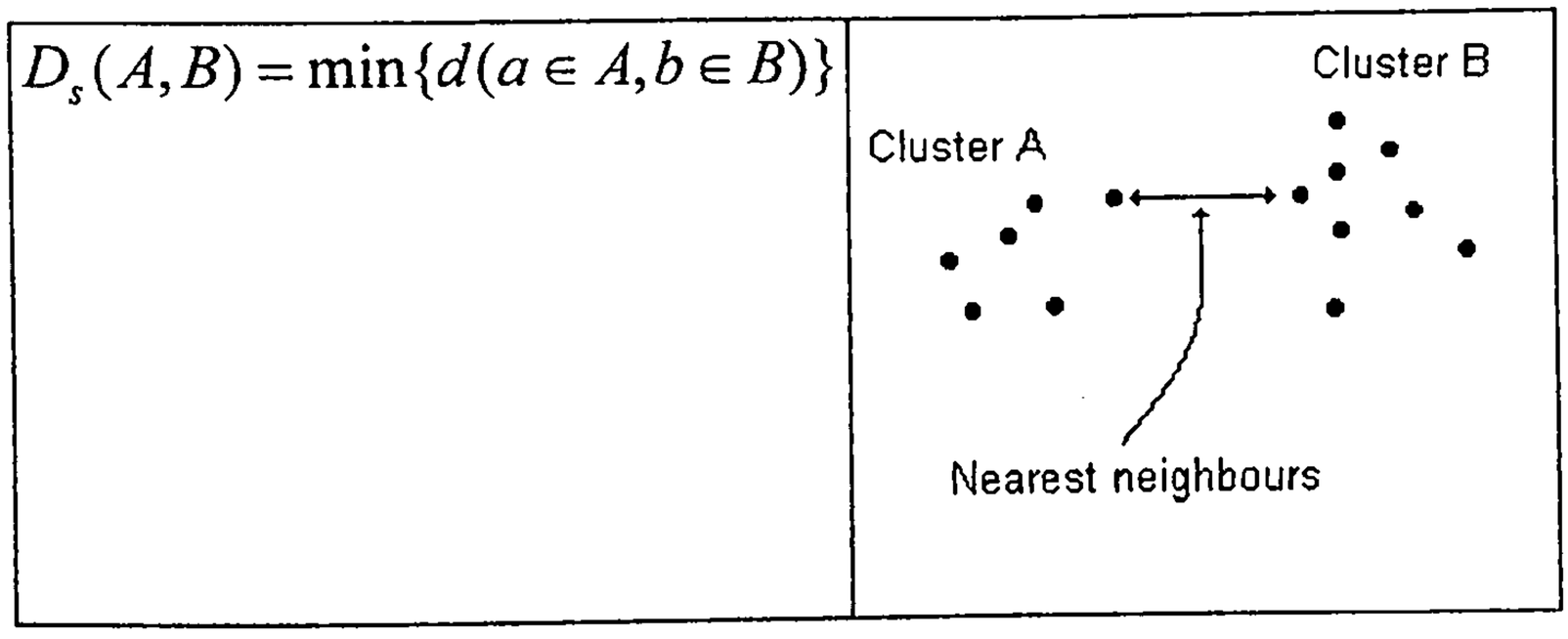


Figure 45: Single link clustering

- Complete link, which defines the distance between two clusters A and B to be that between the two vectors in the respective clusters that are furthest apart, or 'furthest neighbours'.

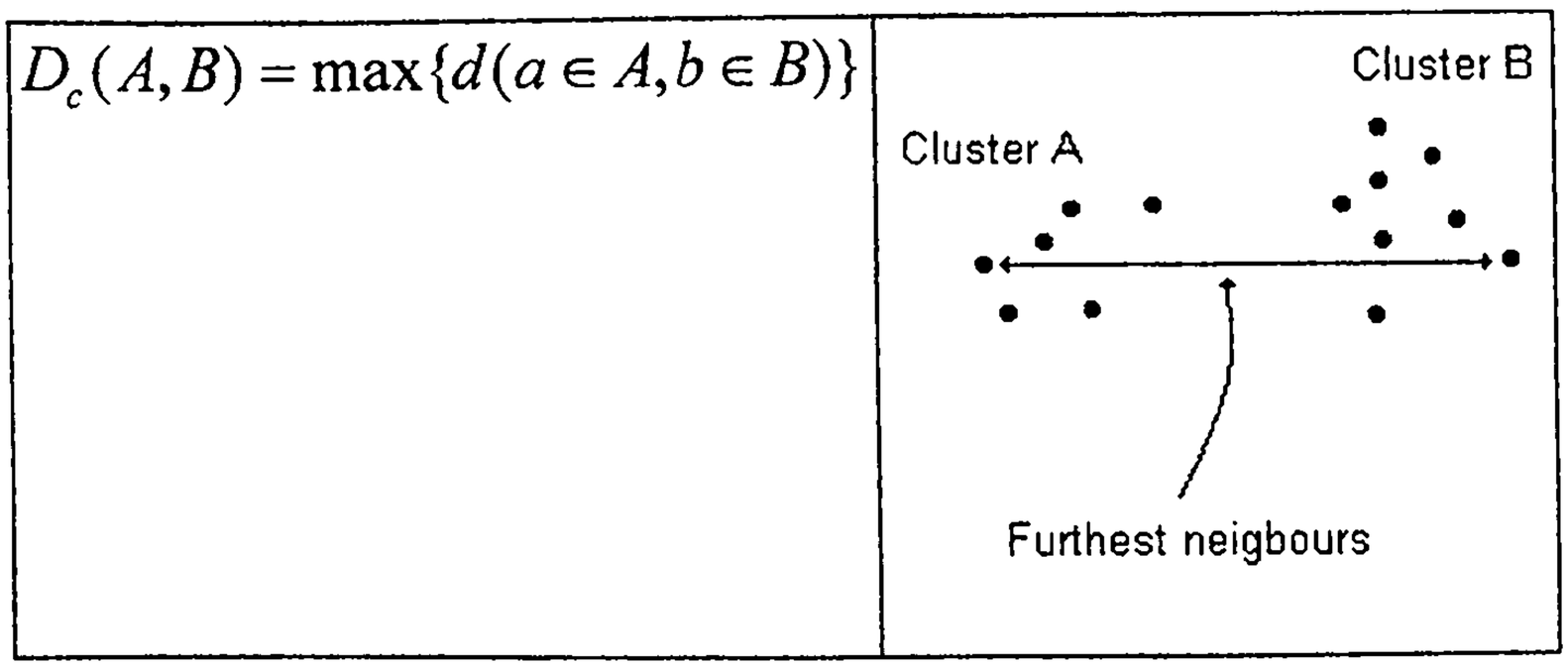


Figure 46: Complete link clustering

- Average link, in which the distances between all ordered pairs of vectors in the above two clusters A and B are measured and summed, and the distance between A and B is the mean of that sum:

$$D_{avg}(A,B) = \frac{\sum_{1=1..m,j=1..n} d(a_i \in A, b_j \in B)}{m \times n}$$

where $D_{avg}(A,B)$ is the average link distance between A and B, d is the distance between a single pair of vectors, m is the cardinality of cluster A, and n is the cardinality of cluster B.

- Ward's method, which involves the notion of sum-of-squares error, abbreviated SSE. Given a set D of n values, the SSE of D is the sum of the squared differences between each value in D and the mean of all values in D:

$$SSE_D = \sum_{i=1..n} \left| d_i \in D - \frac{\sum_{j=1..n} d_j \in D}{n} \right|^2$$

Ward's method calculates the distance between clusters A and B as

$$D_{Ward} = SSE(A,B) - \left(SSE(A) + SSE(B)\right)$$

Since the calculation both of the values in the original proximity matrix and of the distances between composite clusters are based on linear measurement, hierarchical analysis is a collection of linear cluster analysis methods.

c) Interpretation

Because hierarchical cluster analysis is conceptually simple, software implementations of it are generally easy to use, and graphical representation of results as constituency trees is beguilingly clear, there is a strong temptation to use it uncritically --a nicely structured tree is difficult to resist. It has, however, already been noted that different cluster analysis methods can and often do generate different results for the same data, and this applies to hierarchical methods in particular. Extensive empirical results have shown that, relative to a given data matrix, each clustering algorithm has a 'signature' in the sense that the trees it generates tend to have specific characteristics (Everitt *et al.* 2001, ch. 4; Jain & Dubes 1988; Webb 2002, 396-400). Single link famously tends to generate 'chained' structures, that is, trees with a strong tendency to either left or right branching but not both; complete link tends to generate trees with extensive recursive embedding of left and right branching subtrees; average link is intermediate between single and complete link; Ward's method is like complete link, but in addition tends to find spherical clusters of roughly equal size. As such, some methods are more appropriate than others for data with a given density structure. If, for example, the data manifold has an elongated structure, single link would be best and Ward worst. Alternatively, a manifold with well-defined spherical areas of vector density would reverse that. And so on. The problem, of course, is that in exploratory analysis of high-dimensional data the shape of the data manifold is unknown or at least very unclear by definition --finding the manifold structure is the object of the exercize-- and the choice of appropriate hierarchical clustering algorithm is consequently imponderable. Any hierarchical cluster analytical result must, therefore, be validated either by numerical measures of validity or by corroboration from results generated by nonhierarchical methods such as self-organizing maps, or both.

6.1.4 Self-organizing maps (SOM)

The SOM is an artificial neural network that was originally invented to model a particular kind of biological brain organization. It can, however, be used as a data analysis tool without reference to biology, and has been so used in a very wide range of applications (Kaski *et al.* 1998; Oja *et al.* 2001). In data analysis, it is a method for projecting data of arbitrary dimensionality into two-dimensional space and visualizing any structure in the data in a variety of ways. This section first describes the architecture of the SOM and then addresses some theoretical and practical issues that must be kept in mind when applying it to data analysis.

The standard reference work for SOMs is Kohonen (2001). Briefer accounts can be found in most artificial neural network textbooks --for example Ritter *et al.* (1992), Verleysen (1997), Gurney (1997), and Haykin (1999), see also the collections by Oja & Kaski (1999) and Allinson *et al.* (2001). What follows is based in large part on these sources.

a) SOM architecture

A good initial intuition for what a SOM is and how it works can be gained by visualizing it as a physical system, but precise understanding requires a mathematical account. The discussion therefore begins by presenting a physical SOM model to establish the requisite intuition, and the constructs the corresponding mathematical model on the basis of the physical one.

i. *Physical SOM model*

A physical SOM has three components: an input buffer, a two-dimensional lattice of processing units, and connections between the buffer and the lattice:

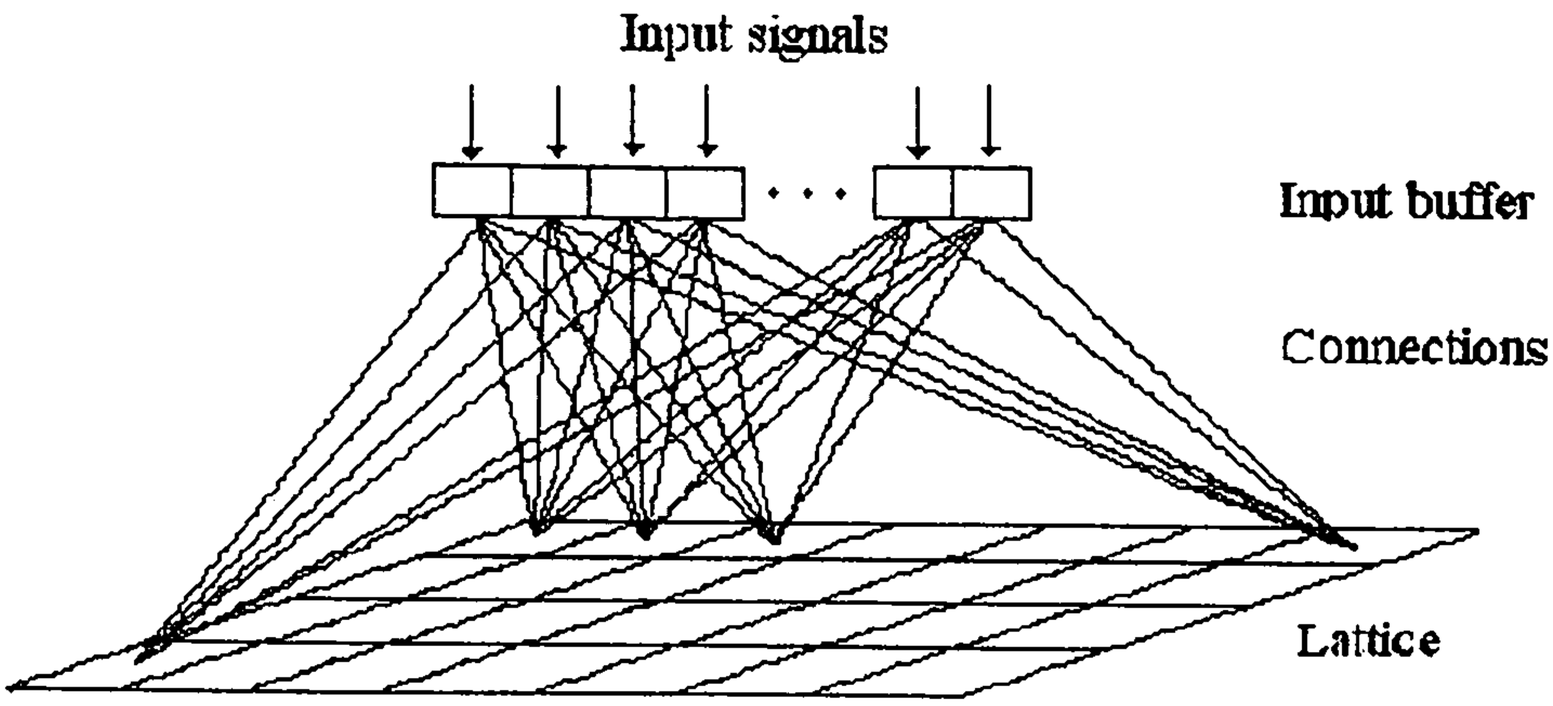


Figure 47: Physical structure of a self-organizing map

- Input buffer: The input buffer is the interface between the SOM and an environment from which signals come. It consists of slots into which components of an input signal are loaded, and the number of slots is determined by the number of signal components. To see what is meant by 'signal components', assume that the inputs are generated by a range of sensors --say 6, as in figure 47-- that monitor some industrial process. To check on the state of the process at any time *t*, the outputs from all the sensors at *t* are recorded simultaneously, yielding a signal with 6 components. That signal is loaded into the buffer such that the first signal component goes into the first slot, the second component into the second slot, and so on. Clearly, the number of signal components and corresponding buffer size can be anything from 1 to some arbitrary whole number *k*, depending on the application.

- Connections: The connections transmit the signals from the buffer to the lattice, and can be thought of as electrical wires. There is a connection from each buffer slot to all the processing units in the lattice, and each unit consequently

receives signals from all the buffer slots; note that, for clarity, only a small number of connections is shown in figure 47.

- Lattice: Each unit in the lattice is activated by the signals transmitted to it via its connections. In the simplest case the activation is additive, that is, the activation of a given unit *u* is the sum of all the signals arriving on the connections converging on *u*, but various transformations of the summed signals are also possible. The lattice is here shown as a two-dimensional, square with square units, though other arrangements can be and are used --hexagonal units are, for example.

To see how a SOM is used for cluster analysis, assume a data set V consisting of *n* length-*k* inputs. Then cluster analysis of V is achieved by loading the *n* data items $v_i \in V$ successively into the input buffer. For each v_i, the values in the buffer are propagated through all the connections to the units in the lattice. Because of the variation in connection strength, a given v_i activates one unit more strongly than any of the others, thereby associating each v_i with a specific unit in the lattice. When all the v_i have been projected in this way, the result is a pattern of activation across the lattice. This pattern is the projection of the *n*-dimensional data into two-dimensional space, and the data's cluster structure can be seen in the lattice configuration.

A crucial feature of the SOM is that the connections can vary in the strength with which they transmit signals. Indeed, for a SOM to be of much use in data analysis, there must be a significant variation in strength across the connections. To see why, consider what happens when all the connection strengths are the same. Each unit is connected to all the slots in the input buffer, and, since all the connections are identical, each unit receives the same signal and is thus activated to the same degree as all the others for some given input. In other

words, the entire lattice assumes a uniform activation --for different inputs this uniform activation may be lesser or greater depending on the magnitude of the signal, but there can never be any variation in activation among the units in the lattice. Now assume that the connection strengths vary. In this case, each unit can in principle have a unique pattern of connectivity to the buffer, and a given input will thus propagate to different units with different intensities, generating a non-uniform pattern of activation across the lattice.

There may well be some application for the uniform-activation behaviour, but the one with which we are concerned here is not one of them: the aim is to project high-dimensional data into the low-dimensional --here two-dimensional-- lattice, and this clearly requires variation in activation patterning across the lattice. The question, therefore, is this: for some collection of input signals, what pattern of connection strength variation should there be in order for the SOM to project a representation of the high-dimensional signals onto the two-dimensional lattice? The answer is that the requisite variation is not explicitly specified, but rather learned from the characteristics of the input signals. Explanation of this learning in terms of the physical model would be cumbersome, and is better done in terms of the mathematical SOM model.

ii. *Mathematical SOM model*

Mathematical modelling of a physical system allows one to abstract away from the myriad details of its physicality and to concentrate on its essentials within a well-defined conceptual framework and associated formalism. For this reason, SOMs are typically discussed and used in terms of mathematical rather than physical models. Specifically, the physical SOM shown in figure 47, and indeed any other SOM, can be modelled using linear algebra.

- Input buffer: Corresponding to the input buffer of the physical model is a vector v whose length equals the number of slots in the buffer: a buffer with k slots appears in the mathematical model as a vector $v = [v_1, v_2...v_k]$, where k is a positive integer, and each of the vector elements v_i contains a number that represents the physical signal component in the corresponding buffer slot. Assume, for example, a buffer with 6 slots that accepts inputs from 6 sensors, as in the physical model. In figure 48, the variations in physical signal intensity are represented in grey-scale, where greater darkness stands for greater intensity, and the corresponding numerical representation of each component is a real-valued number in the range 0..1:

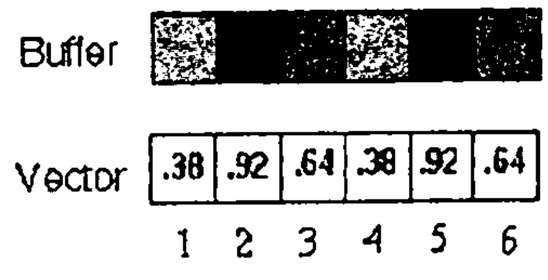


Figure 48: SOM input buffer

- Lattice: Corresponding to the lattice of the physical SOM is a 2-dimensional matrix M whose row and column dimensions are the same as those of the lattice, and whose elements contain numbers that represent physical activation. Figure 49 shows a 4 x 4 lattice with an activation pattern represented in grey-scale, as above, together with the corresponding matrix in which degrees of activation are represented by real-valued numbers:

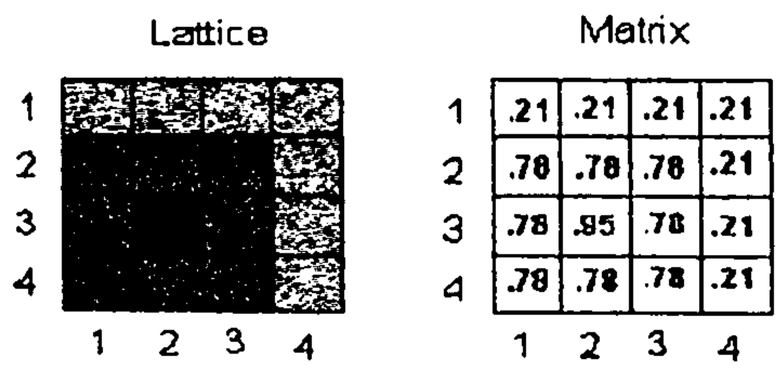


Figure 49: SOM lattice

- Input buffer: Corresponding to the input buffer of the physical model is a vector v whose length equals the number of slots in the buffer: a buffer with k slots appears in the mathematical model as a vector $v = [v_1, v_2 ... v_k]$, where k is a positive integer, and each of the vector elements v_i contains a number that represents the physical signal component in the corresponding buffer slot. Assume, for example, a buffer with 6 slots that accepts inputs from 6 sensors, as in the physical model. In figure 48, the variations in physical signal intensity are represented in grey-scale, where greater darkness stands for greater intensity, and the corresponding numerical representation of each component is a real-valued number in the range 0..1:

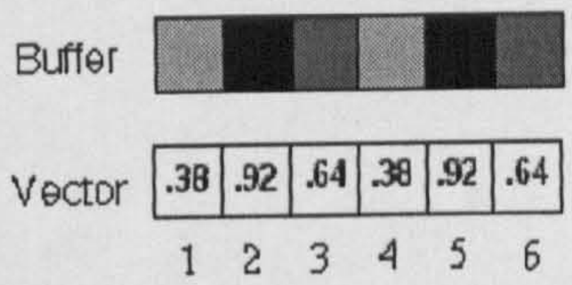


Figure 48: SOM input buffer

- Lattice: Corresponding to the lattice of the physical SOM is a 2-dimensional matrix M whose row and column dimensions are the same as those of the lattice, and whose elements contain numbers that represent physical activation. Figure 49 shows a 4 x 4 lattice with an activation pattern represented in grey-scale, as above, together with the corresponding matrix in which degrees of activation are represented by real-valued numbers:

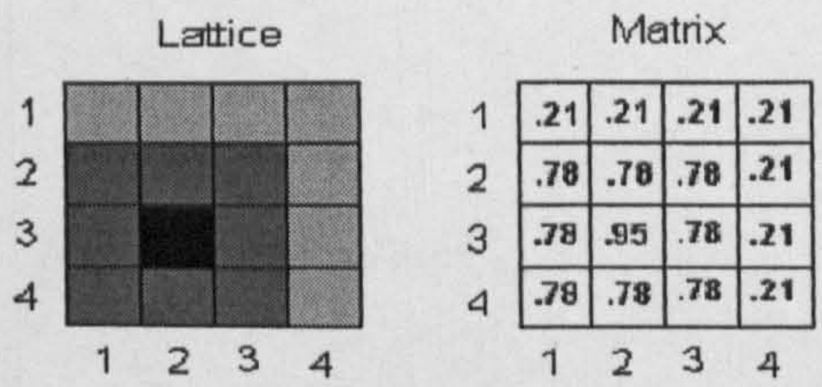


Figure 49: SOM lattice

Particular elements in M are indexed by row and column coordinates *i* and *j* with row 1 at the top and column 1 leftmost, as shown; the matrix element with the highest numerical value is indexed by $M_{3,2}$.

- Connections: Corresponding to the connections from the input buffer to the lattice in the physical SOM is a 3-dimensional matrix each of whose elements contain a number that represents the strength of a single connection.

To see how the connections are represented in such a matrix, we start with the observation that each of the units in the lattice of the physical SOM is connected to each slot in the input buffer, so that the number of connections *k* associated with each unit is the same as the size of the input buffer. The strengths of the connections associated with any given unit *u* can be represented as a length-*k*, real valued vector in which the number in the first element represents the strength of the connection between *u* and the first input buffer slot, the number in the second element represents the strength of the connection between *u* and the second input buffer slot, and so on. For $k = 6$, as before, the connection vector looks like this; the numerical values are arbitrary, for illustration only:

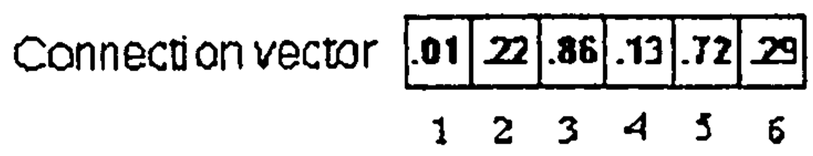


Figure 50: SOM connection vector

Clearly, one such vector is required to represent the connections associated with each of the units in the lattice; for a 4 x 4 lattice, that means 16 vectors. As the size of the lattice grows, however, as it does in practical applications, managing these vectors becomes an issue --a 100 x 100 lattice requires 10,000 vectors, for example.

One solution is to insert the connection vectors into a 2-dimensional matrix C whose row-column coordinates are identical to those of the lattice matrix M, such that the connection vector for $M_{i,j}$ is at $C_{i,j}$. This arrangement can be visualized by standing the connection vectors up vertically on C; the one in Figure 51, for example, could represent the 6 connections from unit $M_{1,1}$ to the buffer:

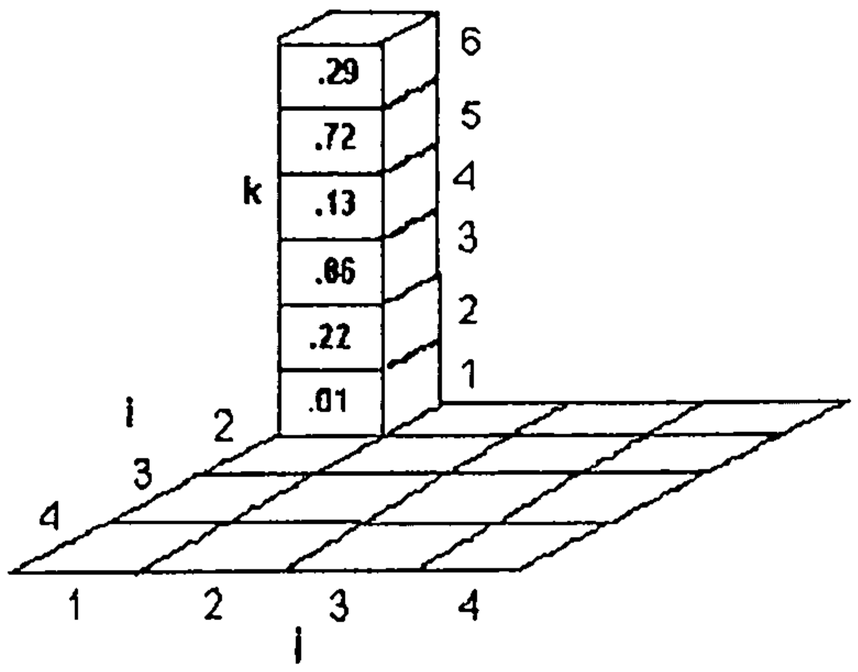


Figure 51: Matrix containing one SOM connection vector

And so on for remaining units from $M_{1,2}$ to $M_{4,4}$. The complete 3-dimensional matrix for a SOM having a 4 x 4 lattice and a 6-slot input buffer can therefore be visualized like this, with a vertical connection vector for each unit; values in the connection vectors are not shown to avoid overburdening the diagram, though each contains 6 numbers as in figure 51:

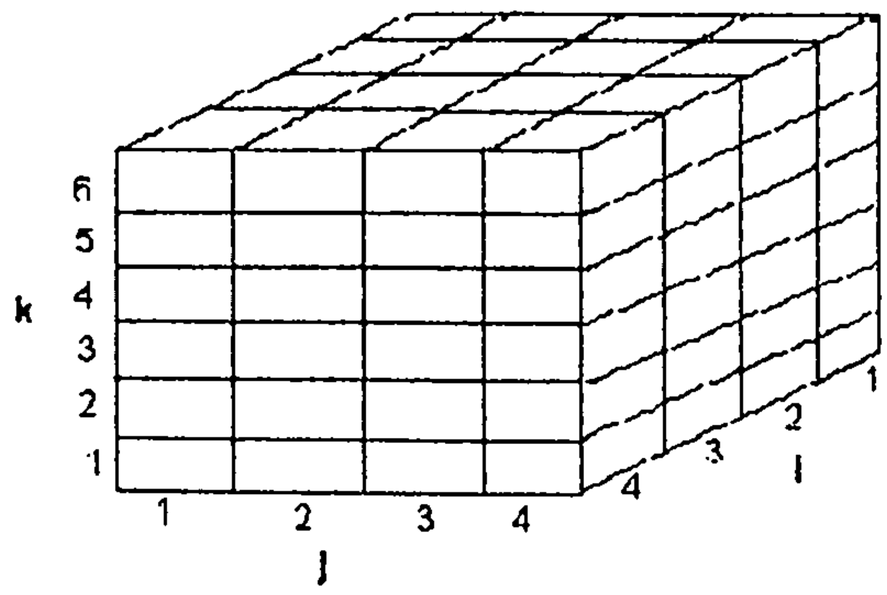


Figure 52: Matrix containing 16 SOM connection vectors

A connection vector is indexed as $C_{i,j}$, and a particular value in that connection vector as $C_{i,j,k}$.

- Physical behaviour: Corresponding to the behaviour of the physical SOM are algorithms that specify sequences of mathematical operations involving a set of input vectors V and the unit and connection matrices M, and C. Two such algorithms are required: one to represent the propagation of physical input signals from buffer to lattice together with the consequent lattice activation, and the other to represent the mechanism for learning connection strengths from characteristics of the inputs.

Signal propagation

The physical activation of the lattice in response to an input signal is mathematically modelled by taking the inner product of an input vector $v \in V$ and a connection vector $C_{i,j}$ --the inner product operation can be applied because the length of the connection vectors in any SOM is defined to be the same as the length of the input vectors. Thus

$$M_{i,j} = \Sigma_{1..k} (v_k \, . \, C_{i,j,k})$$

where i and j are the dimensions of the matrix M representing the physical lattice, k is the length of the vector representing the input signal, and the dot designates the inner or 'dot' product. The result of each inner product is stored at $M_{i,j}$, and the result is a pattern of numerical values in M that represents the physical activation of the lattice in response to input.

Connection learning

In terms of the physical model, SOM learning tries to find a pattern of connection strength variation such that high-dimensional input signals can be projected onto a low-dimensional lattice in a way that preserves any cluster structure that might be present in the set of input signals. In terms of the mathematical model, this can be restated as the attempt to find a set of connection vectors C such that the inner products of the $c \in C$ and the set of input vectors $v \in V$ generates in the lattice matrix M a pattern of activation that represents any cluster structure in V with minimum distortion.

Given a set of input vectors V, SOM learning is a dynamic process that proceeds in discrete time steps $t_1, t_2 \ldots t_p$, where p is a whole number that designates the number of time steps required to learn the desired mapping from high-dimensional vector space to the two-dimensional matrix M. At each time step $t_1, t_2, t_3 \ldots$ a vector $v_i \in V$ is selected, usually randomly, as input to the SOM, and the connection matrix C is modified in a way that is sensitive to the pattern of numerical values in v_i. At the start of the learning process the magnitude of modifications to C is typically quite large, but as the SOM learns via the modification process the magnitude decreases and ultimately approaches 0, at which point the learning process is stopped --the p'th time, as above. The question, of course, is how exactly the input-sensitive connection strength modification works, and the answer involves looking at the learning algorithm in detail.

At the start of learning, the SOM is parameterized with user-specified values:

1. Dimensionality of input vectors

2. Lattice axis lengths: these can be equal, giving a square lattice, or unequal, giving a rectangular one.

3. Unit shape: the 'shape' of a unit is defined by how many other units adjoin it in the lattice. The usual choices are square or hexagonal.

4. Initial neighborhood and neighborhood decrement interval; this is explained below.

5. Initial learning rate and learning rate decrement interval, which is also explained below.

In addition, the values in the connection matrix C are initialized in such a way that they are non-uniform; uniform connections would preclude the possibility of learning, for reasons that are implicit in the foregoing discussion.

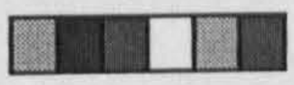
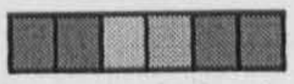
Thereafter, at each time step $t_1, t_2, t_3 \ldots t_p$

1	An input vector $v \in V$ representing an input signal to the physical SOM is selected.
2	The propagation of the input signal through the connections to the m x n lattice in the physical SOM is represented as the inner product of v and the connection vector C_{ij} for each unit of the lattice, where i is in the range $1..m$ and j in the range $1..n$. The result of each inner product is stored in the corresponding cell of the lattice matrix M_{ij}. Once all the inner products have been calculated, because the connections strengths in C were initialized to be non-uniform, the result is a varying pattern of activation across the matrix.

3	The lattice matrix M is now searched to identify the unit with the largest numerical activation value. We shall call this unit u_{ij}, where i and j are its coordinates in the lattice matrix.
4	The connections strengths in C are now updated. This is the crucial step, since it is how the SOM learns the connections required to carry out the desired mapping. This update proceeds in two steps: a) Update of the connections linking the most highly activated unit in the lattice matrix M to the input buffer. This is done by changing the connection vector associated with the most active unit u_{ij} according to the following formula: $$C_{ij}(t+1) = C_{ij}(t) + \alpha(t)(v - C_{ij})$$ where • t denotes the current time step • $C_{ij}(t)$ is the current state of the connection vector associated with u_{ij} • C_{ij} $(t+1)$ is the updated state of the connection vector associated with u_{ij} • $\alpha(t)$ is the learning rate at time t, a parameter which controls the size of the modification to the connection vector. More is said about this in due course. • $(v - C_{ij})$ is the difference between the current input vector and the connection vector. This difference is the sum of element-wise subtraction: $\Sigma_{1..k}(v_k - C_{i,j,k})$, where k is the length of the input and

connection vectors.

In words, the update to the connection vector C_{ij} associated with the most active unit u_{ij} is the current value of C_{ij} plus some proportion of the difference between C_{ij} and the input vector, as determined by the learning rate parameter. The effect of this is to make the connection vector increasingly similar to the input vector. For example, for some v and associated C_{ij}:

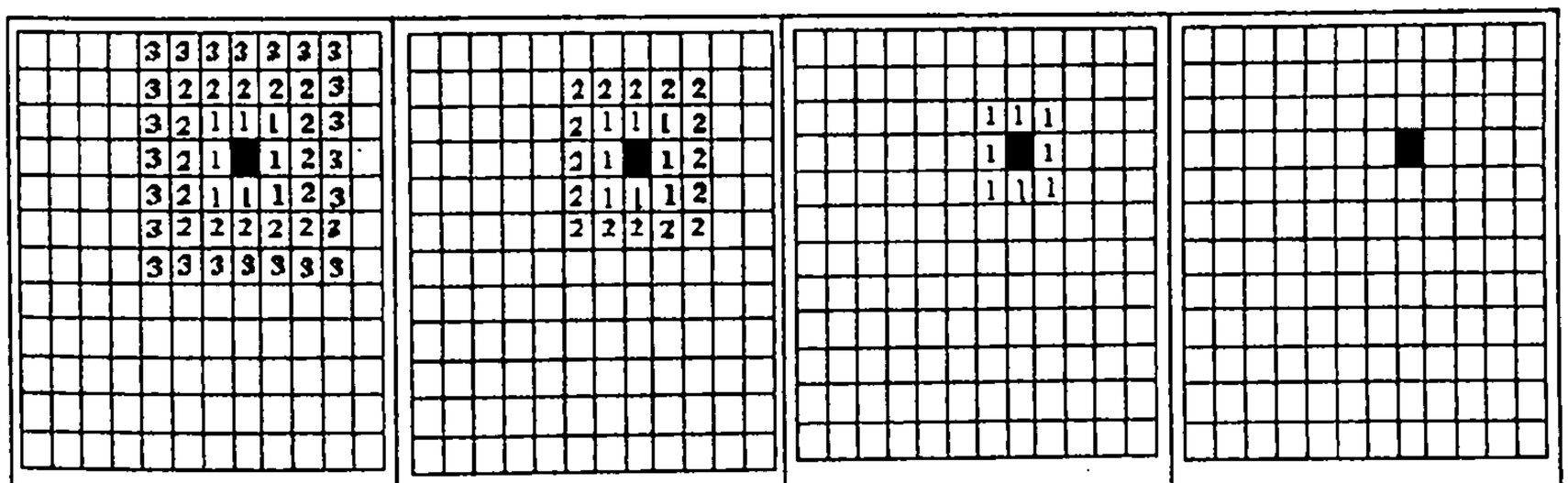
Before update	Input vector	
	Connection vector	
After update	Input vector	
	Connection vector	

Before update the input and connection vectors differ significantly; to make this clearer the difference is also shown in grey-scale. After update, the connection vector has moved closer to the input vector. This update in turn means that, the next time the particular input vector *v* is loaded, the inner product of *v* and C_{ij} and that the activation of M_{ij} will be even greater. In this way, *v* is associated ever more strongly with a particular unit on the lattice.

b) It is not only the connections associated with the most-activated unit u_{ij} that are modified. Those associated with units in the neighborhood of u_{ij} are also

modified in the way just described:

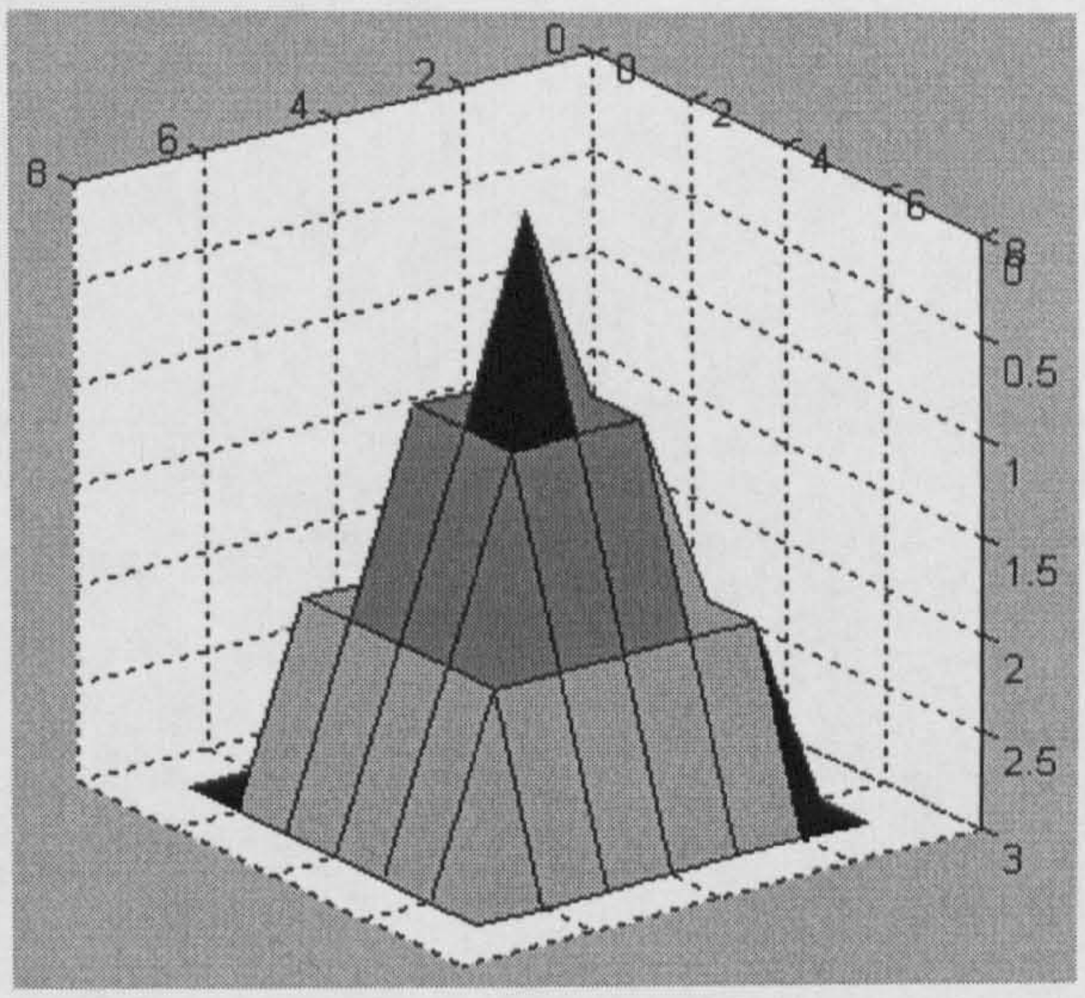
- The neighborhood of u_{ij} is the units that fall within some spatial adjacency distance 1, 2, 3... of it. For example, the following figure depicts distances 3, 2, 1, and 0 from u_{ij} on the SOM lattice, where u_{ij} is shown as a black square:



Neighborhood 3	Neighborhood 2	Neighborhood 1	Neighborhood 0

In SOM training, the neighborhood is initialized to some large value and is decreased as training proceeds. The choice of initial neighborhood size and the rate of decrease is user-specified; more is said about this in due course.

- The degree of connection modification within a neighborhood is not uniform, but decreases as a function of the distance of any given unit in the neighborhood from u_{ij}. This function is usually some discrete approximation to the Gaussian distribution; using such a function, neighborhood 3 can be conceptualized like this:



and the connection vectors for all the units within a neighborhood distance *n* can be updated according to a function like

$$c_{ij}(t+1) = c_{ij}(t) + \frac{\alpha(t)(v - c_{ij}(t))}{(n+1)}$$

where

- c_{ij} is the connection vector associated with the map coordinates i and j
- t is the current time step
- α is the current learning rate parameter
- v is the current input vector
- n is the size of the current neighbourhood.

Using this function, the connections for each of the 8 units at distance 1 from u_{ij} are modified only half as much as those of u_{ij}, the connections for the 16 artificial neurons at distance one third as much,

and so on outwards from u_{ij}. The effect of this is to associate v not only with activation of a specific unit in the lattice, but with a penumbra of activation. As the neighbourhood connections are made to increasingly converge with v, so not only a single unit but a whole region of the lattice is associated with v. Seen physically in terms of grey-scales, the lattice response to v increasingly looks like this:

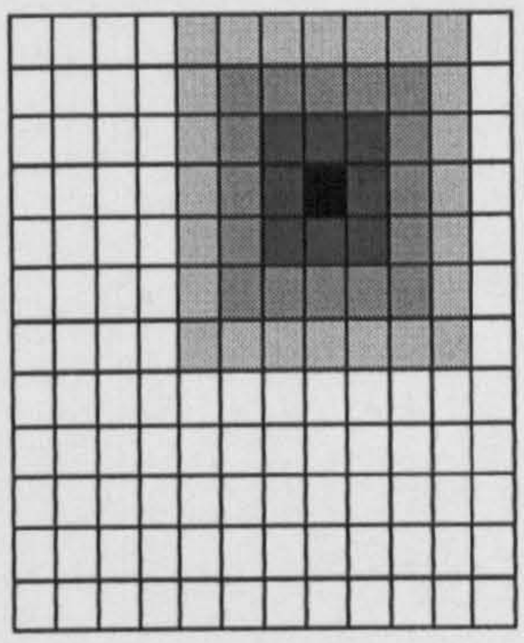


Figure 53: SOM training algorithm

b) SOM issues

The foregoing account of SOM architecture described what Kohonen calls 'the original incremental SOM algorithm' (Kohonen 2001, 109). Numerous optimizations have been developed since Kohonen first proposed it (see for example Kohonen 2001, 148-52 & ch. 5), but the original algorithm is used in the analyses that follow both because it has a good track record of satisfactory results in the literature, and to avoid straying too far from the main thrust of the discussion by engagement with the various issues that these optimizations raise. There are, however, some issues that do need to be discussed with respect the 'standard' SOM.

i. *Output interpretation*

When a SOM is used for cluster analysis, inspection of the pattern of activation on the lattice can be misleading. There is a strong temptation to interpret the pattern spatially, that is, to interpret any groups of adjacent, highly activated units as clusters, and the distance between and among clusters on the lattice as proportional to the relative distances among data items in the high-dimensional input space. That temptation needs to be resisted. The SOM maps a manifold in a relatively high-dimensional space into a relatively low-dimensional one in a way that preserves the manifold's topology. To see the implications of this for cluster interpretation of the output lattice, some additional discussion of the SOM learning algorithm is required.

We have seen that each lattice unit has an associated vector which represents its connections to the input buffer. Since the dimensionality of the connection vectors is the same as that of the input buffer, and the dimensionality of the input buffer is the same as that of whatever *n*-dimensional input space is currently of interest, the connection vectors are, in fact, coordinates of points in that *n*-dimensional space. Assume that there is a data manifold in the input space and that the connection vectors have been randomly initialized. In this initial state, there is no systematic relationship between the points specified by the set of connection vectors and the surface of the manifold. By incrementally bringing the connection vectors closer to training vectors taken from the data manifold, a systematic relationship is established in the sense that the connection vectors come to specify points on the manifold; at the end of training, the connection vectors map each of the points on the manifold specified by the training vectors to a particular lattice unit. Moreover, it can and usually does happen that data vectors which are close together on the manifold activate the same unit u_{ij}, as described earlier. In this case, the connection vector for u_{ij} has to be

brought closer not only to one but to some number k of input vectors. Since these k vectors are close but not identical, the SOM algorithm adjusts the connection vector of u_{ij} so that it becomes a kind of 'average' vector that specifies a point on the manifold which is in some sense intermediate between the k input vectors. In this way, u_{ij} becomes associated not only with a single point on the data manifold, but with an area of the manifold surface containing the k vectors that map to it. This is shown in Figure 54 below:

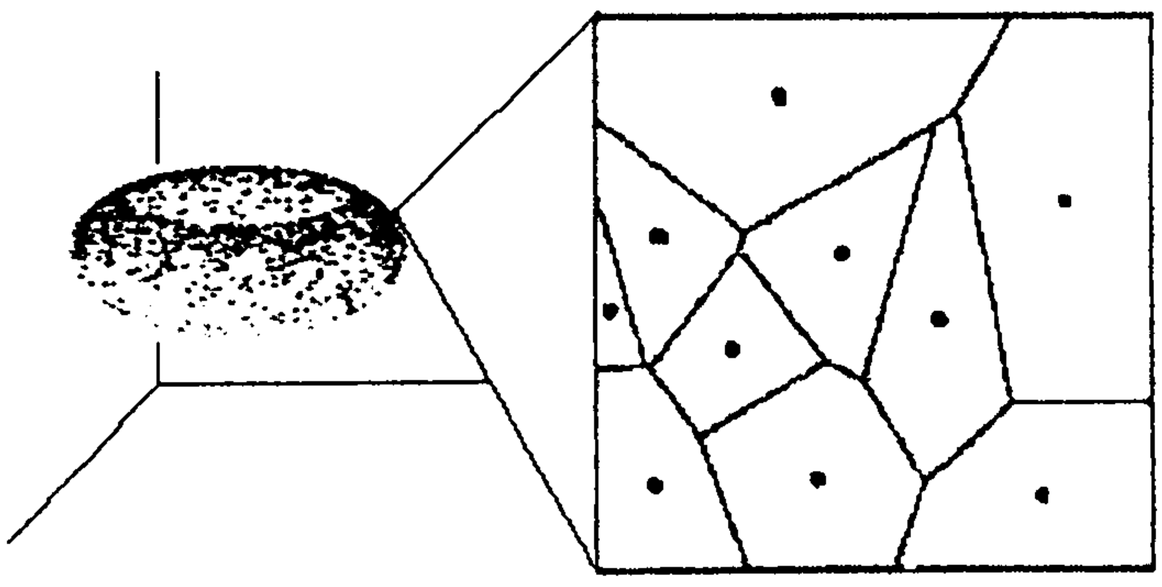


Figure 54: Voronoi tesselation of a manifold surface

The doughnut shape on the left is a manifold in 3-dimensional space, and the square on the right is intended to represent the way in which a SOM trained on this manifold partitions its surface: each dot represents an 'average' vector in the above sense associated with a specific lattice unit u_{ij}, and the lines enclosing a dot represent the boundaries of the area of the manifold containing the k vectors that map to u_{ij}. Mathematically:

- The process of finding an 'average' vector that is intermediate between k vectors is known as vector quantization: for a set V of k vectors, find a 'reference' vector v_r of the same dimension as those in V such that the absolute difference $d = |v_r - v_i|$ between each $v_i \in V$ and v_r is minimized. The SOM training algorithm quantizes the input vectors, and the connection vectors are the result (Kohonen 2001, 59-60; Ritter *et al.* 1992, ch. 14; de Bodt *et al.* 1997; Vesanto 2002).

- The partition of a manifold surface into areas surrounding reference vectors is a tesselation. The SOM algorithm implements particular type, the Voronoi tesselation, in which a reference vector is known as a centroid, and the area of k associated vectors surrounding a centroid as a neighbourhood; the neighbourhood of a given reference vector v_r in a Voronoi tesselation is defined as the set of vectors closer to v_r than to any other reference vector. (Verleysen 1997; Kohonen 2001, 59-60).
- The set of neighbourhoods defined by the Voronoi tesselation is known as the manifold's topology (Mendelson 1975; Munkres 2000).

And because, finally, the SOM algorithm adjusts the connection vector not only of the most-activated unit u_{ij} but also of units in a neighbourhood of gradually diminishing radius, it ensures that adjacent manifold neighbourhoods map to adjacent lattice units.

How does all this relate to cluster interpretation of a SOM lattice? A Voronoi tesselation is an instance of a topology, that is, a manifold and a discrete collection of subsets of points on the manifold called neighbourhoods. When it is said that a SOM preserves the topology of the input space, what is meant is that it represents the neighbourhood structure of a manifold: when data is input to a trained SOM, the vectors in a given Voronoi neighbourhood are mapped to the same lattice unit, and the vectors in adjoining Voronoi neighbourhoods are mapped to adjacent lattice units. The result of this topology preservation is that all vectors close to one another in the input space in the sense that they are in the same or adjoining neighbourhoods will be close on the SOM output lattice. The problem is this: just because active units are close together on the SOM lattice does not necessarily mean that the vectors which map to them are topologically close in the input

space. This apparently-paradoxical situation arises for two reasons (see discussion in, for example, Ritter *et al.* 1992, ch. 4) :

- The topology of the output space into which the SOM maps a data manifold is fixed in advance: in the standard SOM, it is two-dimensional and has a rectangular or hexagonal neighbourhood that is uniform across the lattice except for at the edges, where the neighbourhoods are necessarily truncated. The representation on the input manifold on the lattice is, moreover, linear in that the lattice is a plane. There is no guarantee that the output topology will be able to represent the input perfectly or indeed very well for any given input manifold. In other words, there may be some degree of distortion in the lattice representation (Verleysen 1997).
- The dynamics of SOM training do not at any stage use global distance measures. The mapping from input to output space depends entirely on local neighbourhood adjacency. As such, the SOM cannot be expected to preserve proportionalities of distance between individual vectors and vector neighbourhoods, and in fact it does not do so.

As a result, the SOM may squeeze its representation of the input topology into the lattice in such a way that units associated with centroids that are far apart on the input manifold may nevertheless be spatially close to one another in the lattice.

How, therefore, can a SOM lattice be interpreted so as to differentiate units that are spatially close because they are topologically adjacent in the input space and therefore form a cluster, and units that are spatially close but topologically more or less distant in the input space? The answer is that it cannot be done reliably, or at least not by visual inspection. Consider the following SOM map of a data set that has been extensively used in clustering literature: the Iris database available online from numerous sources, i.e., currently

http://www.hakank.org/weka/iris.arff. It lists, for 150 plants, the sepal length, sepal width, petal length, and petal width; each plant is labelled according to its membership of one of three classes of Iris. A random selection of rows is given below to provide a flavour of the data:

Sepal length	**Sepal width**	**Petal length**	**Petal width**	**Variety**
5.1	3.0	1.4	0.2	Iris-setosa
4.9	3.2	1.3	0.2	Iris-setosa
		....		
4.7	3.1	1.5	0.2	Iris versicolour
4.6	3.6	1.4	0.4	Iris versicolour
		....		
5.0	3.9	1.7	0.3	Iris virginica

Table 21: Sample of Iris data

A SOM with a 40 x 40 unit lattice was trained on this data; details of the training parameters are not given because this is just an example, and is not otherwise relevant to the main discussion:

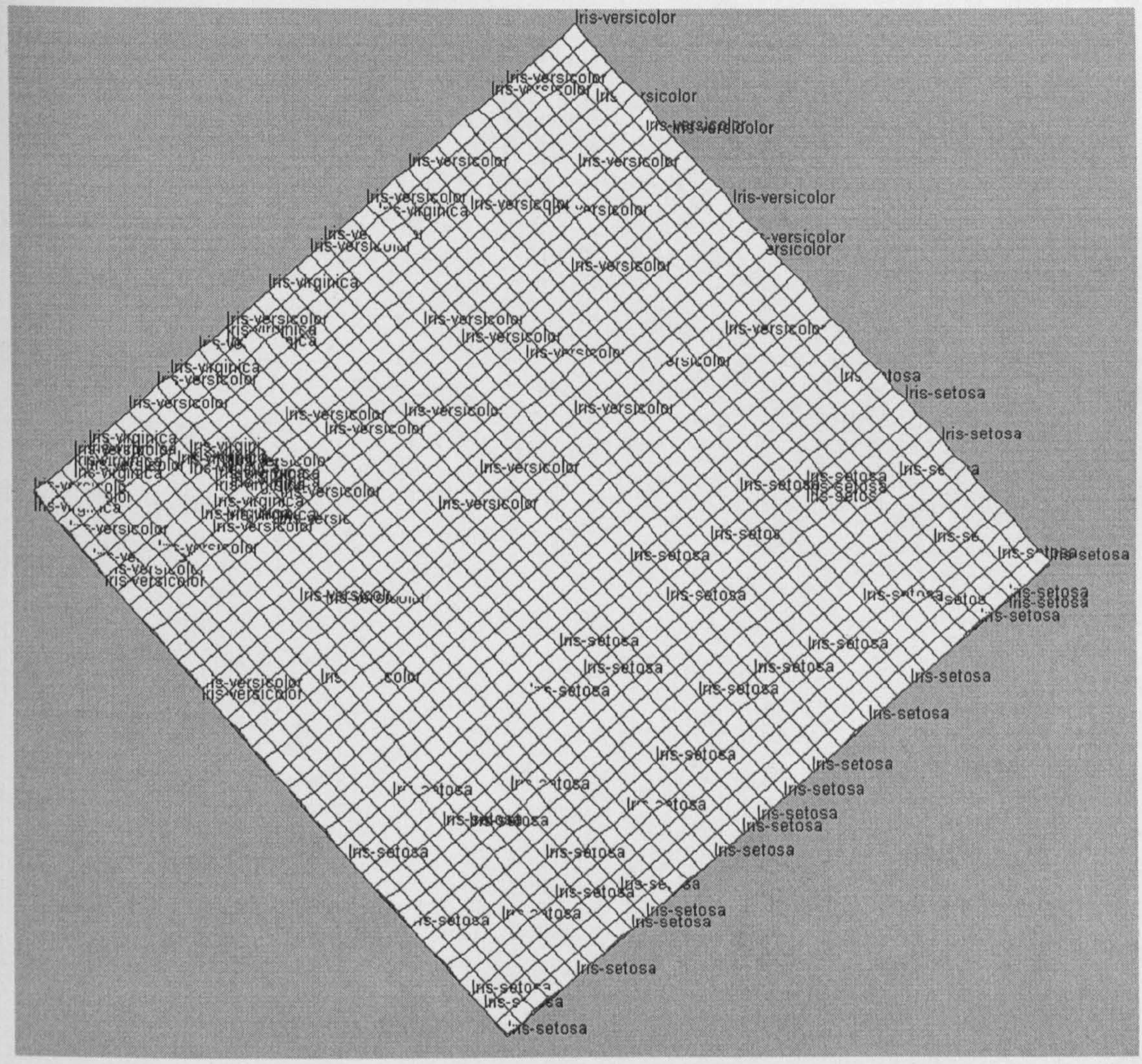


Figure 55: SOM map of Iris data

Where, if anywhere, are the clusters on this map? Here is one possibility:

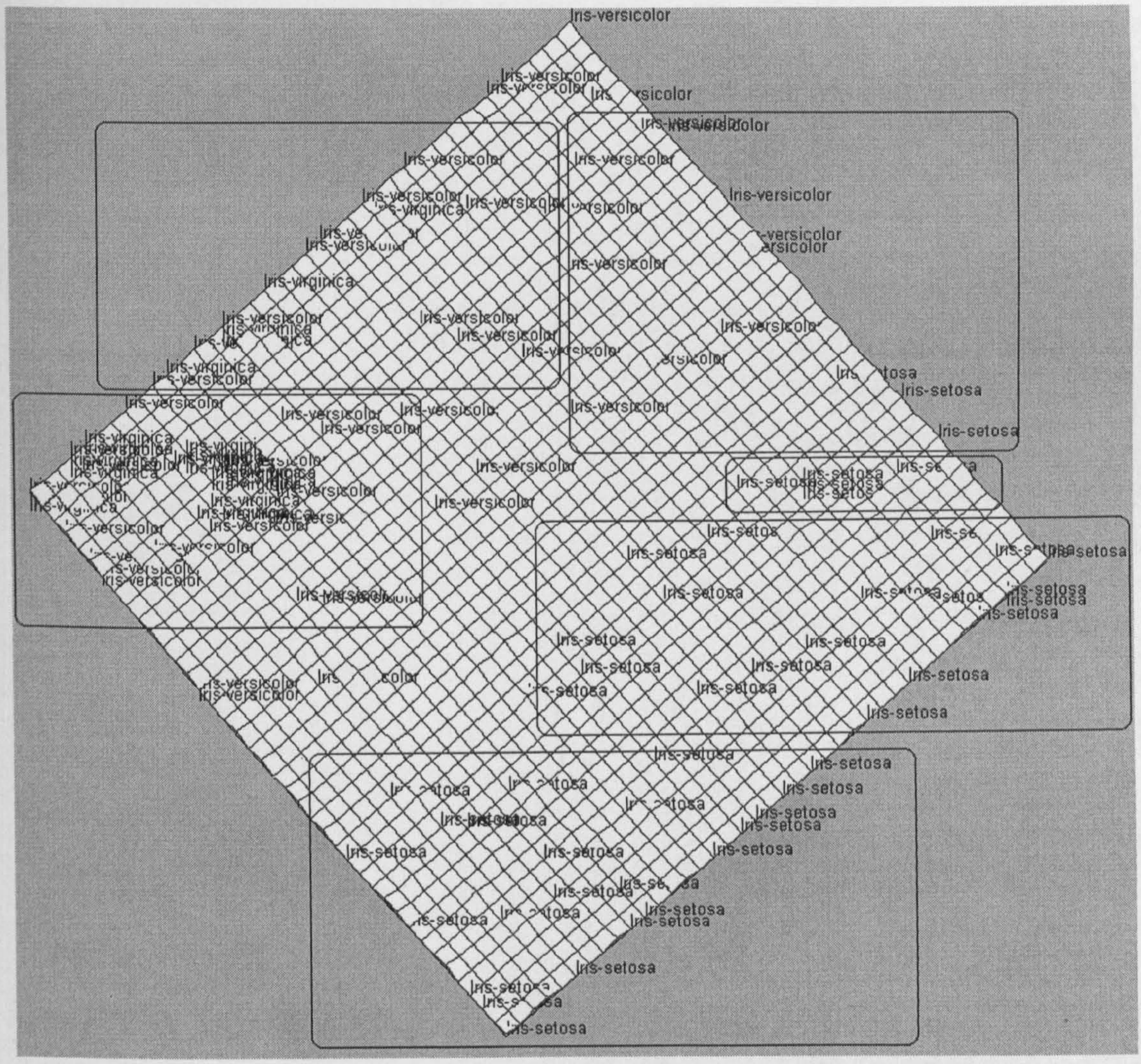


Figure 56: One possible partition of the Iris data map

And another:

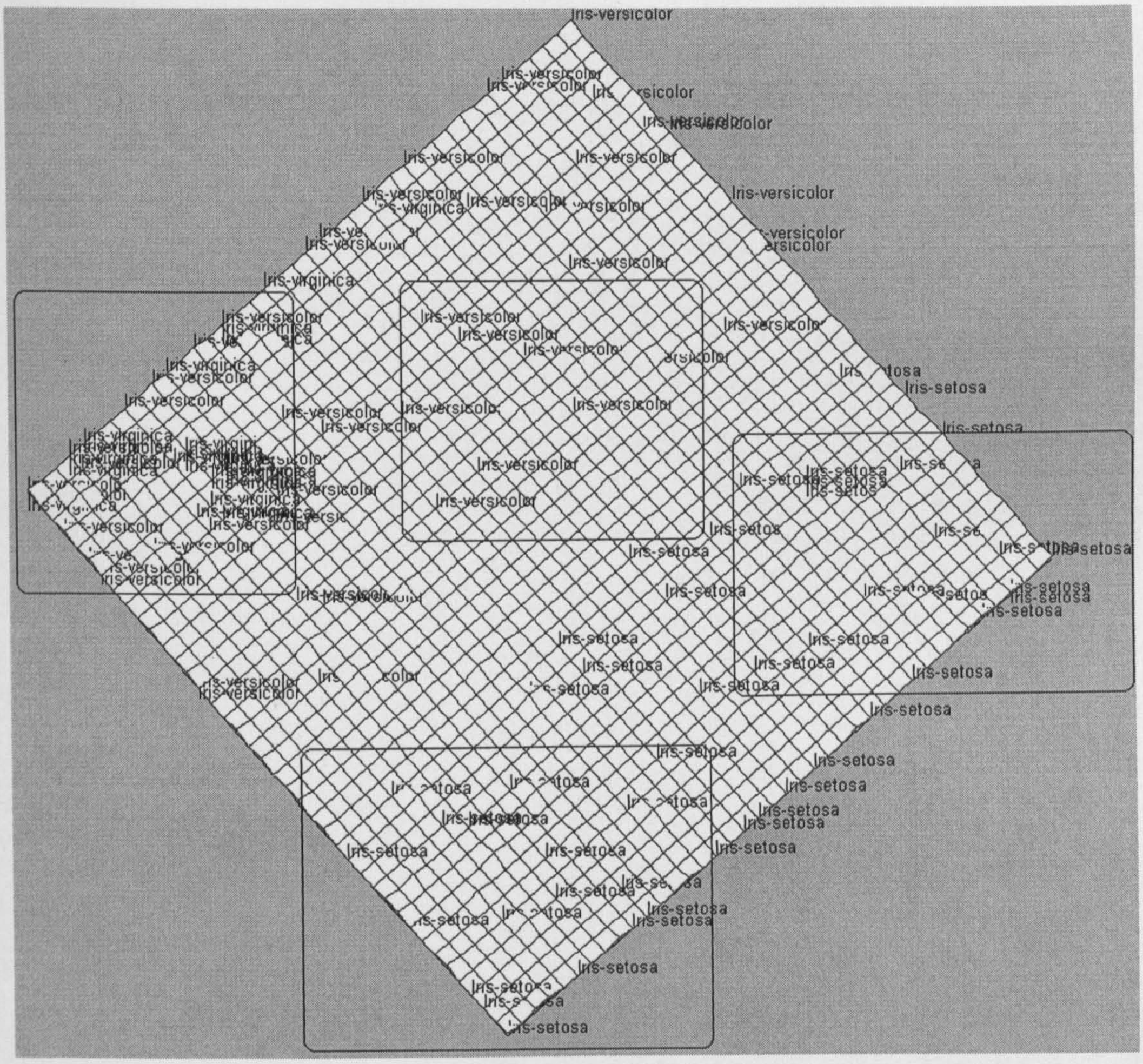


Figure 57: Another possible partition of the Iris data map

In fact, there are very numerous other possible partitions as well. The point is that the map gives no clear indication where any cluster boundaries might be. The eye picks out likely concentrations, but when asked to decide whether a given data item is or is not part of a visually-identified cluster, one is at a loss.

An apparent solution to this problem is to fall back on knowledge of the subject domain -- here flowers-- as a guide to partitioning of the map. But using a priori knowledge of the data to disambiguate the map is not a genuine solution, for two reasons:

- It misses the point of the exercise. The map is supposed to reveal the structure of the data, not the other way around.
- The structure of large, complex, real-world data sets to which the SOM is applied as an analytical tool is not usually recoverable from mere inspection –if it were, there would be little point to SOM analysis in the first place.

The conclusion, therefore, is that, to be useful as an analytical tool, the SOM's representation of data structure has to be unambiguously interpretable on its own merits, and the problem is that a lattice activation map like that in the above figures does not contain enough information to permit this in the general case.

This is a well known problem with SOMs, and a variety of ways of identifying cluster boundaries on the SOM lattice have been proposed (Kaski 1997; Kaski *et al.* 2000; Merkl 1997, 1998, 1999, 2000; Merkl & Rauber 1997a, 1997b, 2000; Pölzlbauer *et al.* 2005a, 2005b; Ultsch & Siemon 1990; Ultsch 1993; Vesanto 1999, 2000, 2002; Vesanto & Alhoniemi 2000). A very widely used one is the U-matrix (Ultsch & Siemon 1990; Ultsch 1993), which, because it is used in what follows, is described in some detail here.

The U-matrix representation of SOM output uses relative distance between connection vectors to find cluster boundaries. Specifically, given an m x n output map M, the Euclidean distances between the connection vector associated with each map unit M_{ij} (for I = 1..m, j = 1..n) and the connection vectors of the immediately adjacent units $M_{i-1,j}$, $M_{i+1,j}$, $M_{i,j-1}$, and $M_{i,j+1}$ are calculated and summed, and the result for each is stored in a new matrix U_{ij} having the same dimensions as M. If the set of units immediately adjacent to M_{ij} is designated as $M_{adjacent\ (I,J)}$, and d represents Euclidean distance, then

$$U_{ij} = \Sigma\, d\,(M_{adjacent\ (ij)}, M_{i,j})$$

U is now plotted using a colour coding scheme to represent the relative magnitudes of the values in U_{ij}, or a three dimensional representation in which low values are shown as low points on the graph and high values as high points, or a combination of the two. Any significant cluster boundaries will be visible. Why? The connection vectors are the coordinates of the centroids of the Voronoi □eighbourhoo of the data manifold and thus represent the manifold's topology, as we have seen. Where the sum of distances between the connection vector associated with M_{ij} and those associated with $M_{adjacent(ij)}$ is small, the distance between those centroids on the manifold is small; conversely, a large sum indicates a large distance between centroids on the manifold. Low-magnitude regions in U thus represent topologically close regions on the manifold, and high-magnitude ones topologically distant regions on the manifold. In a three-dimensional plot, therefore, the U-matrix appears as a landscape in which the low-magnitude regions are 'valleys' and the high-magnitude ones mountains; the valleys are the clusters, and the mountains are the boundaries separating them.

As a simple example of how a U-matrix representation of the SOM lattice output actually looks, consider the following trivial data set:

	v1	**v2**	**v3**	**v4**
Item 1	1	0	0	0
Item 2	0	1	0	0
Item 3	0	0	1	0
Item 4	0	0	0	1

Table 22: A data set

A SOM with a 12 x 12 lattice was trained on this data, with the following result:

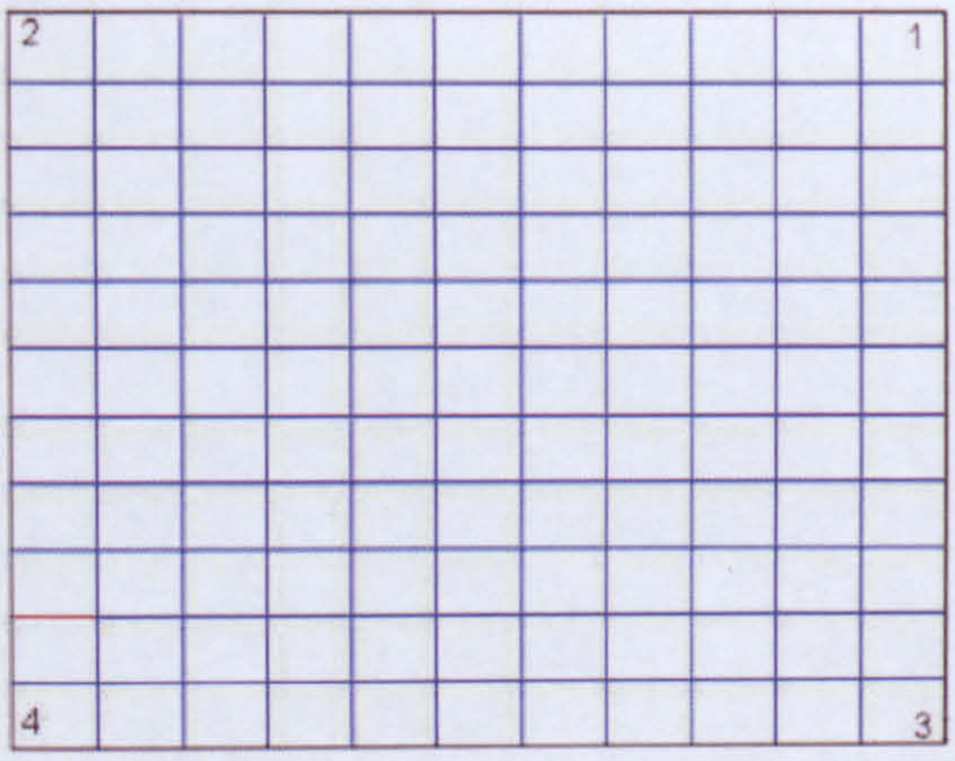


Figure 58: SOM map of data set in Table 22

The U-matrix representation using only colour coding to represent variation in magnitude looks like this:

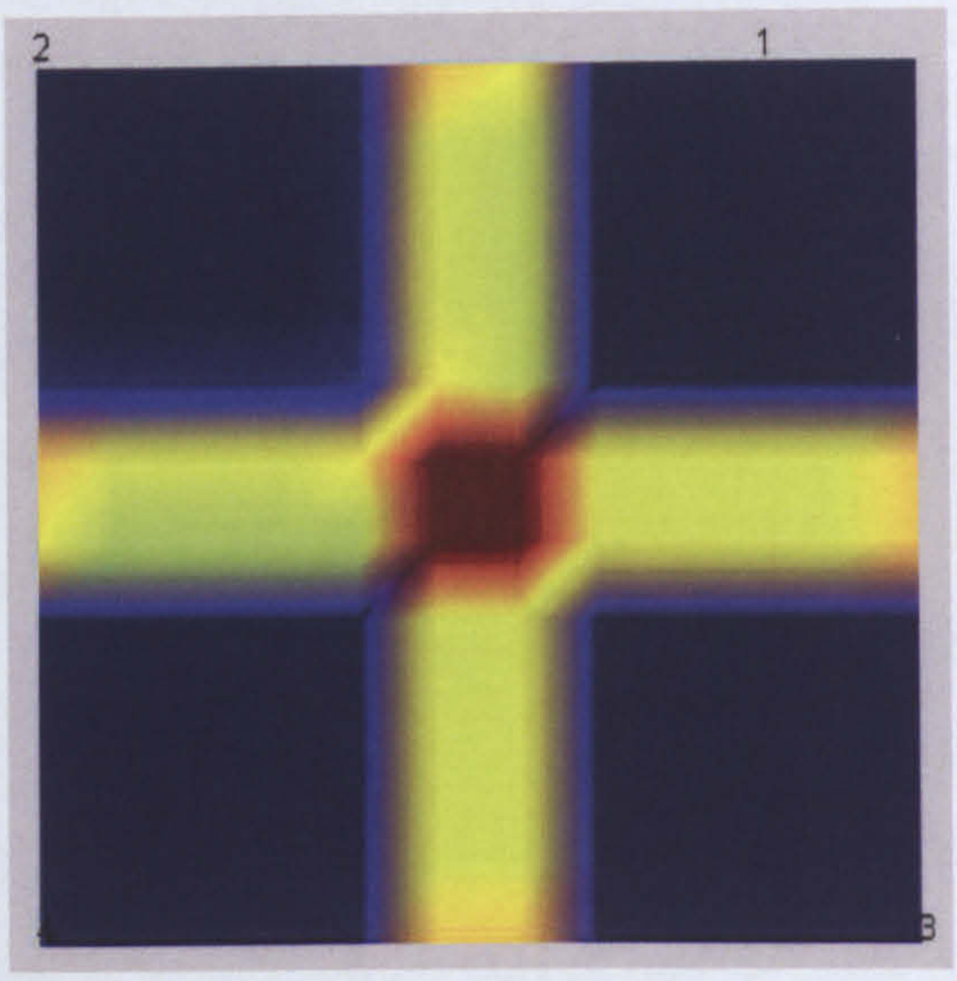


Figure 59: U-matrix representation of SOM map in Figure 58

The four blue areas are the clusters, and the yellow / red areas are the boundaries between clusters. If this colour-coded two dimensional plot is supplemented by a three-dimensional representation, the cluster landscape emerges:

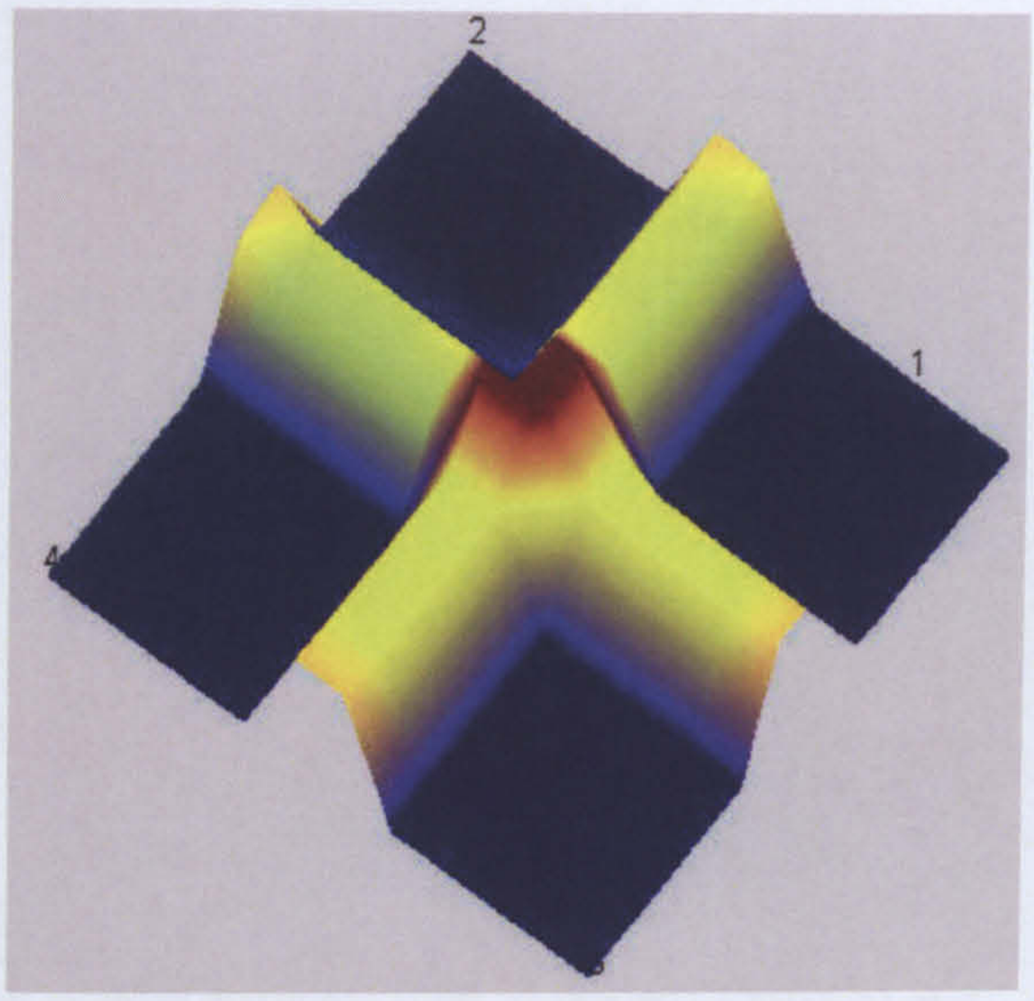


Figure 60: Three-dimensional representation of U-matrix in Figure 59

When the U-matrix representation is applied to the SOM output for the Iris data, the cluster structure is clearly evident, and it is nothing like the partitions suggested earlier:

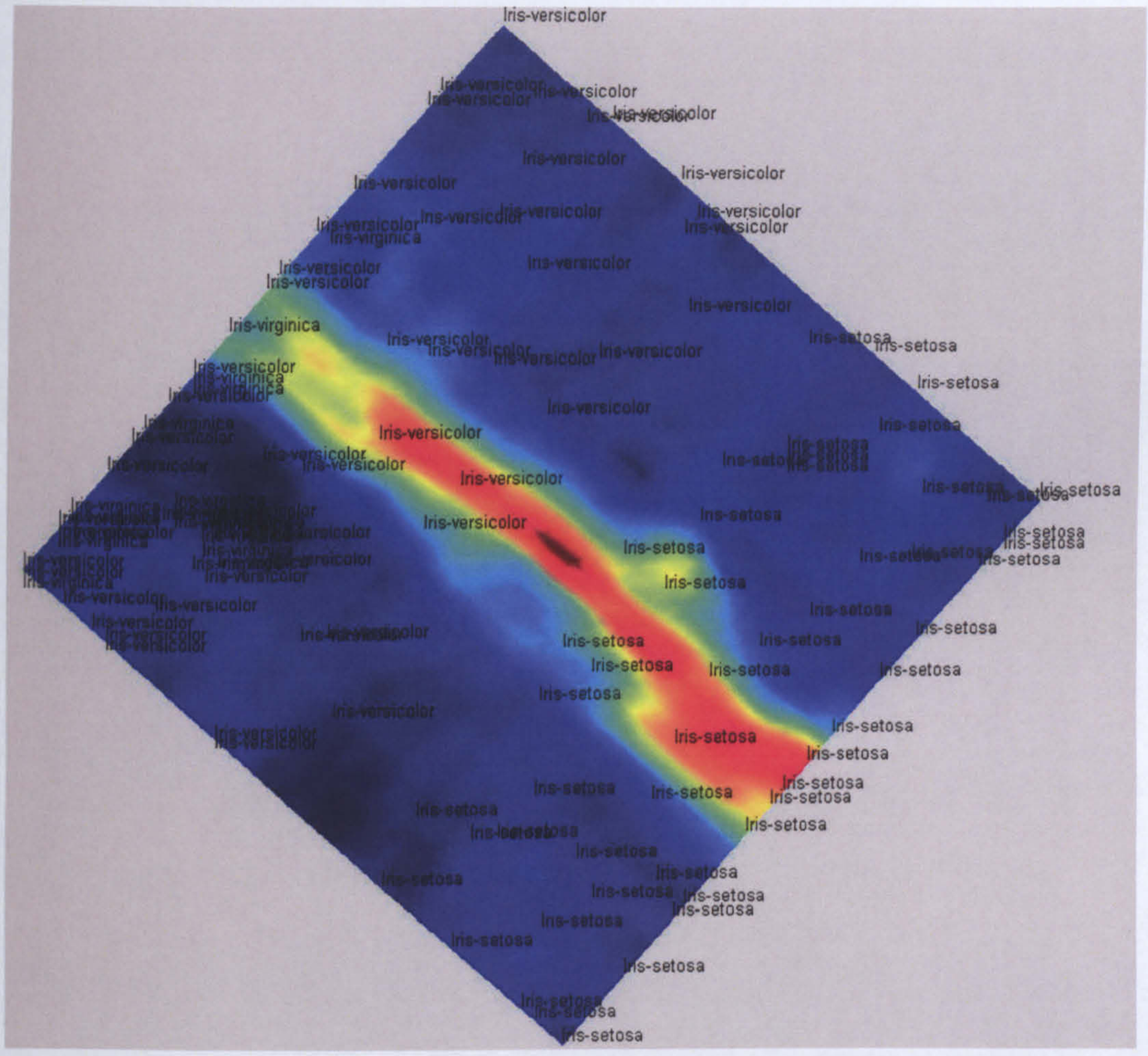


Figure 61: Two-dimensional U-matrix representation of Iris data

There are two main clusters separated by a clearly defined boundary: within each cluster the relative spatial distance between data items reflects the input manifold topology, but spatial distance between items on either side of the boundary does not. This can, again, be shown even more clearly by a rotated U-matrix activation landscape; the map has been inverted so that the mountain is now a valley and the valleys mountains because it makes the labels easier to see:

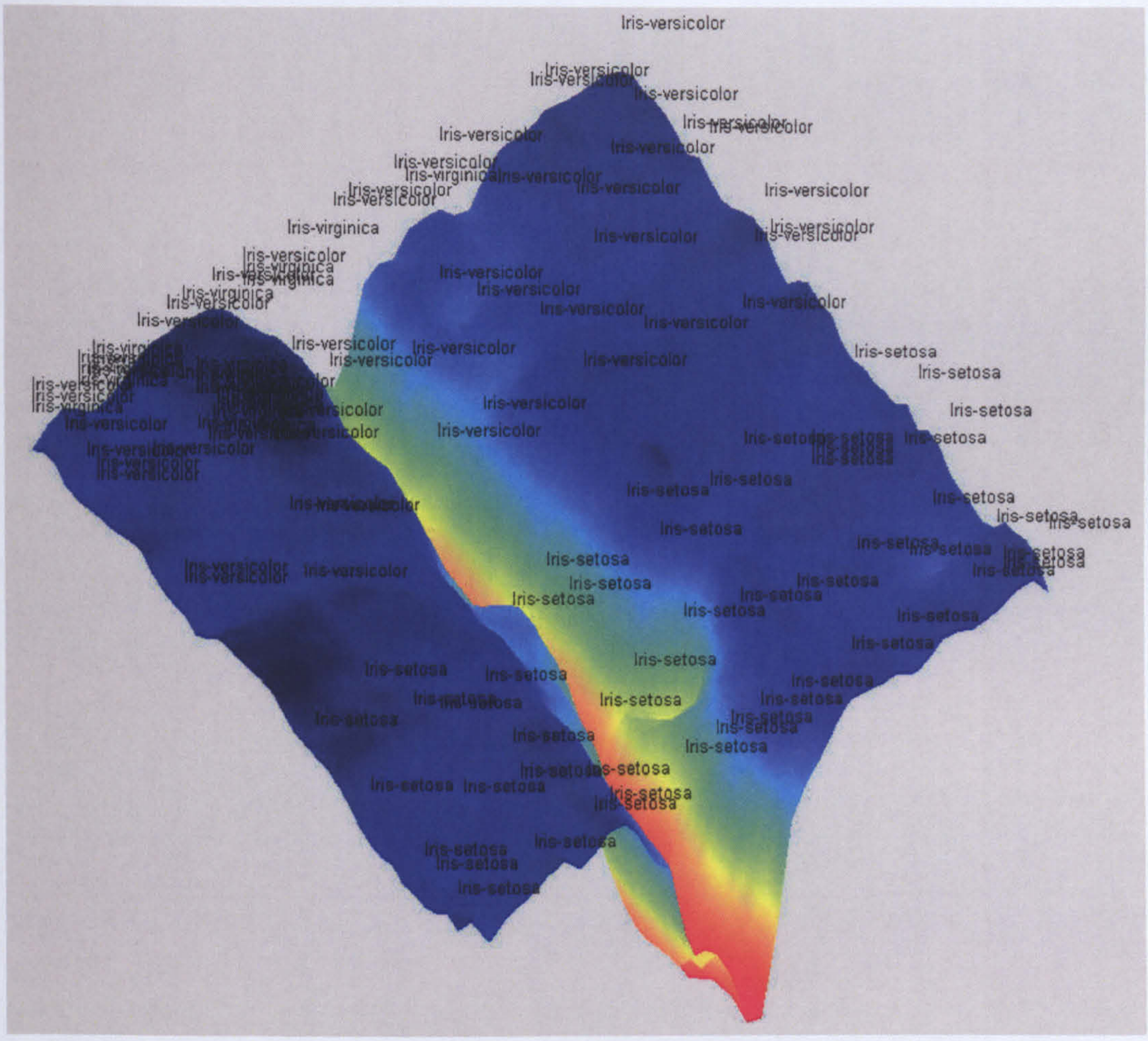


Figure 62: Three-dimensional U-matrix representation of Iris data

U-matrix visualization has been developed further into P-matrix and then U*-matrix (Ultsch 2003a; Ultsch & Mörchen 2005) but, because of its simplicity and clarity, and because it suffices for present purposes, the U-matrix representation of SOM output is used in the discussion that follows.

ii. *Nonlinearity*

The motivation for using the SOM in addition to hierarchical cluster analysis in this study is to cover the possibility that the Qur'an data contains significant nonlinearities. The SOM is one of a range of nonlinear methods, that is, methods which take data nonlinearities into

account in generating their cluster analyses. What does it mean to say that the SOM 'takes nonlinearities into account' (Ritter *et al.* 1992, ch. 14)?

In general, linear analytical methods construct clusters on the basis of measures of relative linear distance between and among points on the data manifold in Euclidean space. This works well for manifolds that are linear or near-linear, but can give poor results where the manifold is significantly nonlinear. Consider the following example:

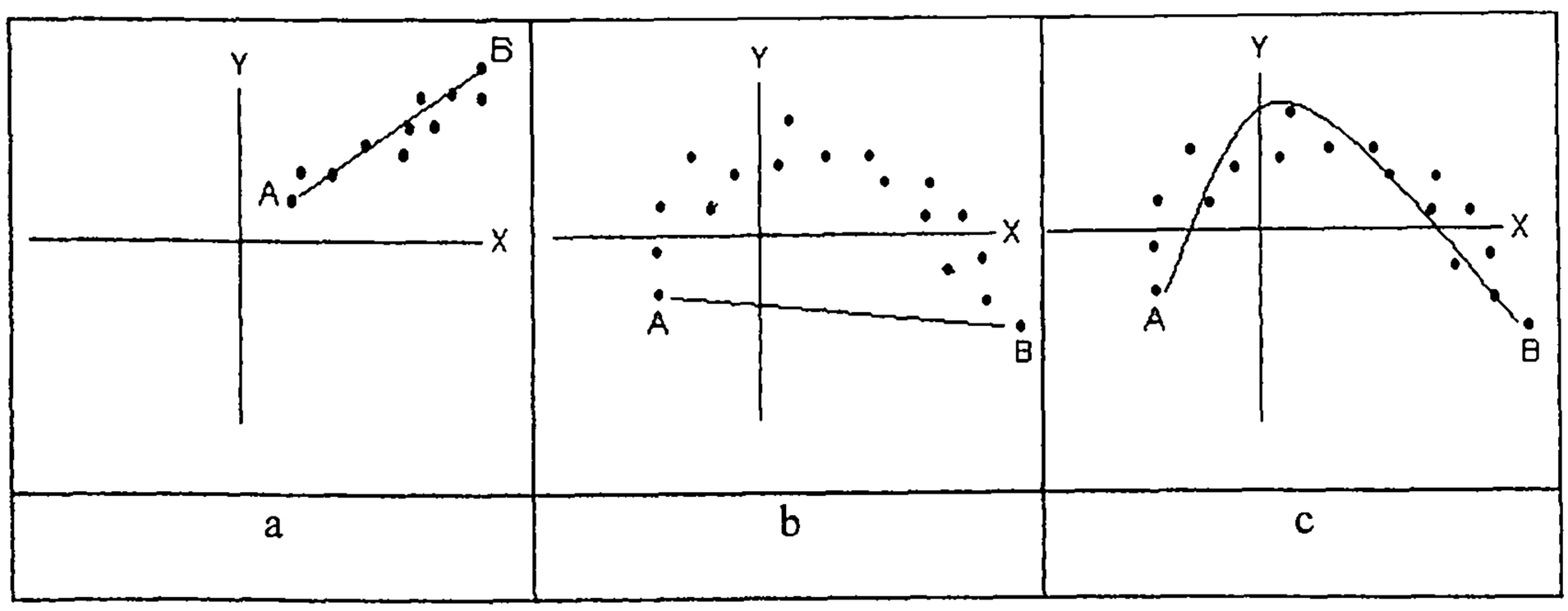


Figure 63: Distance measurement in linear and nonlinear manifolds

Because, in Figure 63a, the relationship between the *x* and *y* variables is strongly linear, a linear distance measure between points gives a good indication of the true distance between them on the manifold. In Figure 63b, however, a linear measure seriously underestimates the distance between A and B on the nonlinear manifold; what is required is a nonlinear distance measure that follows the surface of the manifold rather than cutting across it, as in Figure 63c. The SOM does not, however, use such a measure. Instead, by the topology preservation method described above, it follows the surface of the manifold, be it linear or nonlinear, as it □eighbourho it:

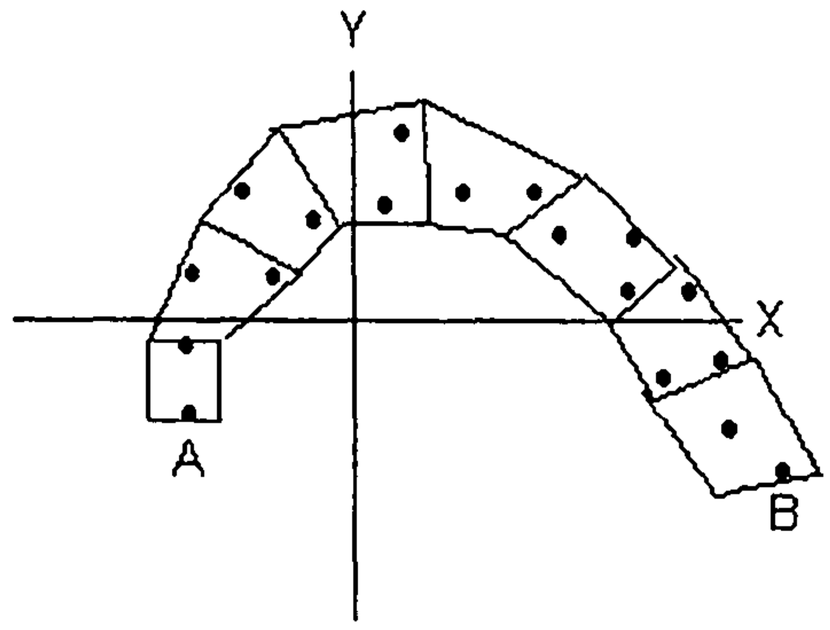


Figure 64: SOM □eighbourhoo of nonlinear manifold in Figure 63

If the topology of the input manifold is nonlinear, therefore, the SOM preserves that nonlinearity in the mapping to the output lattice, subject to the limitations mentioned above.

6.□.□.□.□ *Theoretical problems*

The SOM has a variety of theoretical liabilities: see Bishop *et al.* (1998); Kaski (1997); Kaski & Lagus (1996); Kohonen (2001, chs. 3, 5); Hammer & Villmann (2001); Ultsch & Herrmann (2005); Ultsch & Mörchen (2005); Verleysen (1997). These are summarized by Bishop *et al.* (1998, 215):

> 'While this algorithm has achieved many successes in practical applications, it also suffers from some significant deficiencies, many of which are highlighted in Kohonen 1995. They include: the absence of a cost function, the lack of a theoretical basis for choosing learning rate parameter schedules and □eighbourhood parameters to ensure topographic ordering, the absence of any general proofs of convergence, and the fact that the model does not define a probability density. These problems can be traced to the heuristic origins of the SOM algorithm'.

To this list can also be added computational complexity, which scales linearly with the number of data items, but quadratively with the number of map units –if the number of items in the data is large it may be necessary to have a large lattice to display the cluster structure clearly, to the point where training becomes too time consuming to be practical (Kaski 1997; Vesanto & Alhoniemi 2000; Vesanto 2002; Lagus 2002; Lagus *et al.* 2004).

Some of these problems are more relevant to present concerns than others. Computational complexity turns out not to be an issue because, in the present application, the map size actually used in the analyses that follow was moderate and the SOM required commensurately moderate training times. The lack of a cost function and the SOMs failure to define an explicit probability density function are theoretically more important, since the lack of these things prevents the SOM being analyzed in terms of existing probability theory and thus establishment of a convergence proof for any but 1-dimensional SOMs; all the above-cited references point out, and the SOM's inventor Kohonen himself admits (Kohonen 2001, 127 ff), that mathematical analysis of the SOM has proven difficult for these reasons. On the other hand, Bishop *et al.* (1998) in their turn admit in the above quotation that none of this has prevented the SOM being widely and successfully applied. In essence, therefore, the position is that anyone using a SOM has to accept that, at present, there are aspects of it which are imperfectly understood, and to the extent that this is so undermines the scientific validity of the results it generates. This is the position taken here.

More directly relevant is the need, in any specific application, to define a range of parameters for the SOM that have no clear scientific or mathematical motivation; this applies not just to learning rate and neighbourhood function definitions, but also to the size and shape of the output lattice, lattice edge connectivity (Ultsch 2003b; Ultsch & Mörchen 2005; Ultsch & Herrmann 2005), and connection vector initialization (Verleysen 1997).

The general view in the SOM literature appears to be that the specific choices made in any given case are 'important but not critical' (Verleysen 1997; Kononen 2001, ch.3). Numerous applications have shown SOMs to be fairly insensitive to variation in parameter selection (Kohonen 2001, ch. 3) in the sense that, relative to a given data set, they give essentially the same result across a wide range of combinations. Users typically rely on empirically-based rules of thumb (i.e. Kohonen 2001, ch.3, especially 259-61; Ultsch & Herrmann 2005) and on experimentation with different settings until satisfactory results are obtained. In other words, parameter selection is 'heuristic' –a selection is made from the theoretically infinite and practically very large number of possible parameter combinations in accordance with what the user thinks will work best. This 'it works in practice' approach is, of course, scientifically dubious, and attempts have been and are being made to address the problem. SOM and SOM-inspired architectures that learn output topology and various other parameters dynamically as training proceeds have been developed (Verleysen 1997; Kaski 1997), as have ways of measuring the quality of topology preservation in the SOM output (i.e, Kaski 1997; Kohonen 2001, 161; Villmann *et al*. 1994, 1997; Bauer *et al.* 1999; Cottrell *et al.* 2001; Vesanto 2002; Kaski & Lagus 1996). The present discussion does not, however, engage with these developments for the purely practical reason that doing so in any but a superficial way would substantially prolong the discussion and take it too far from its main thrust. The aim is to use the SOM as an analytical tool, not to engage in detail with current research issues bearing on its theoretical basis. That said, it is recognized that treating the SOM as a heuristic method 'that works' compromises the reliability of results based on it, and that this must be kept in mind when interpreting those results.

6.2. Exploratory analysis of the Qur'an data

This section applies hierarchical cluster analysis to the Qur'an data created earlier. The discussion is in three main parts. The first analyzes the data matrices Q1 – Q4 using various hierarchical clustering algorithms and identifies the main *sura* clusters, the second analyzes Q1 – Q4 using a SOM and identifies the main clusters, and the third compares the results of the hierarchical and SOM analyses and assesses the significance of the comparison.

6.2.1 Hierarchical cluster analysis

The foregoing discussion of hierarchical cluster analysis described a variety of different proximity measures and clustering algorithms, and noted that, with respect to some given data D, different combinations of proximity measure and clustering algorithm can be expected to yield analyses of D that differ from one another to greater or lesser degrees. When one comes to apply hierarchical analysis to specific data, therefore, the question of which combination to choose naturally arises. The foregoing discussion also noted that there is no generally-applicable answer. Different definitions of what constitutes a cluster give different analyses of the cluster structure of D, just as different zoological taxonomies categorize animals differently. Which is best? That presupposes a goodness criterion; there are arguments for and against the appropriateness of the various cluster definitions, but no absolute criterion has emerged. Given a range of possibly-different analyses generated by a range of proximity measure / clustering algorithm combinations, therefore, there is no obvious criterion for choosing among them (Thabet 2005). Given also that the purpose of exploratory analysis is to generate hypotheses about a domain of enquiry rather than to provide definitive answers to rigorously formulated questions, the sensible course is to interpret any differences in the analyses as alternative views of the data in the hope that

these views will yield one or more hypotheses about the conceptual structure of the Qur'an.

Four clustering algorithms were selected for the analyses that follow. They are the most commonly used in the literature, and were described earlier: single link, complete link, average link, and Ward's method. The squared Euclidean distance measure is used exclusively for the reason cited earlier: Ward's method requires use of squared Euclidean, and for consistency across analyses it was used with the other algorithms as well. It should be noted that, at an early stage of research, Euclidean distance was tried with the non-Ward methods, but the results were almost indistinguishable from those obtained with squared Euclidean. Each of the data matrices Q1 – Q4 is analyzed using squared Euclidean in combination with each of the above four algorithms, yielding a total of 16 analyses. The raw analyses are first presented to the reader for unbiased inspection, and are subsequently discussed.

6.2.1.1 Raw cluster trees

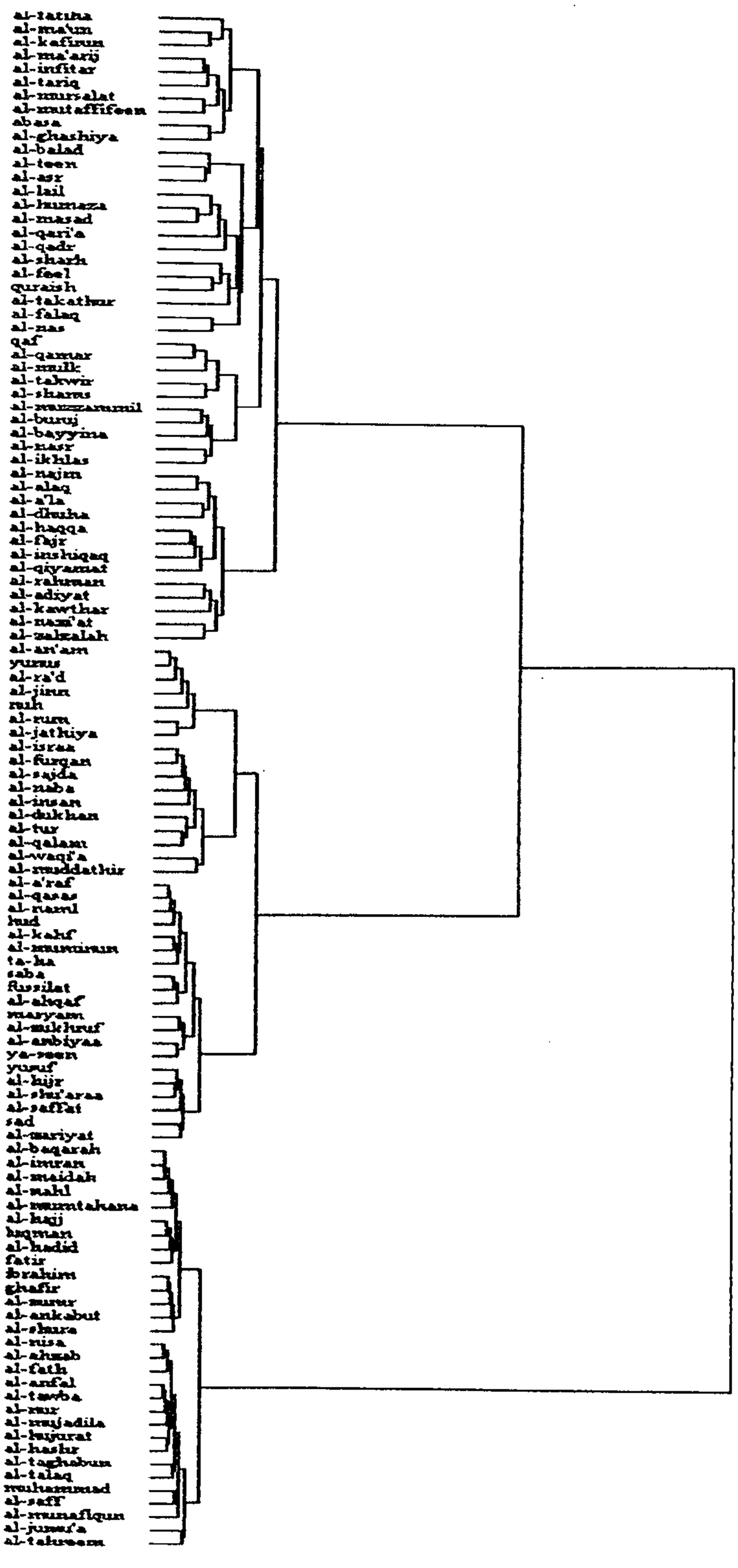


Figure 65: Q1, squared Euclidean distance, Ward's method

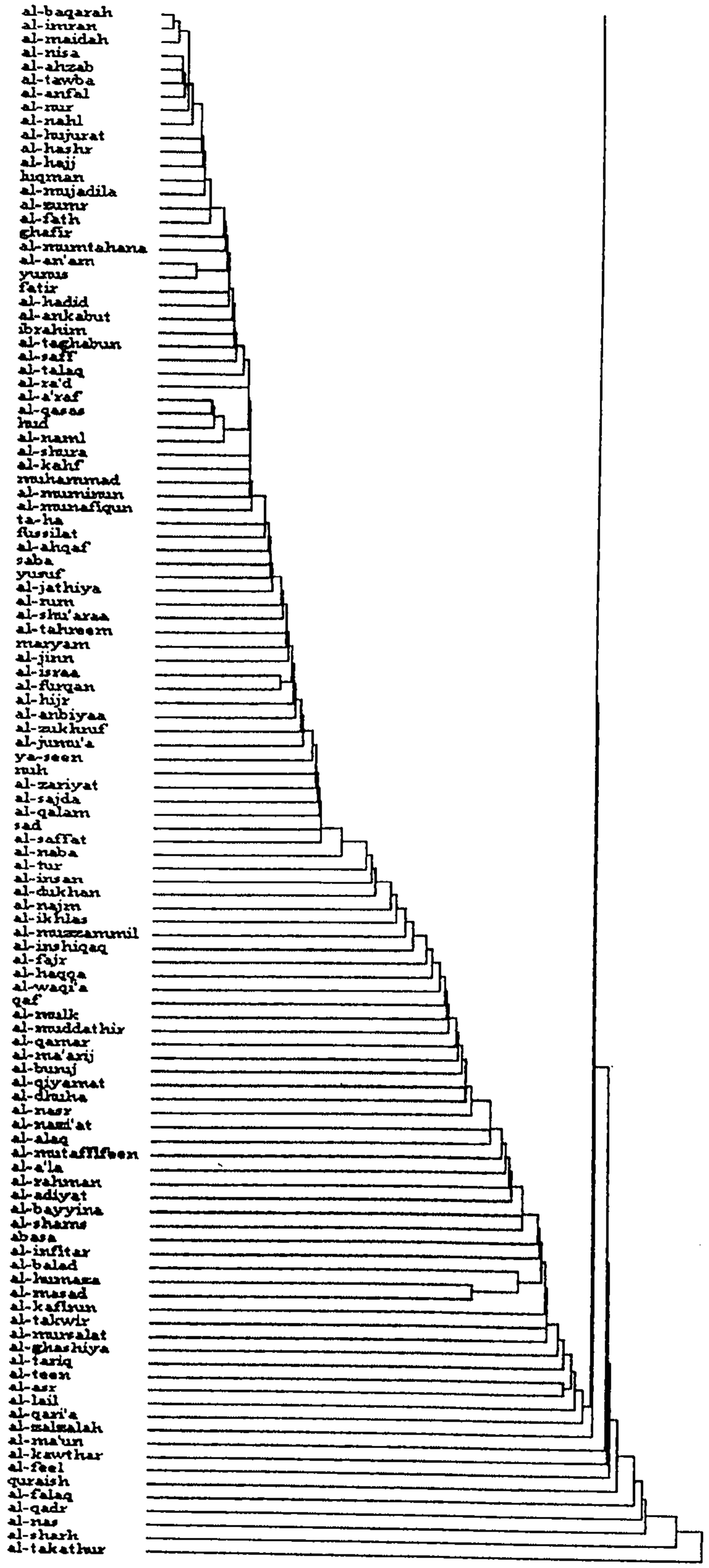


Figure 66: Q1, squared Euclidean distance, single link

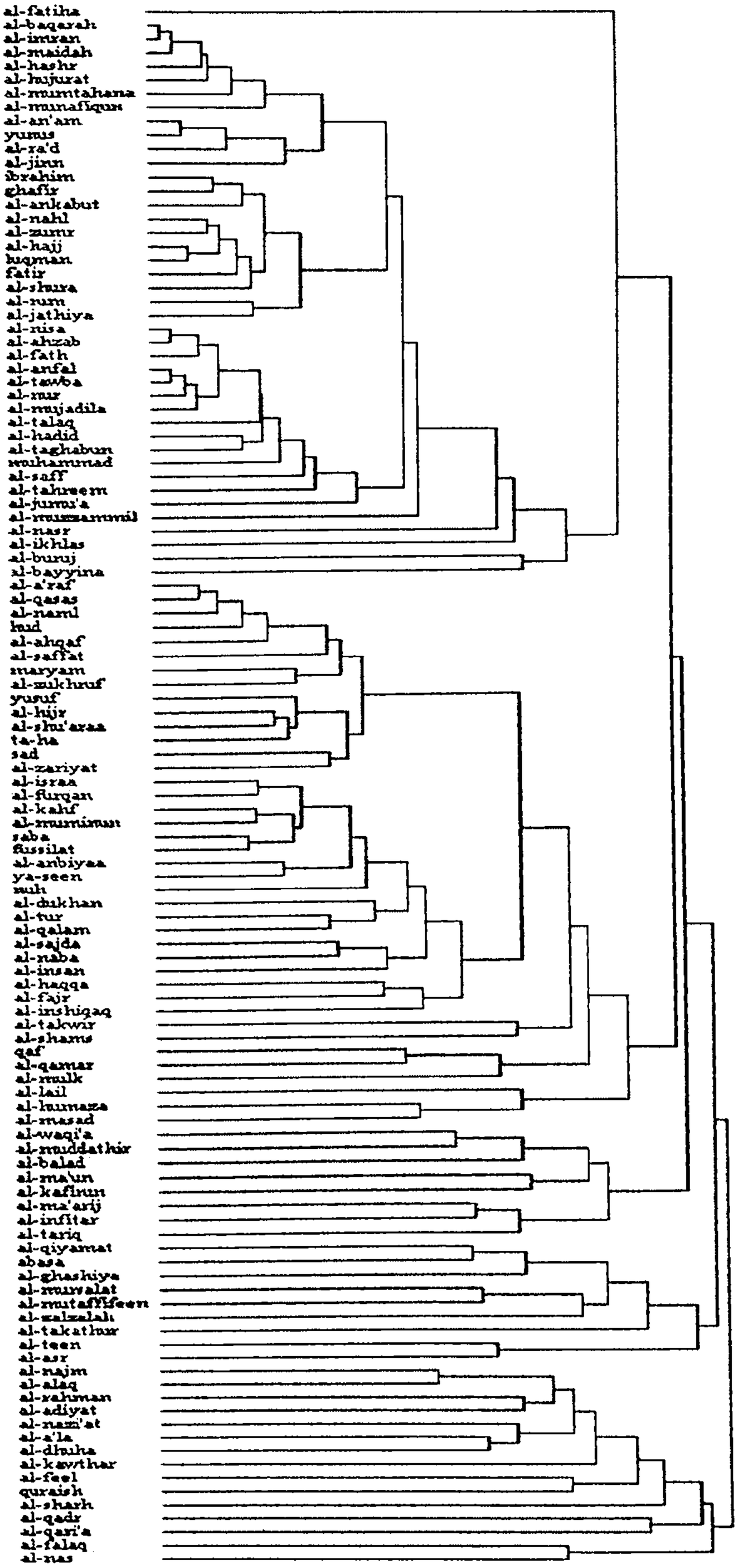


Figure 67: Q1, squared Euclidean distance, complete link

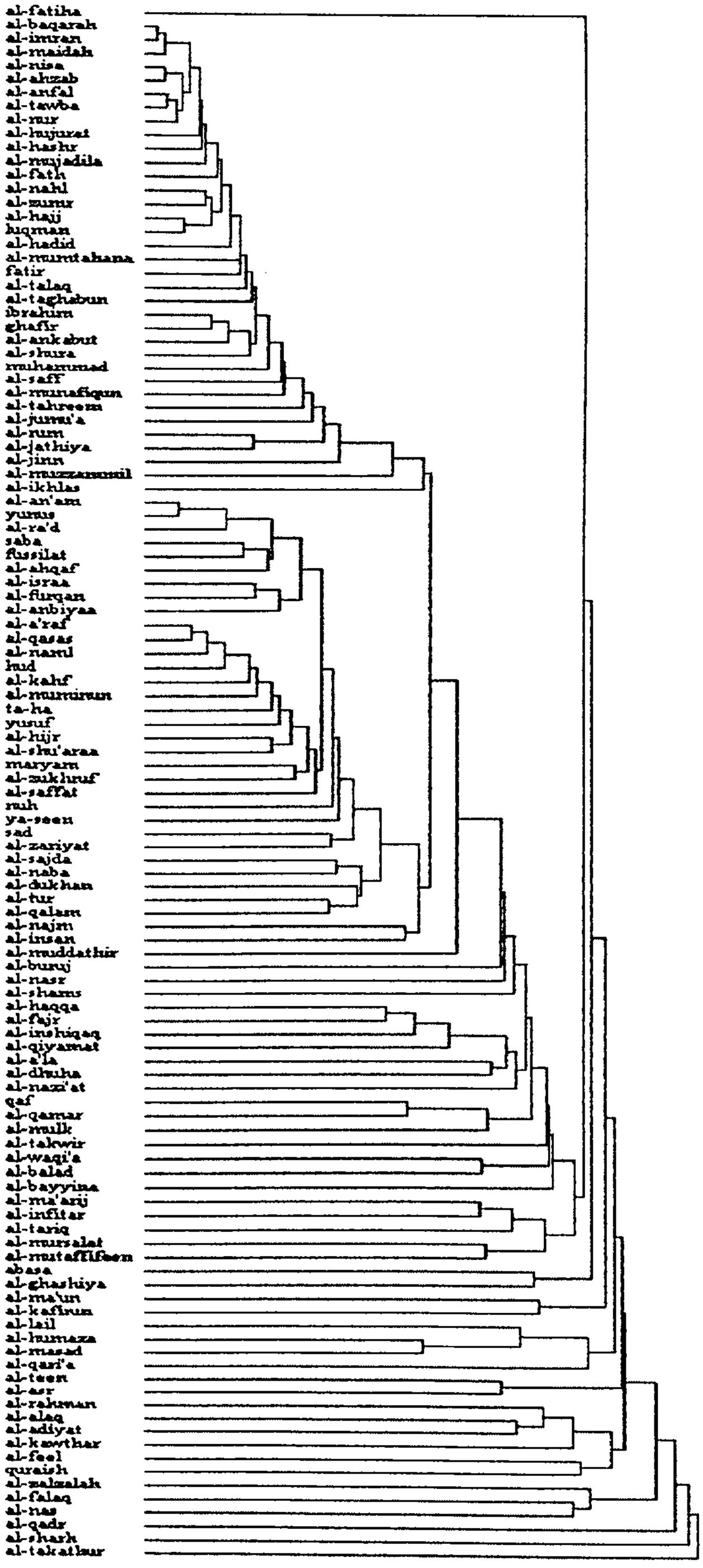


Figure 68: Q1, squared Euclidean distance, average link

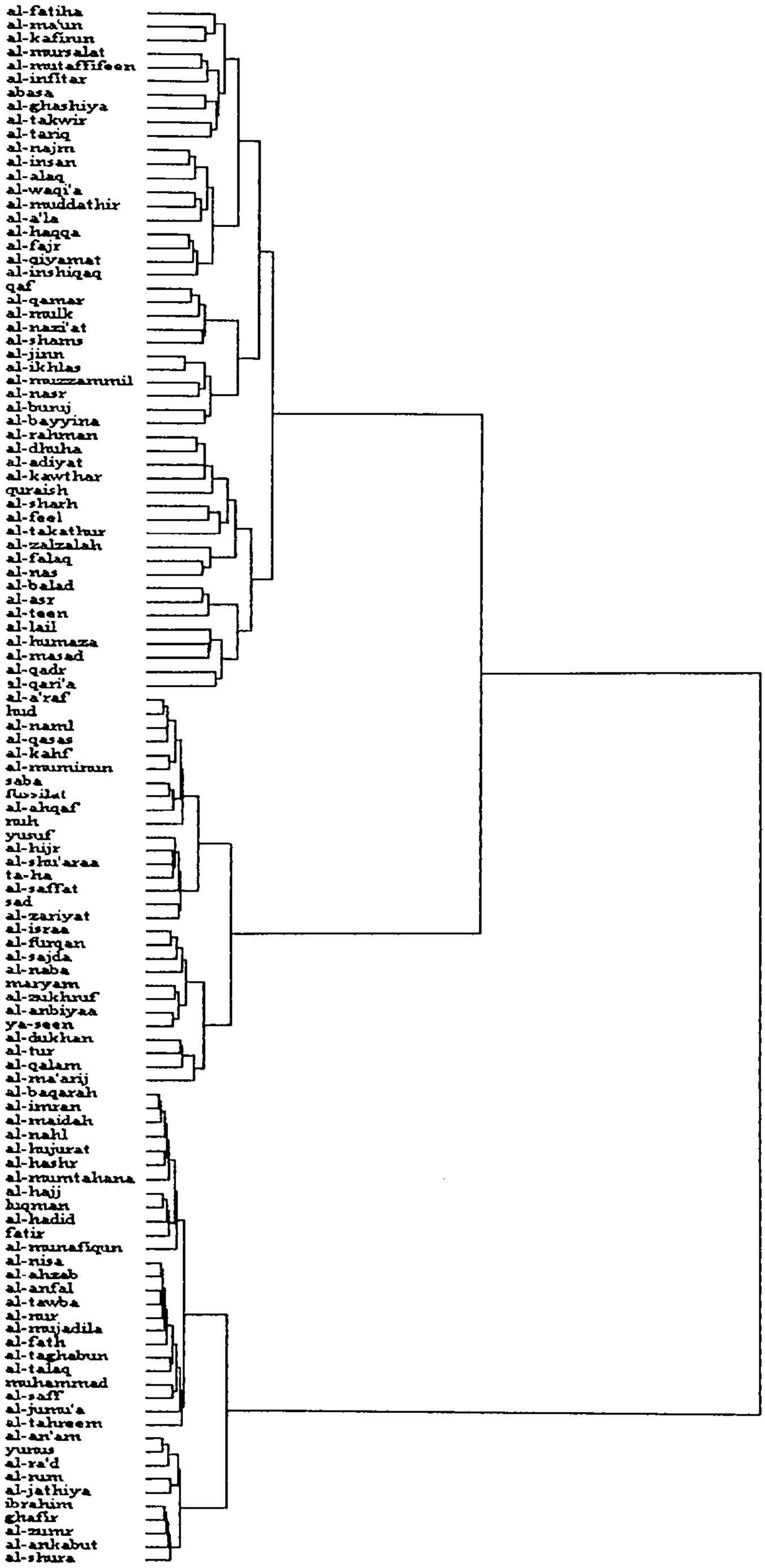


Figure 69: Q2, squared Euclidean distance, Ward's Method

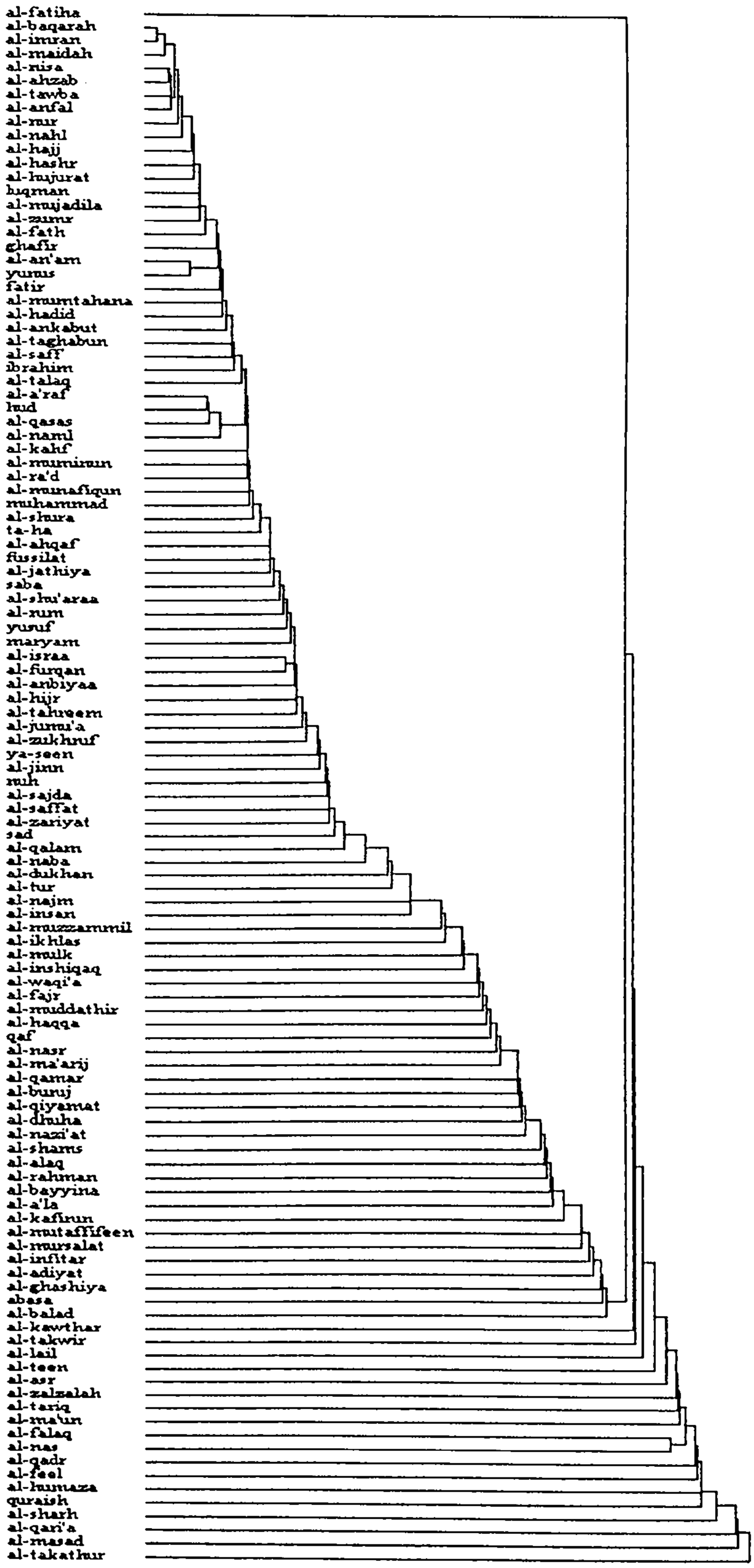


Figure 70: Q2, squared Euclidean distance, single link

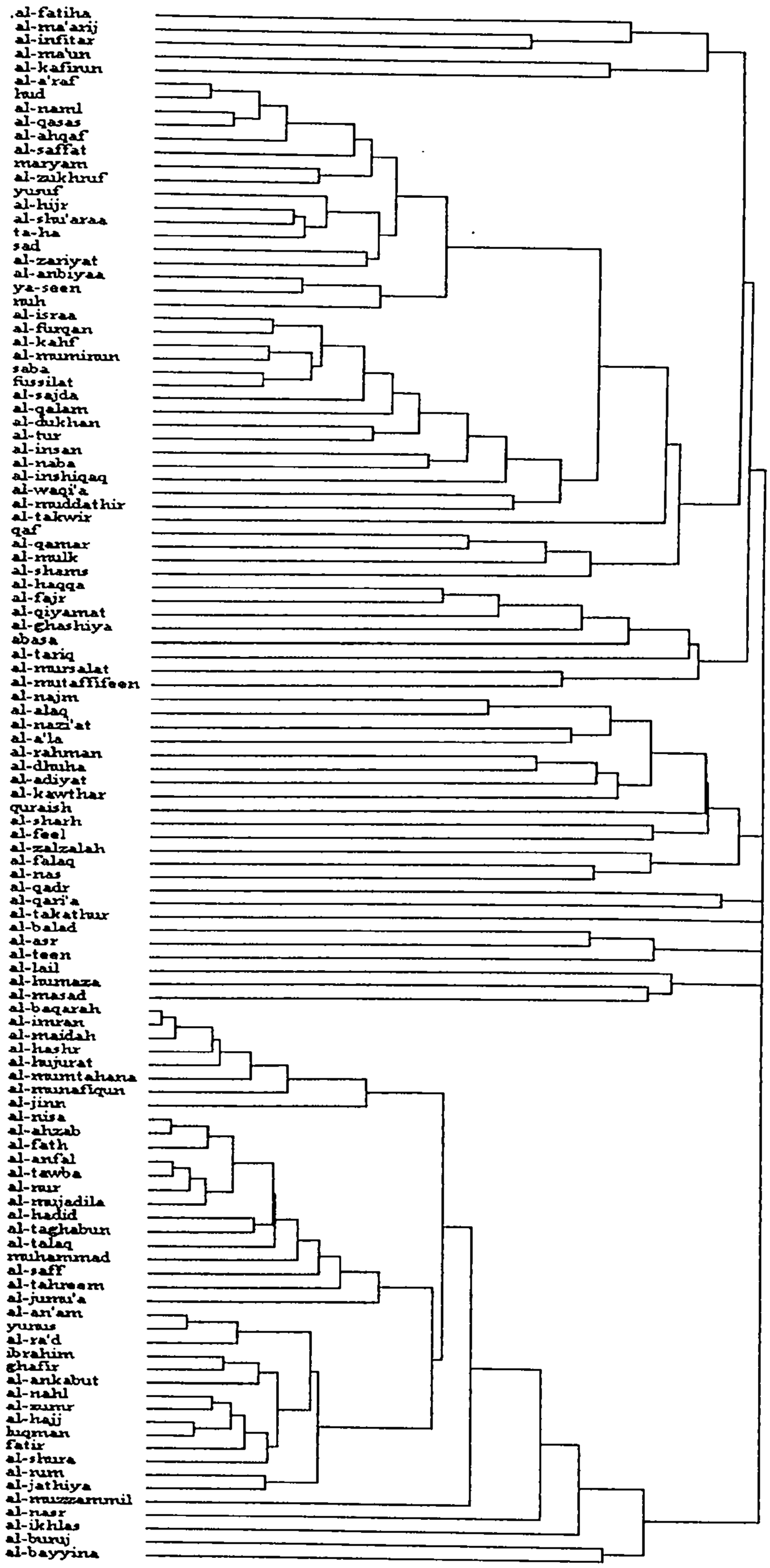


Figure 71: Q2, squared Euclidean distance, complete link

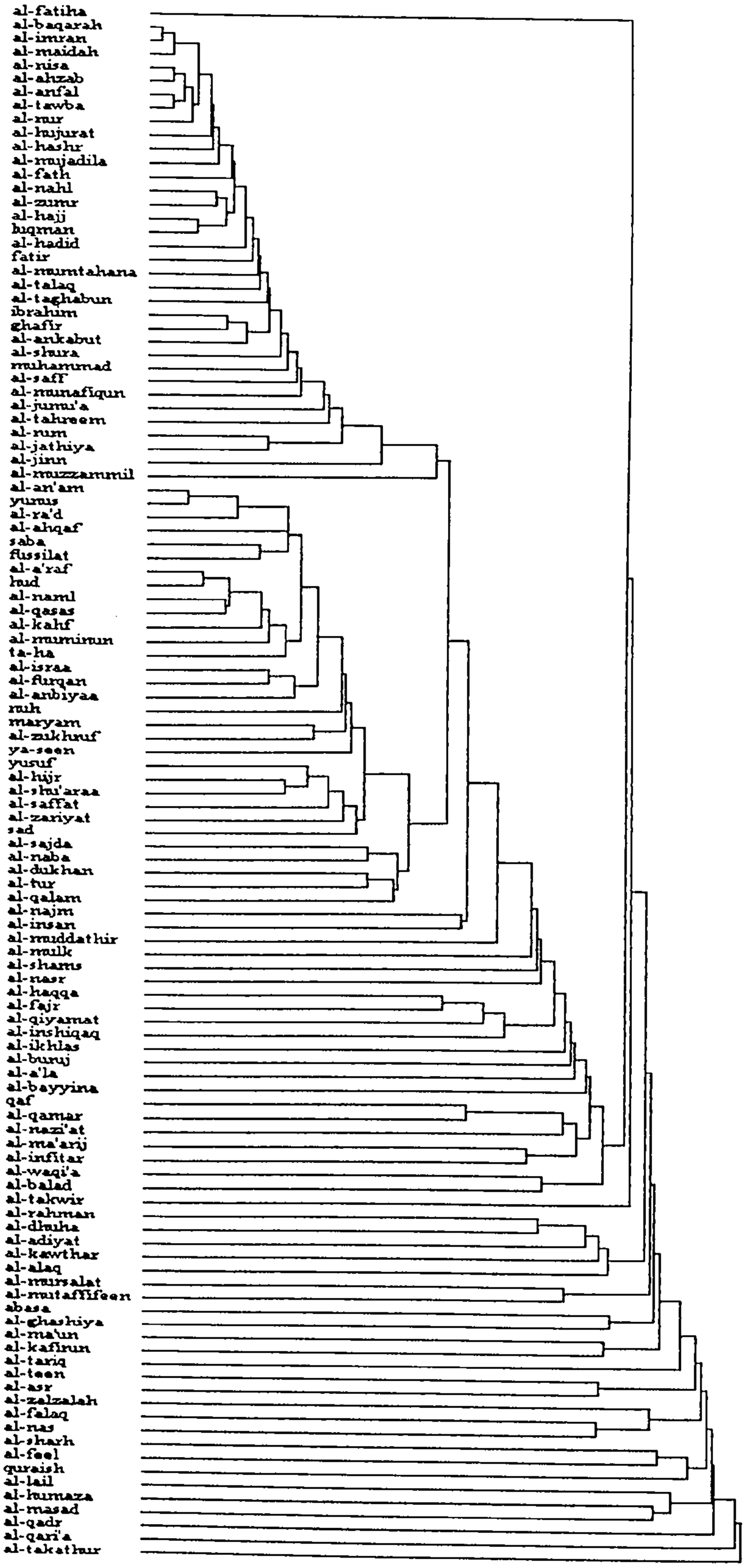


Figure 72: Q2, squared Euclidean distance, average link

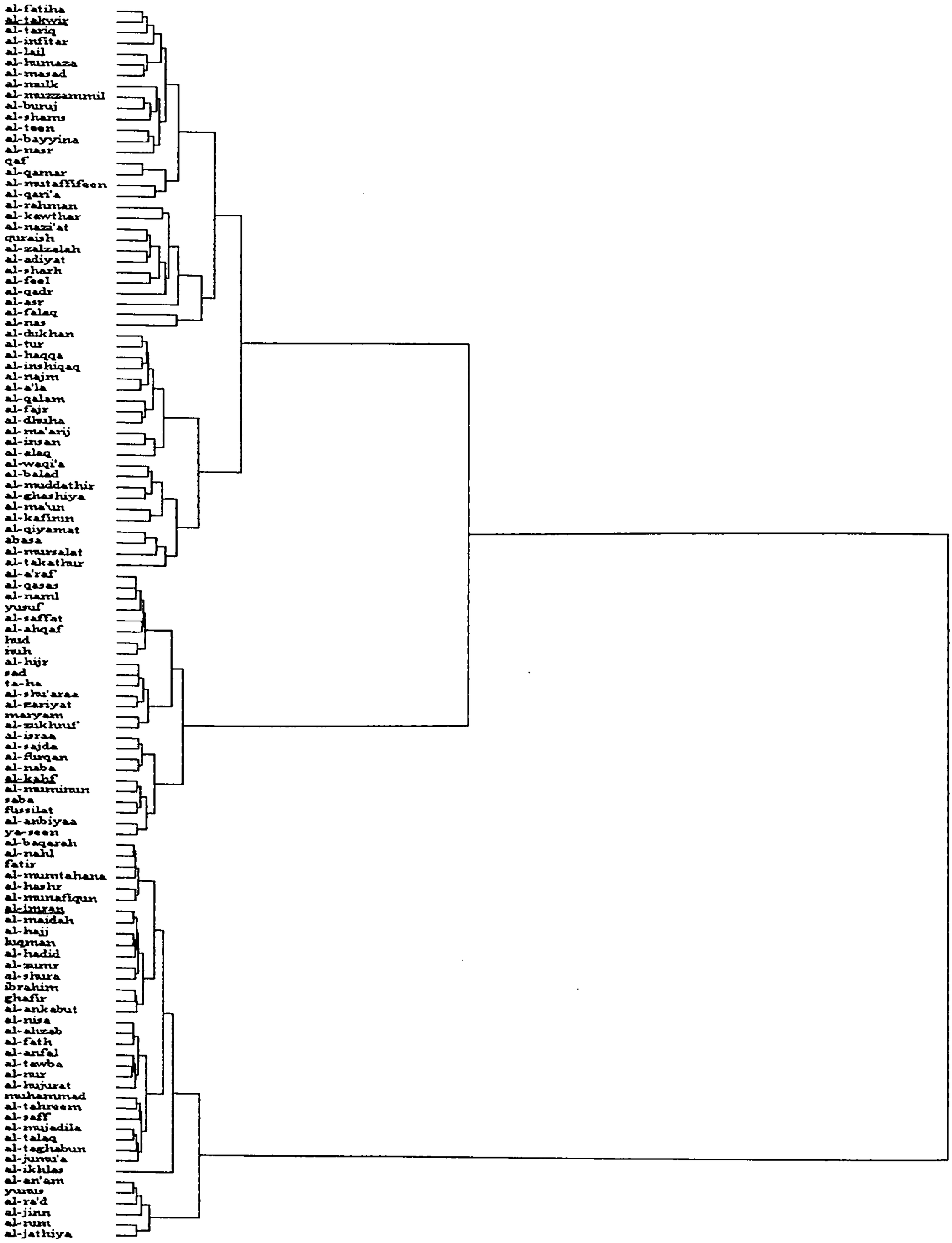


Figure 73: Q3, squared Euclidean distance, Ward's Method

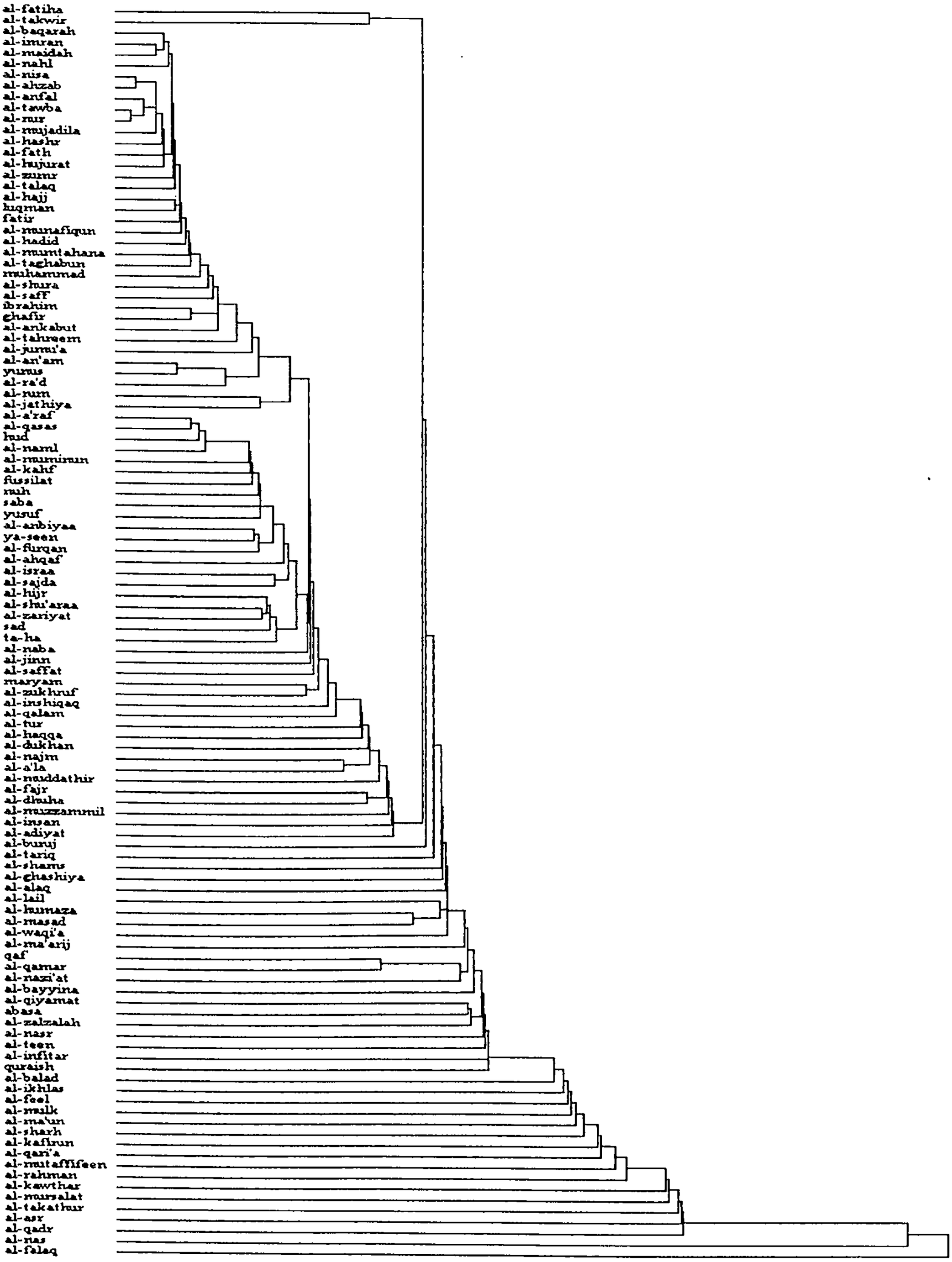


Figure 74: Q3, squared Euclidean distance, single link

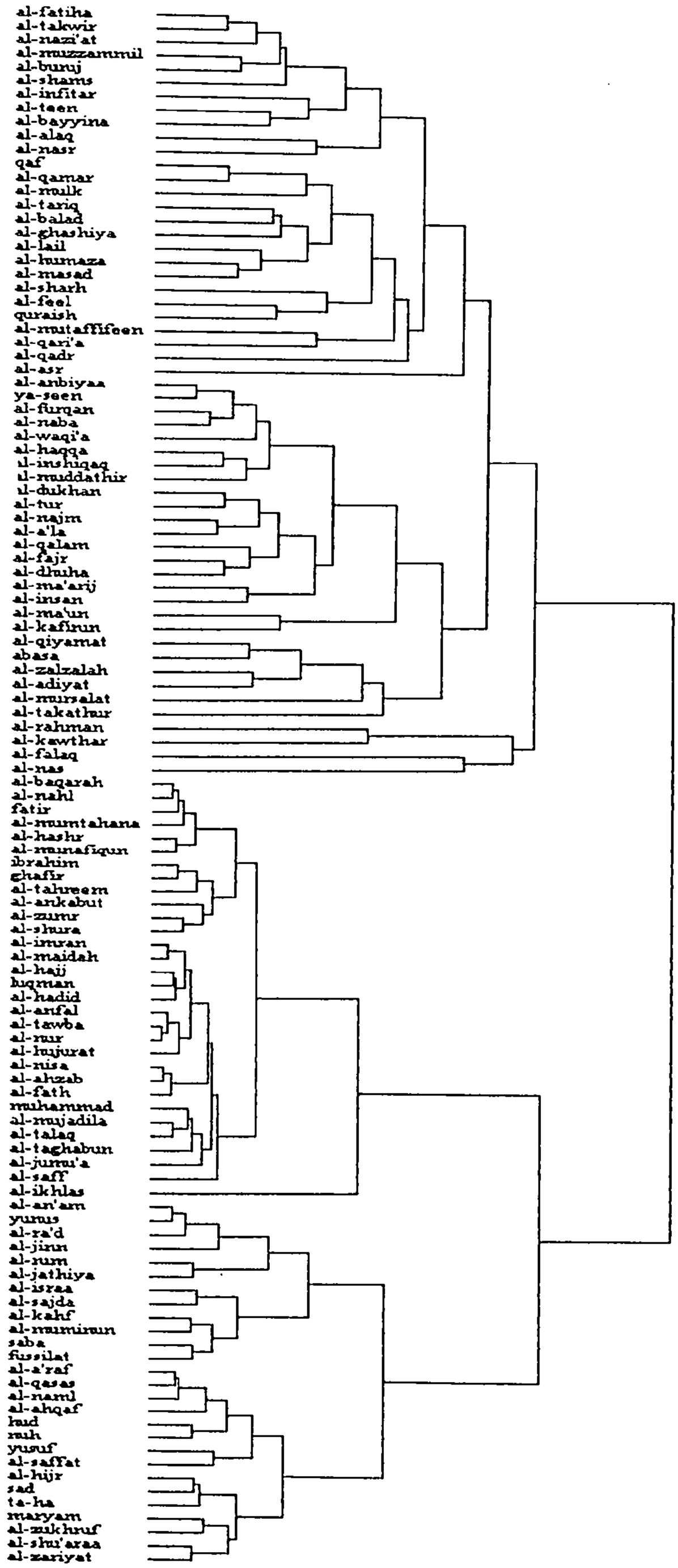


Figure 75: Q3, squared Euclidean distance, complete link

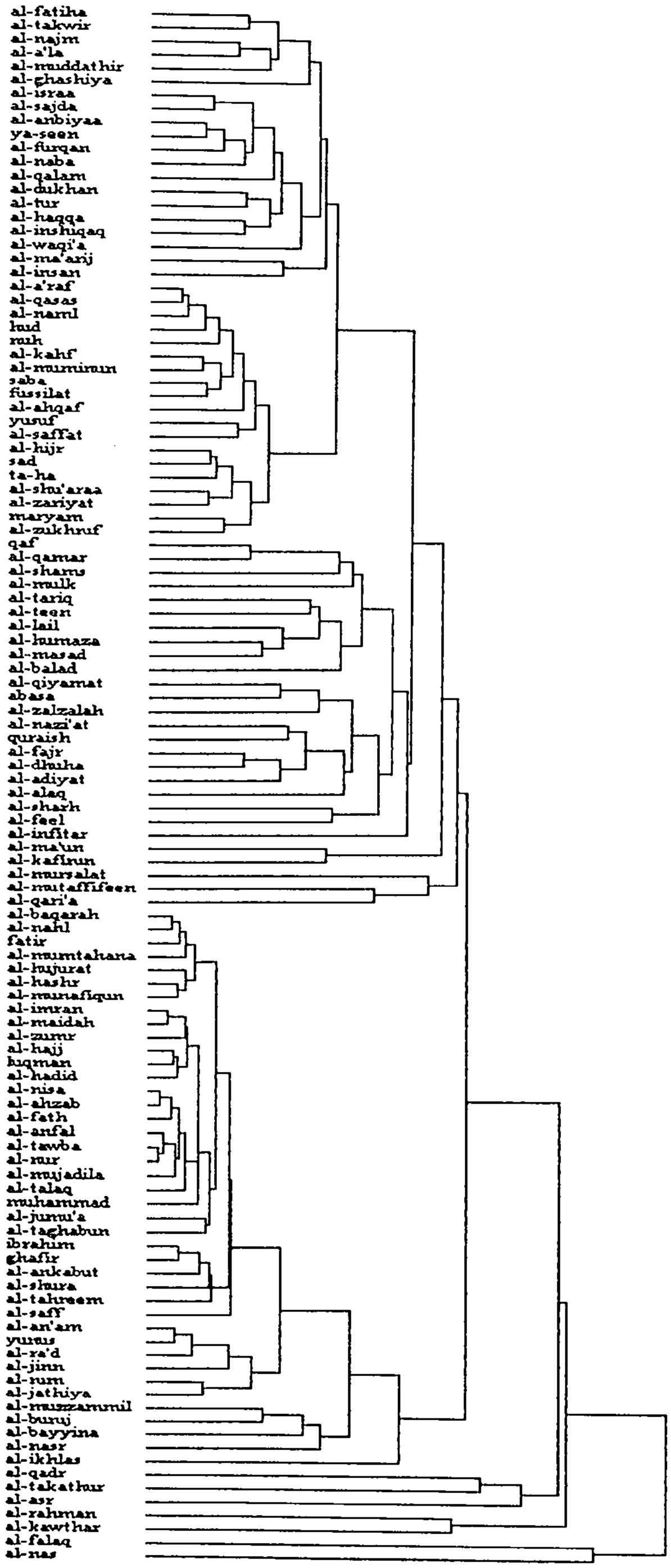


Figure 76: Q3, squared Euclidean distance, average link

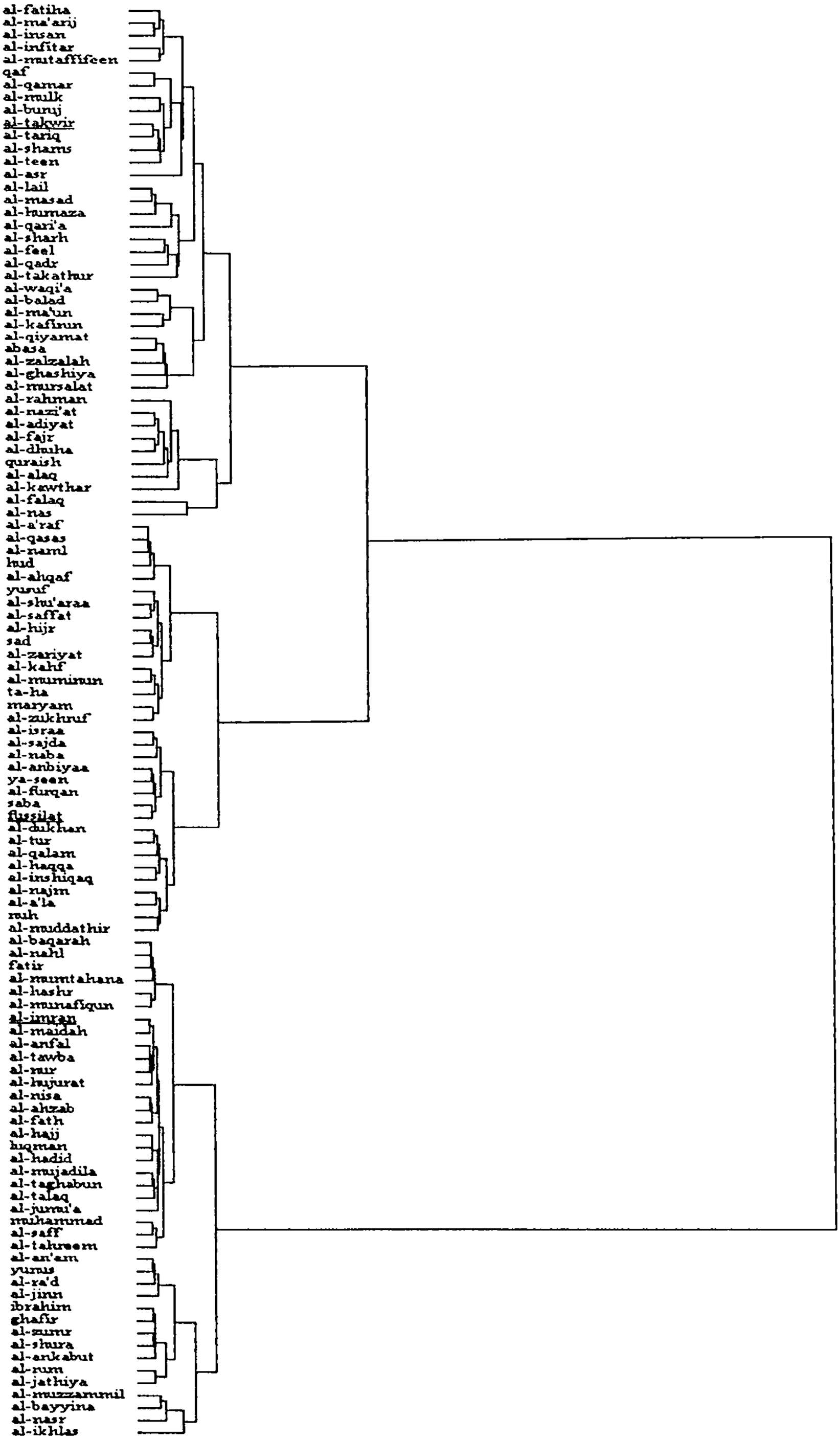


Figure 77: Q4, squared Euclidean distance, Ward's Method

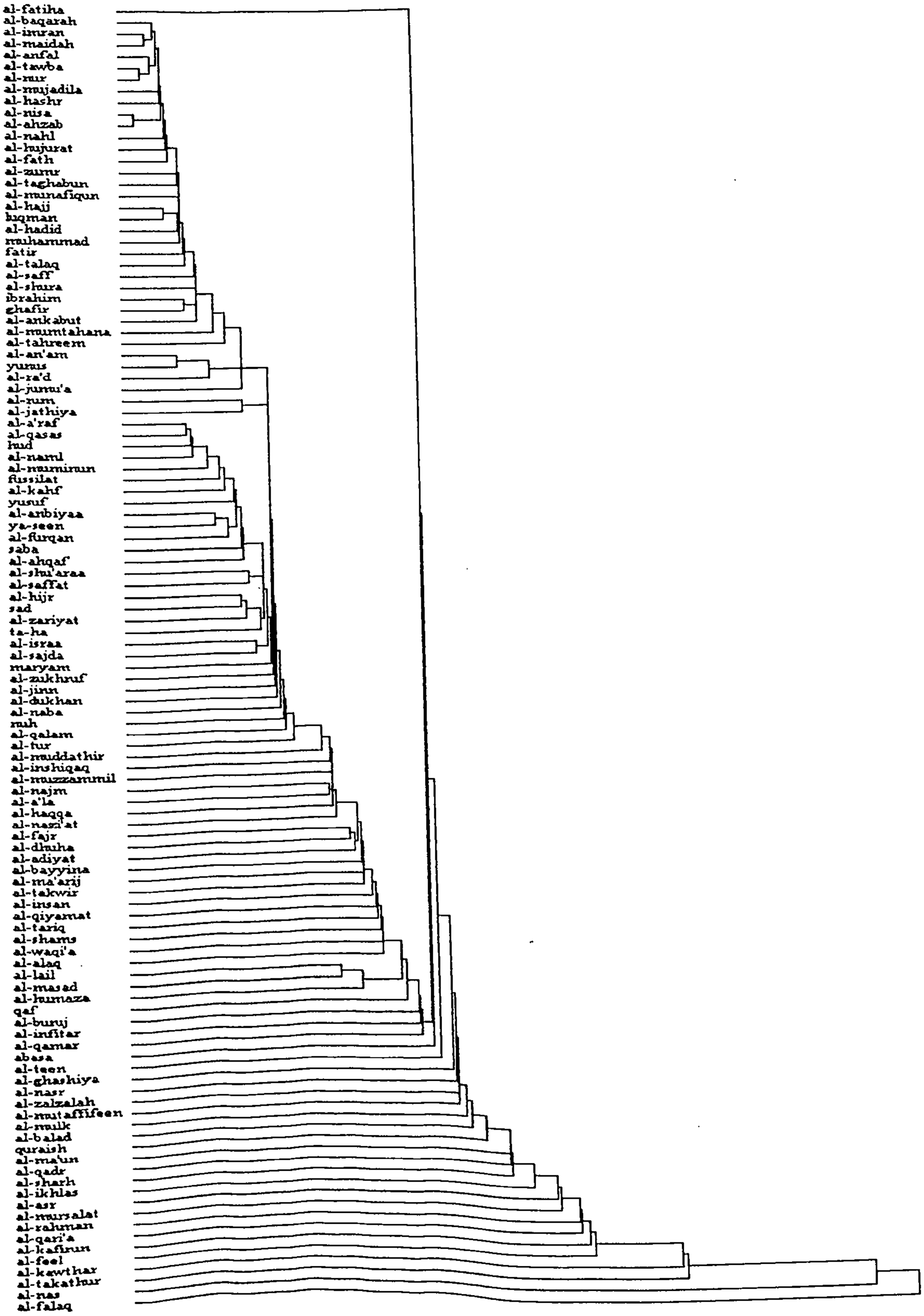


Figure 78: Q4, squared Euclidean distance, single link

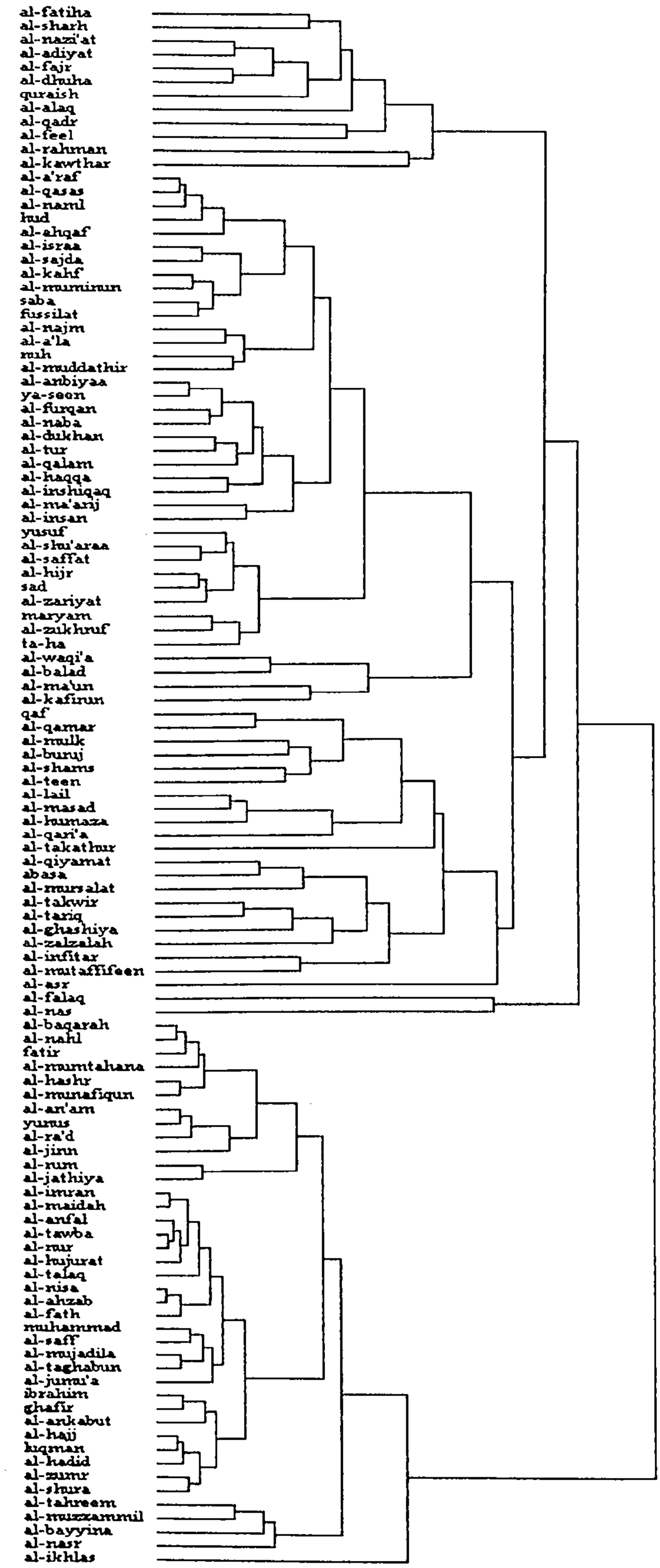


Figure 79: Q4, squared Euclidean distance, complete link

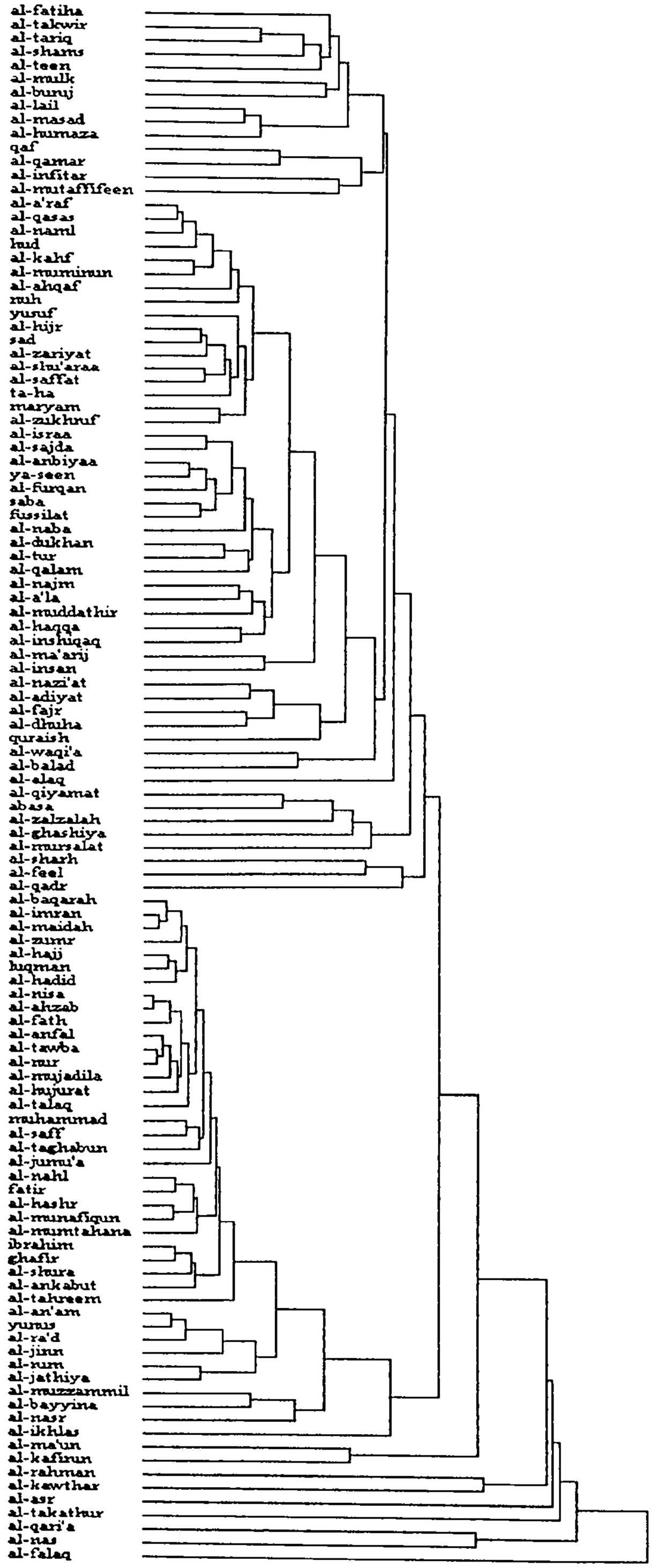


Figure 80: Q4, squared Euclidean distance, average link

6.2.1.2 Discussion

Across Q1 – Q4, each clustering algorithm structures the similarity relations among the *sura* matrix rows in its characteristic way, as discussed earlier. These characteristic structures are shown schematically in Figure 81 below; the trees have been rotated 90 degrees counter clockwise for clarity.

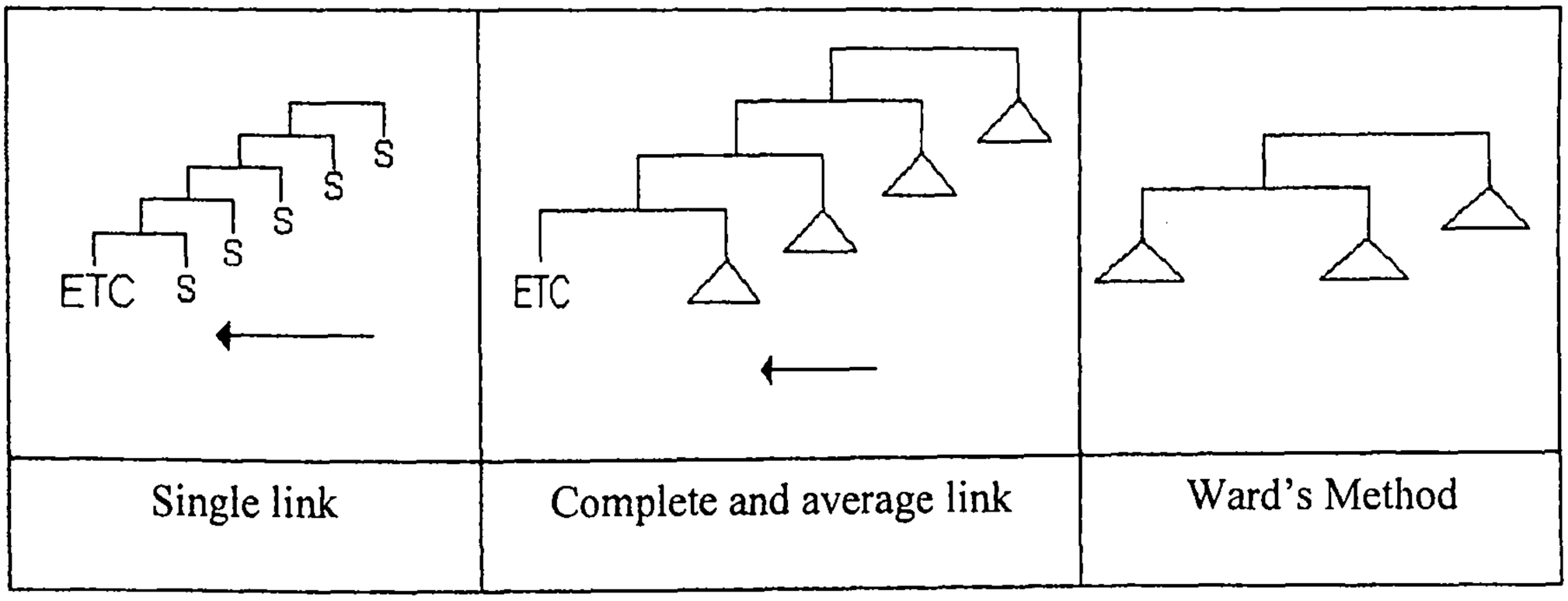


Figure 81: Schematic cluster trees

- Single link: The 'S' labels at the leaves of the tree are *sura* names, and the arrow together with 'ETC' is meant to indicate that the same structure as that shown in the diagram propagates leftwards. In the actual Q1 – Q4 trees there are occasional and shallow right branchings, but the overwhelming tendency is the left branching shown in the schematic.
- Complete and average link: The complete link tree is similar to the single link one in that there is a strong left branching tendency indicated, as before, by the left-facing arrow and 'ETC'. There is, however, also a significant element of right-branching, indicated in the schematic by the triangles at the leaves of branches. Each such triangle contains a subtree with a significant element of right branching but with the same basic left-branching tendency as that shown in the schematic; each such subtree is a recursively self-similar version of the schematic. The average

link tree morphology is similar to the complete link one, though with a generally stronger left-branching tendency and a commensurately weaker right-branching in the subtrees.

- Ward's method assigns three strongly delineated clusters to the *sura* matrix rows, and the subtrees contain both left and right branching in approximately equal proportions. The strong left-branching tendency of the other methods is absent.

The four clustering methods, therefore, assign three structures to the similarity relations among the *sura* matrix rows that flatly contradict one another.

6.2.1.2.1 *Sura* groups

The different hierarchical structures that the various methods assign to the *sura* matrix rows notwithstanding, examination of all the trees across Q1 – Q4 reveals a strong consistency in the way that the *suras* are grouped in the trees in terms of their relative distance from one another. To show this, each of the original cluster trees is graphically annotated to show the groupings; the emended trees are first presented and then discussed.

Some notes on the annotated trees:

- In comparing the trees across Q1 – Q4, the Ward's method tree is used as the standard for comparison because it makes the intuitively clearest claim about the constituency structure of the *sura* matrix rows. It is stressed that this is done for clarity of exposition only. There is no implication that the Ward's Method analysis is in any sense 'better' or 'truer' than the others.
- The three main groups are shown using square braces to the left of the *sura* names at the leaves of each tree with labels A, B, and C

- Group membership is strongly though not absolutely consistent among trees. To show variation in group membership, labels have been inserted within the square braces immediately to the left of the *sura* names; these labels are A, B, and C, and are relative to the Ward's Method tree in each case. Subsequent discussion will clarify this labelling.

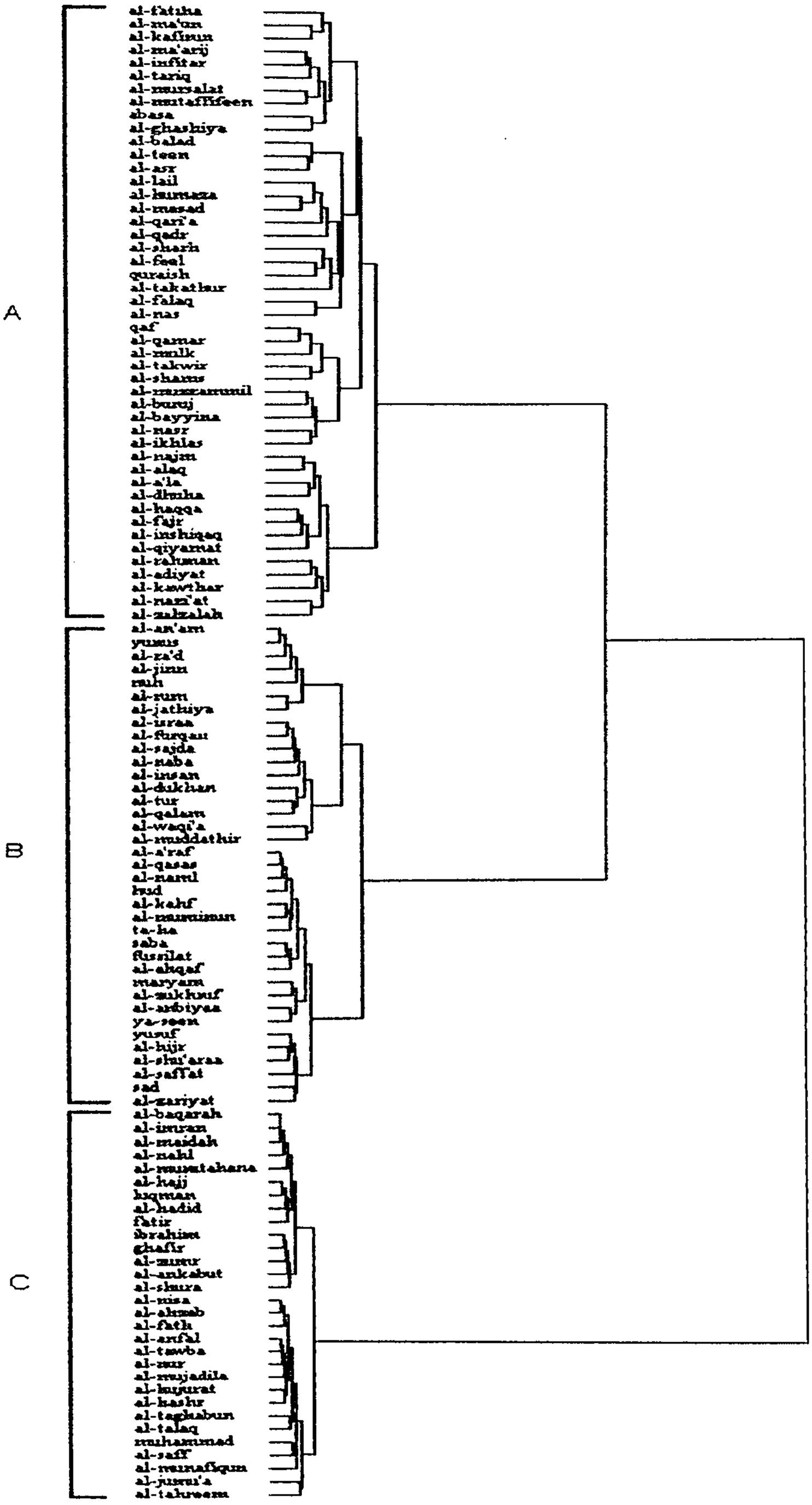


Figure 82: Q1, squared Euclidean distance, Ward's method

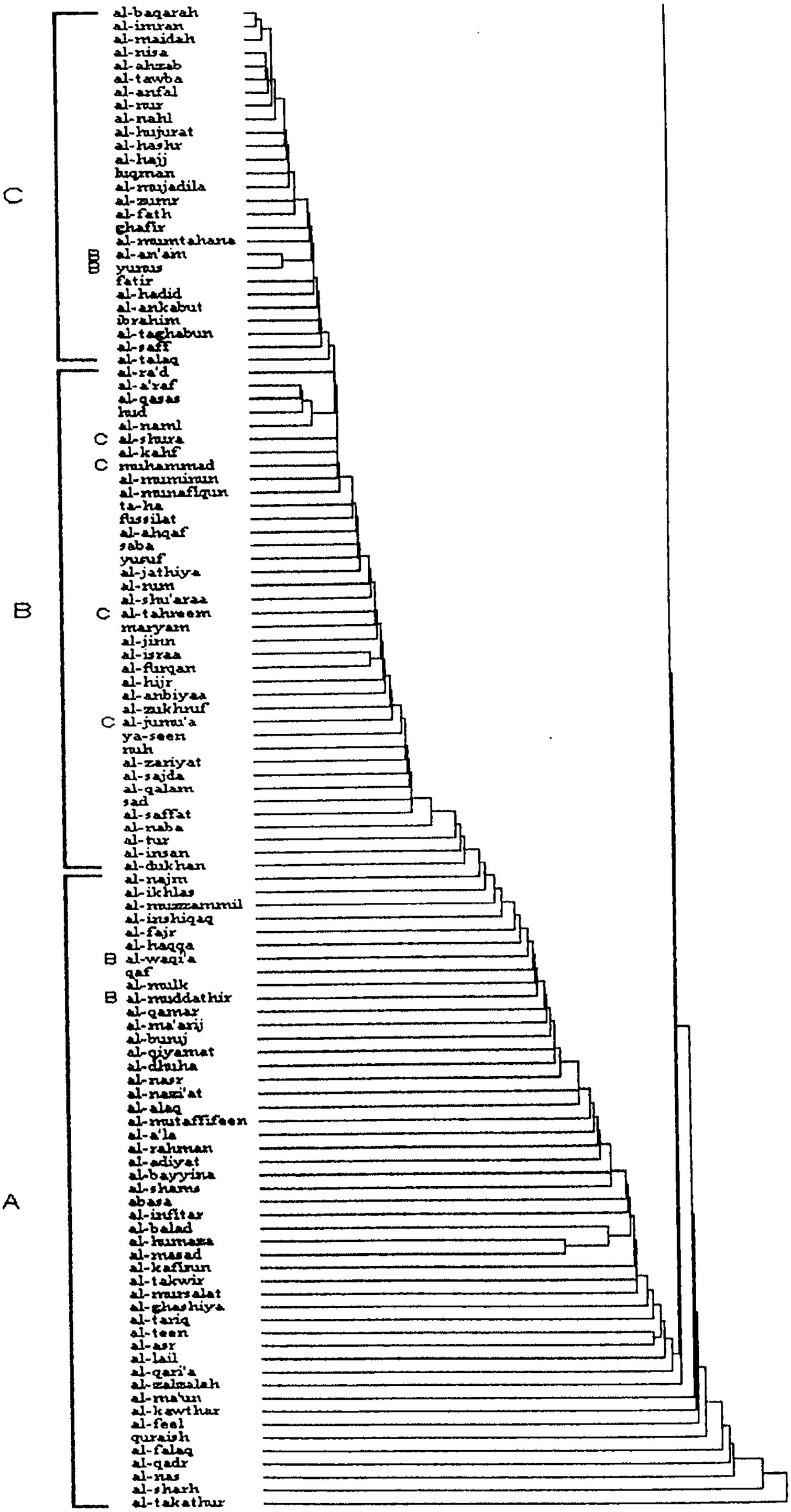


Figure 83: Q1, squared Euclidean distance, single link

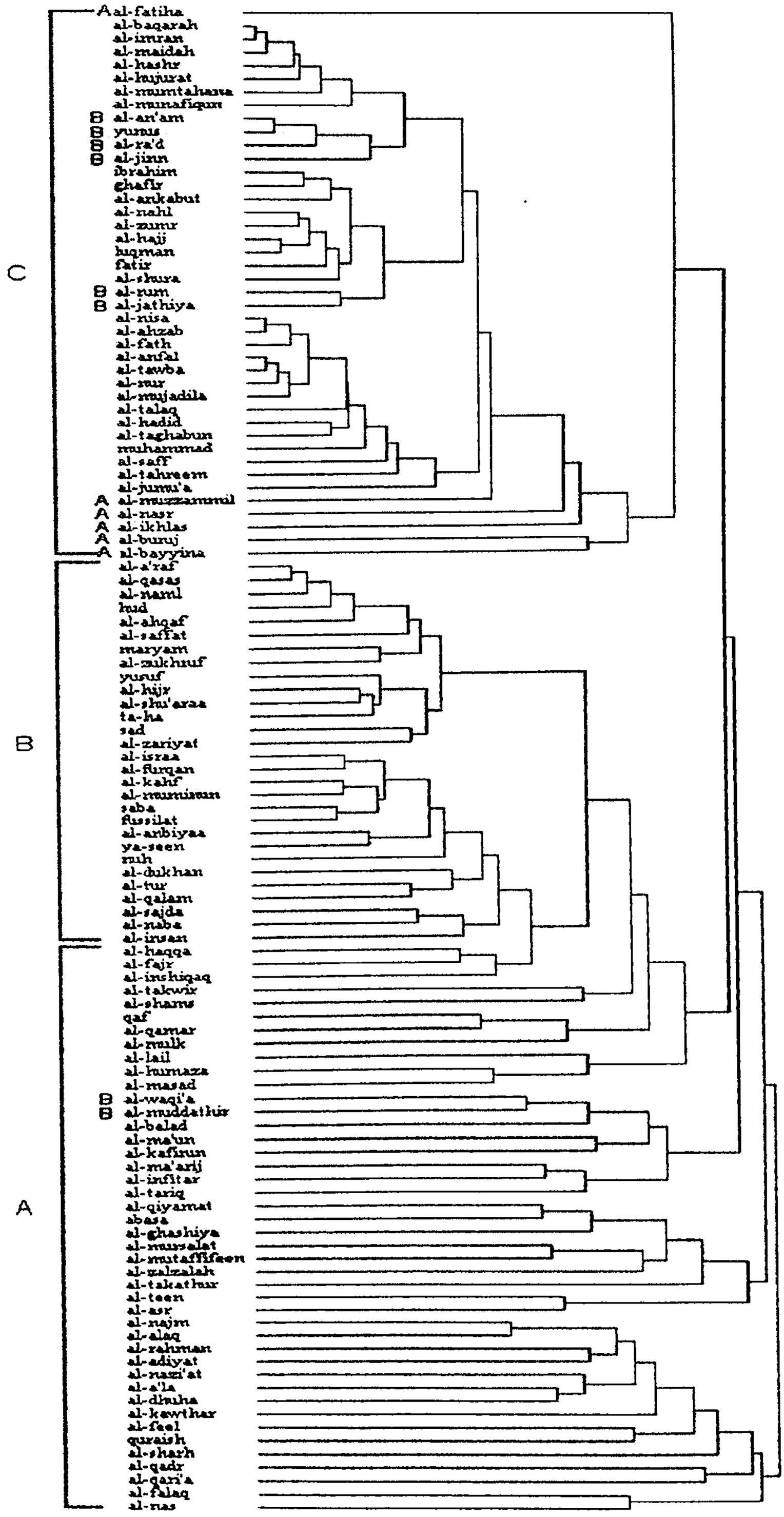


Figure 84: Q1, squared Euclidean distance, complete link

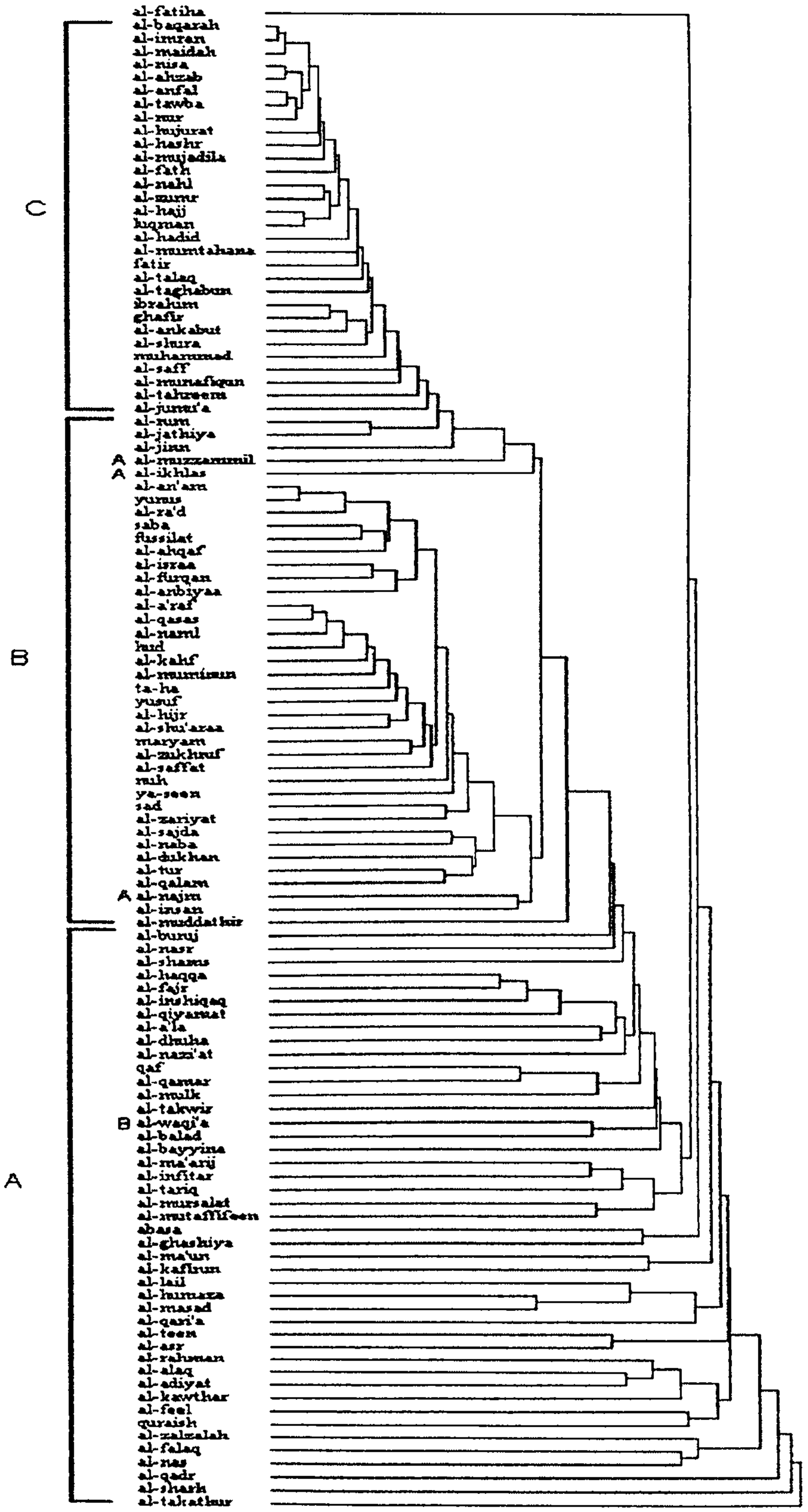


Figure 85: Q1, squared Euclidean distance, average link

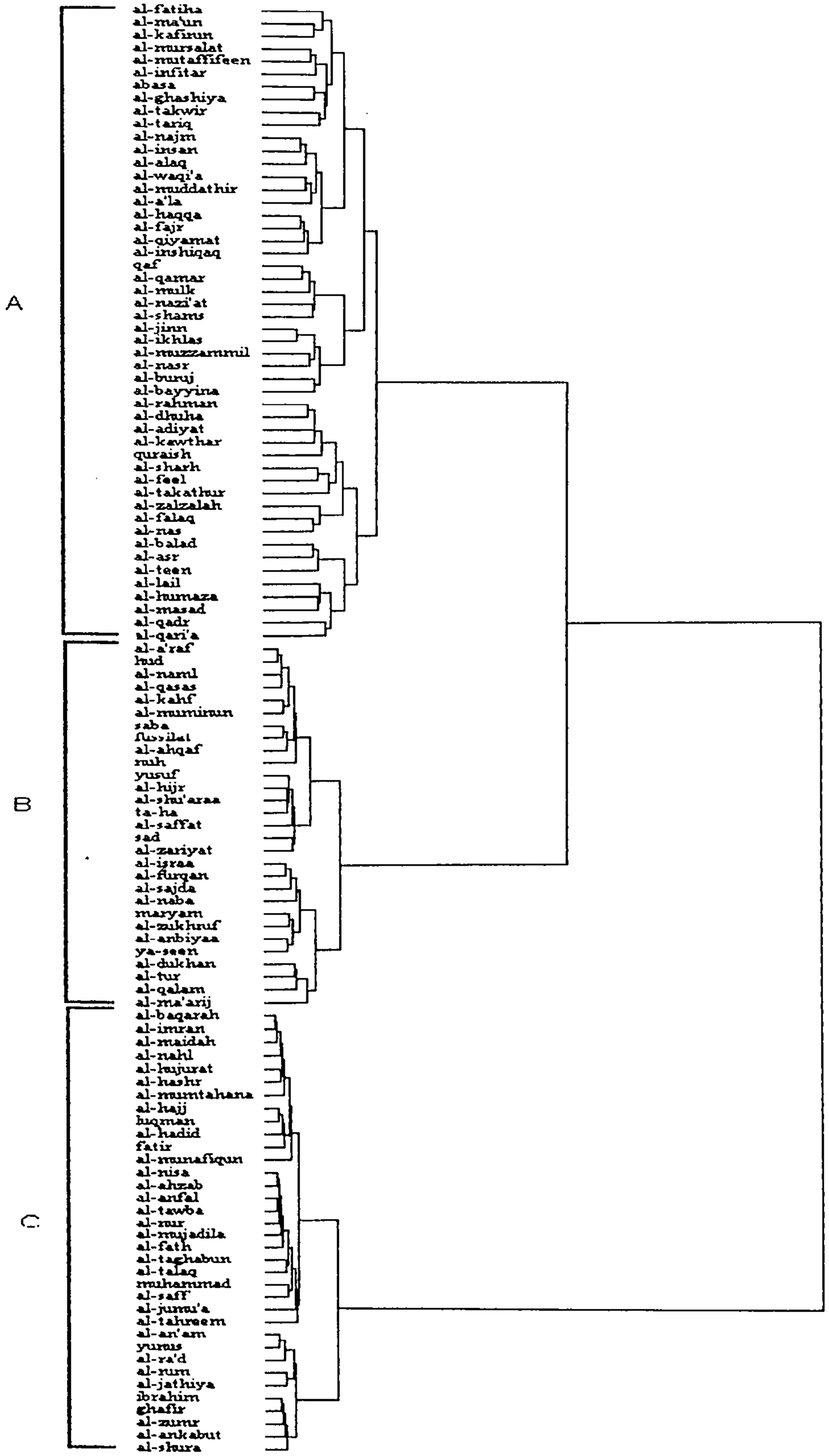


Figure 86: Q2, squared Euclidean distance, Ward's method

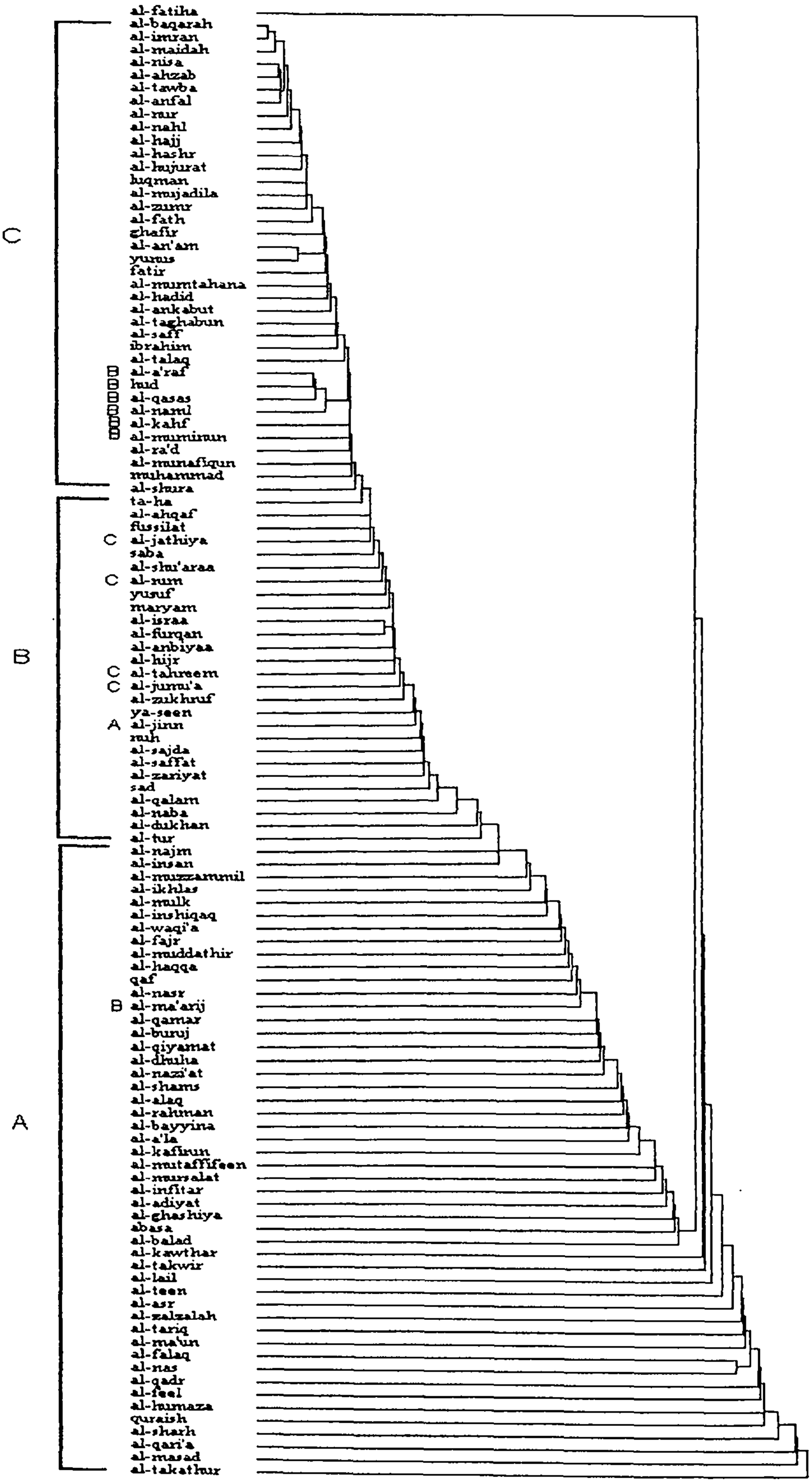


Figure 87: Q2, squared Euclidean distance, single link

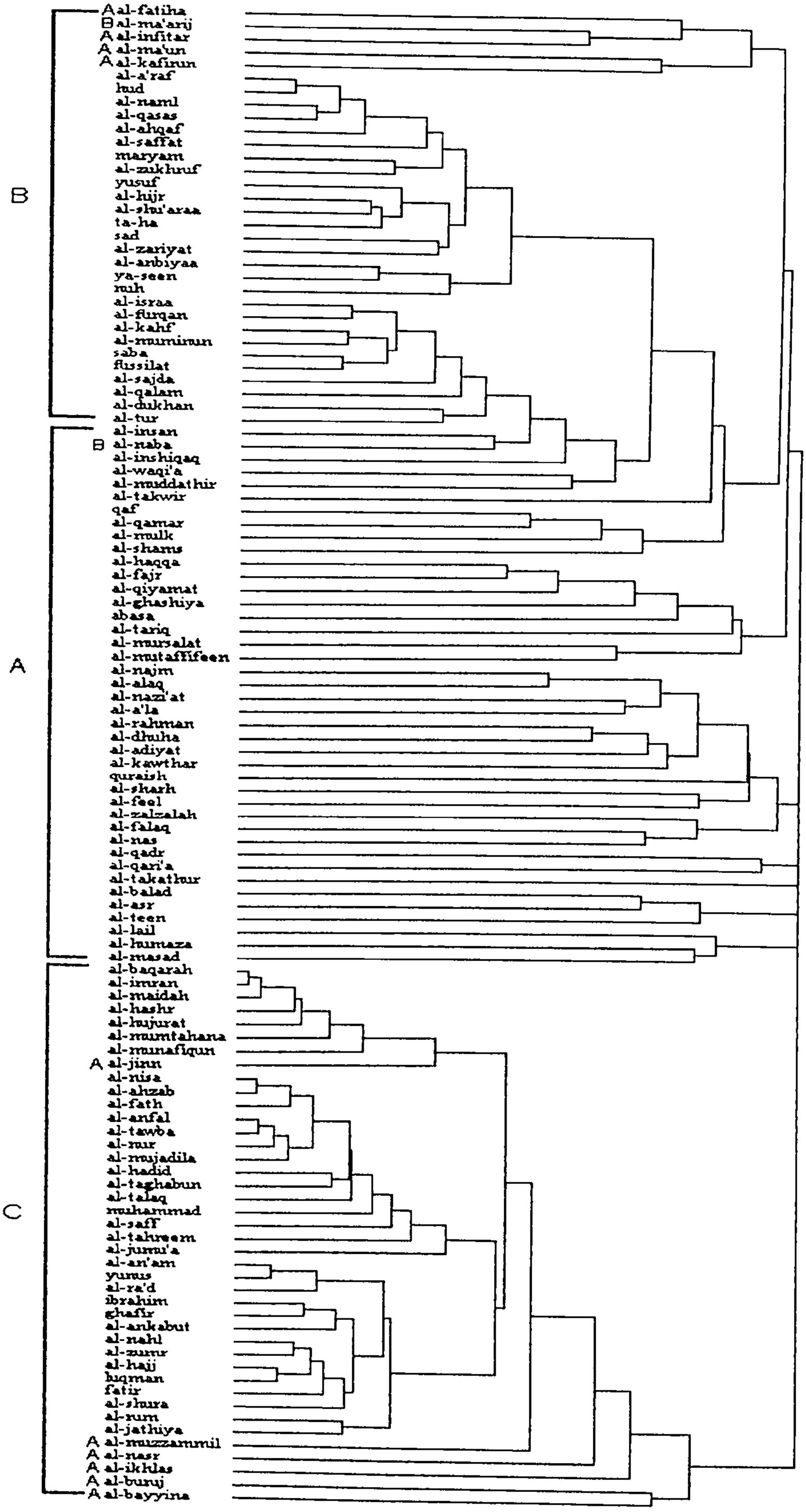


Figure 88: Q2, squared Euclidean distance, complete link

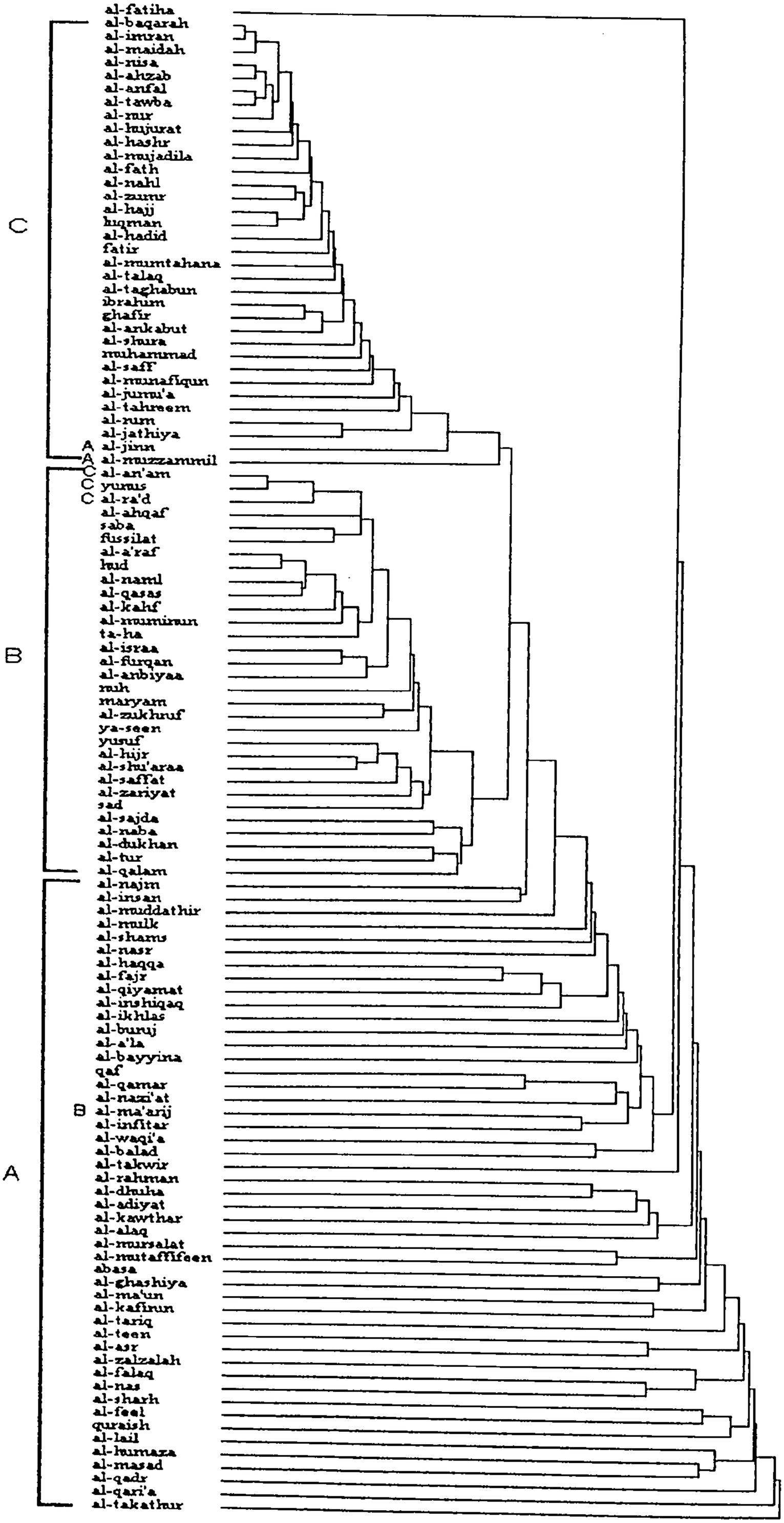


Figure 89: Q2, squared Euclidean distance, average link

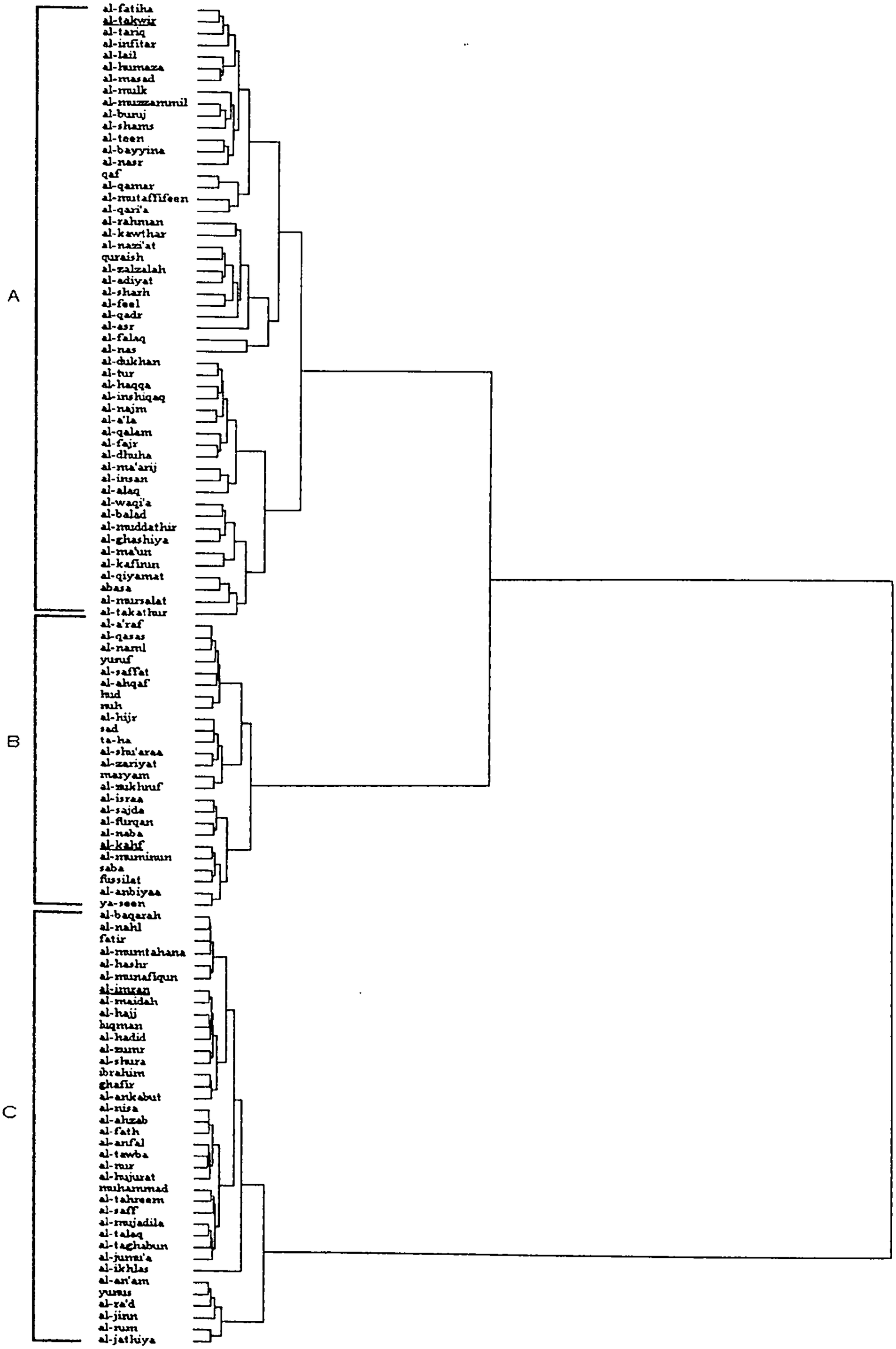


Figure 90: Q3, squared Euclidean distance, Ward's method

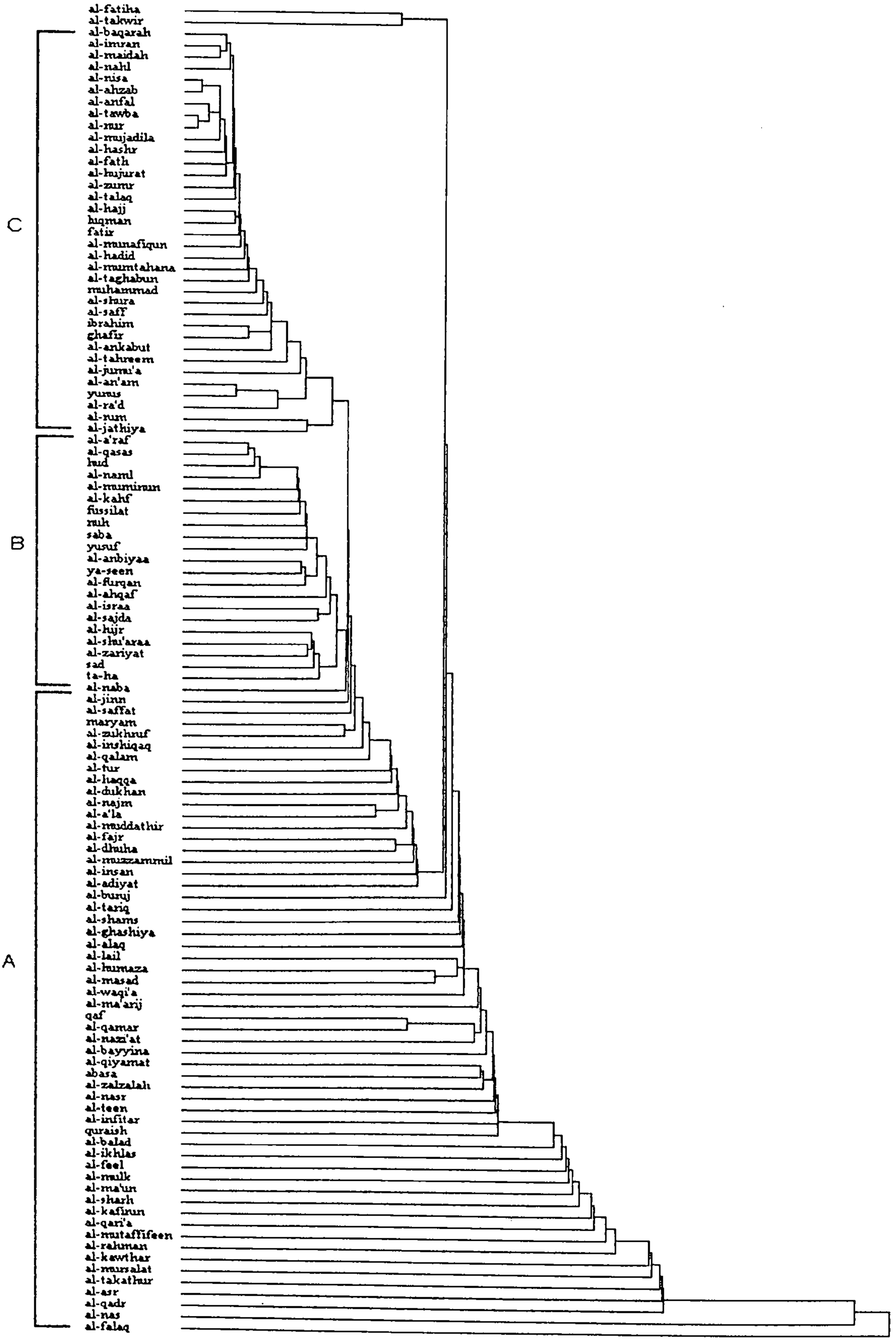


Figure 91: Q3, squared Euclidean distance, single link

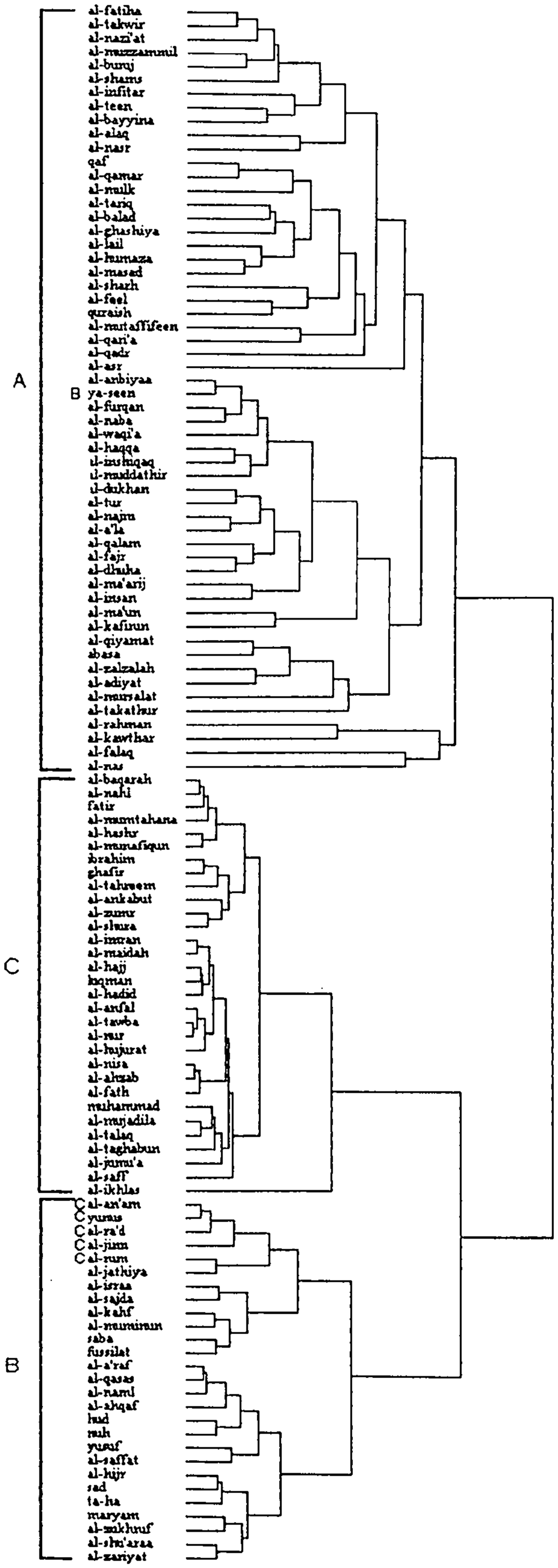


Figure 92: Q3, squared Euclidean distance, complete link

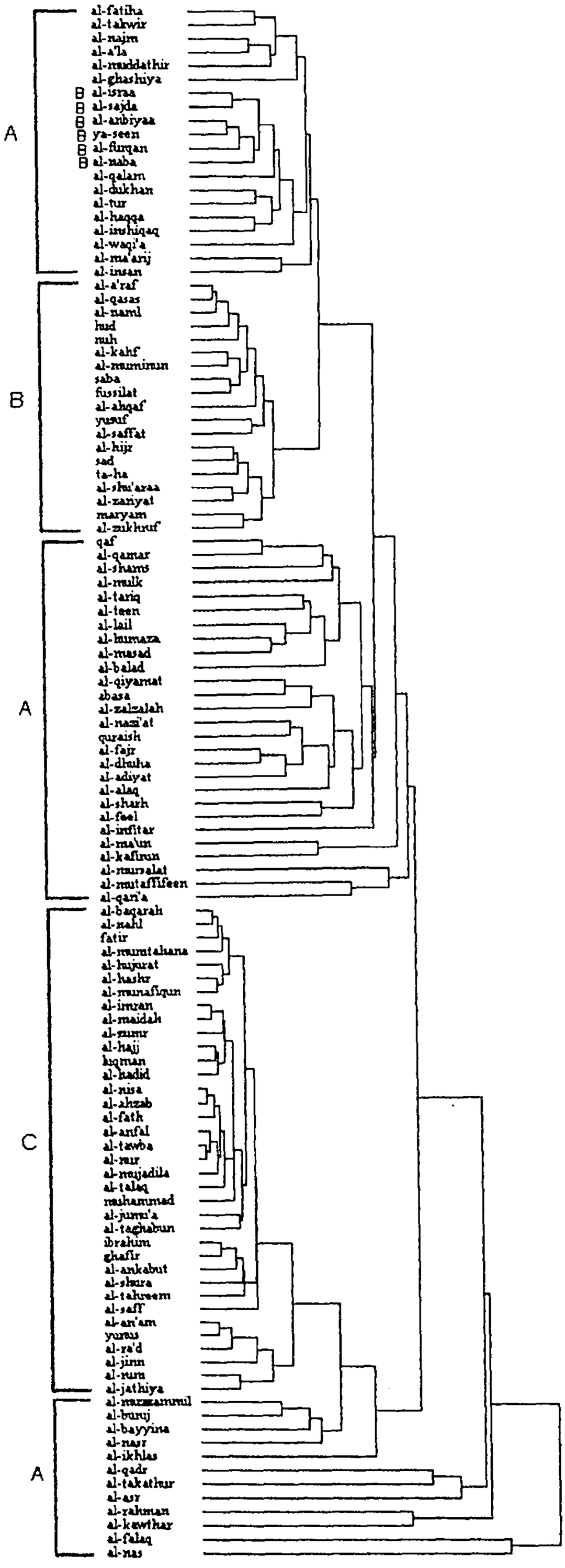


Figure 93: Q3, squared Euclidean distance, average link

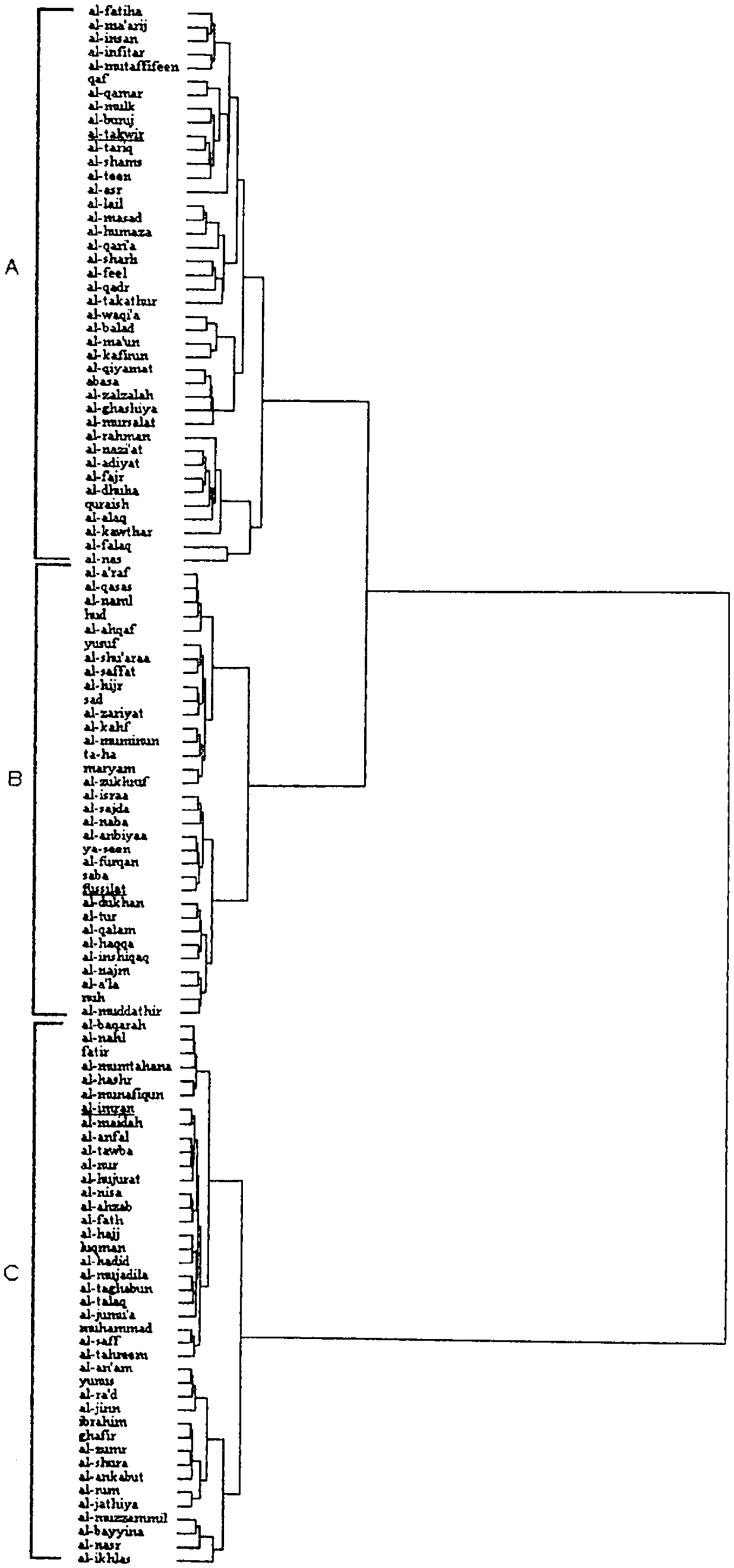


Figure 94: Q4, squared Euclidean distance, Ward's method

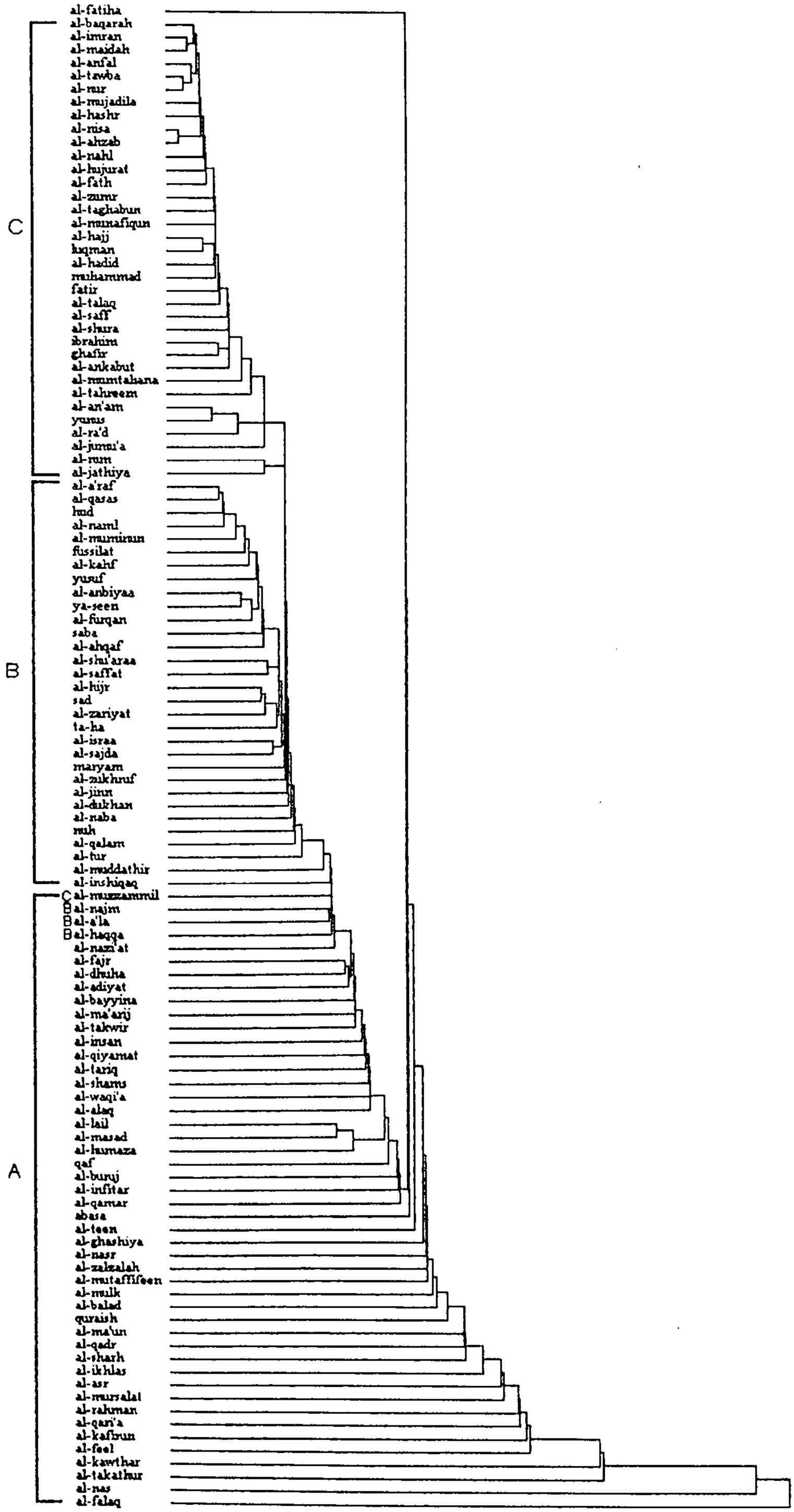


Figure 95: Q4, squared Euclidean distance, single link

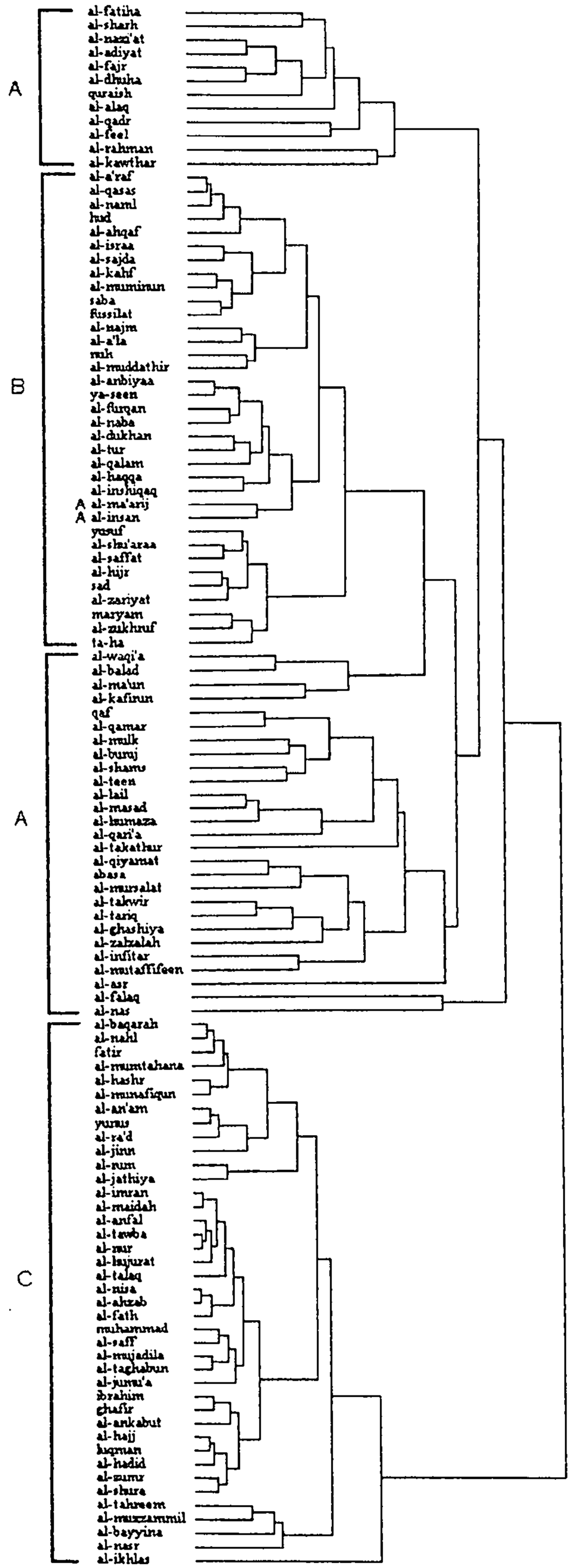


Figure 96: Q4, squared Euclidean distance, complete link

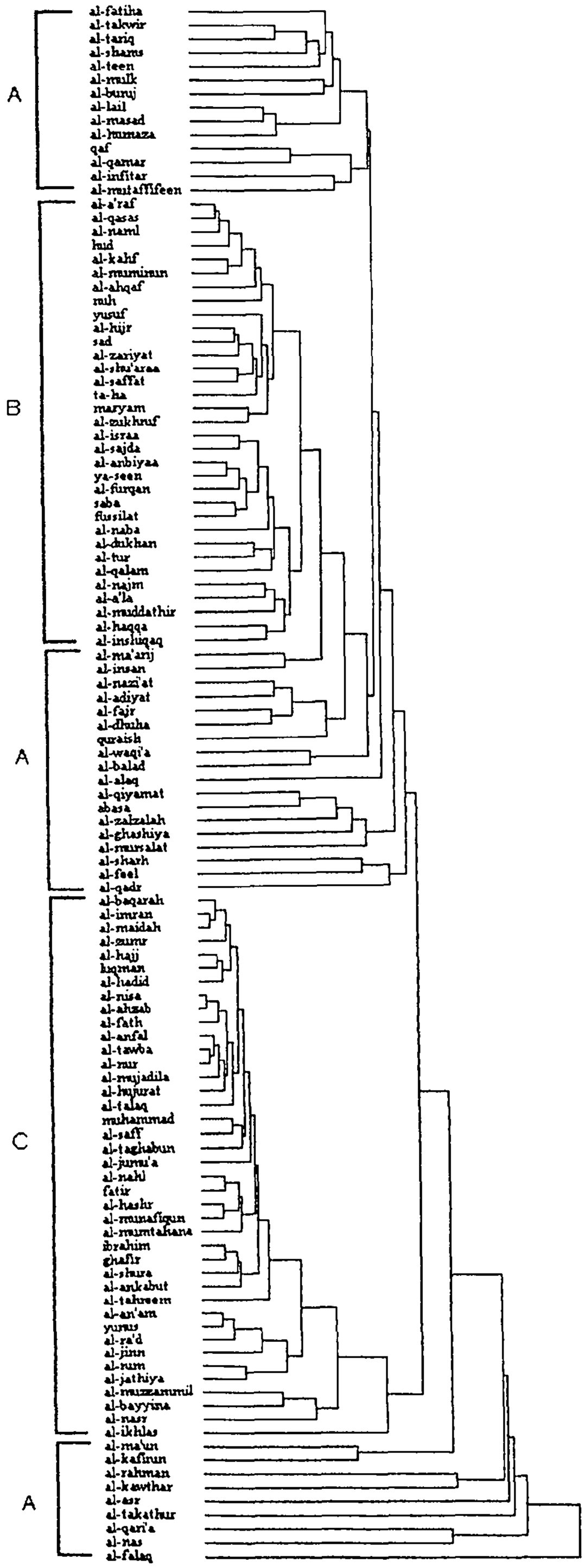


Figure 97: Q4, squared Euclidean distance, average link

6.2.1.2.2 Summary

All the clustering methods across Q1 – Q4 agree in partitioning the majority of the *sura* matrix rows into three discrete groups, labelled A, B, and C in the foregoing tree diagrams: 72 of the 114 *suras* belong to these groups. Of the remaining 41 *suras*, 21 are assigned sometimes to A and sometimes to B, a further 15 sometimes to B and sometimes to C, and 3 sometimes to A and sometimes to C. The remaining 3 *suras* are spread across A, B, and C. This distribution is summarized in Table 23 below:

Sura	***Sura no***	**A**	**B**	**C**
Abasa	80	x		
Al-Adiyat	100	x		
Al-Alaq	96	x		
Al-Asr	103	x		
Al-Balad	90	x		
Al-Dhuha	93	x		
Al-Fajr	89	x		
Al-Falaq	113	x		
Al-Feel	105	x		
Al-Ghashiya	88	x		
Al-Humaza	104	x		
Al-Kawthar	108	x		
Al-Lail	92	x		
Al-Masad	111	x		
Al-Mulk	67	x		
Al-Mursalat	77	x		
Al-Mutaffifeen	83	x		
Al-Nas	114	x		
Al-Nazi'at	79	x		
Al-Qadr	97	x		
Al-Qamar	54	x		

Al-Qari'a	101	x		
Al-Qiyamat	75	x		
Al-Rahman	55	x		
Al-Shams	91	x		
Al-Sharh	94	x		
Al-Takathur	102	x		
Al-Takwir	81	x		
Al-Tariq	86	x		
Al-Teen	95	x		
Al-Zalzalah	99	x		
Qaf	50	x		
Quraish	106	x		
Al-Ahqaf	46		x	
Al-Hijr	15		x	
Al-Saffat	37		x	
Al-Shu'araa	26		x	
Al-Zariyat	51		x	
Al-Zukhruf	43		x	
Fussilat	41		x	
Maryam	19		x	
Nuh	71		x	
Saba	34		x	
Sad	38		x	
Ta-Ha	20		x	
Yusuf	12		x	
Al-Ahzab	33			x
Al-Anfal	8			x
Al-Ankabut	29			x
Al-Baqarah	2			x
Al-Fath	48			x
Al-Hadid	57			x
Al-Hajj	22			x

Al-Hashr	59			x
Al-Hujurat	49			x
Al-Imran	3			x
Al-Maidah	5			x
Al-Mujadila	58			x
Al-Mumtahana	60			x
Al-Munafiqun	63			x
Al-Nahl	16			x
Al-Nisa	4			x
Al-Nur	24			x
Al-Saff	61			x
Al-Taghabun	64			x
Al-Talaq	65			x
Al-Tawba	9			x
Al-Zumr	39			x
Fatir	35			x
Ghafir	40			x
Ibrahim	14			x
Luqman	31			x
Al-A'la	87	x	x	
Al-Anbiyaa	21	x	x	
Al-Dukhan	44	x	x	
Al-Fatiha	1	x	x	
Al-Furqan	25	x	x	
Al-Haqqa	69	x	x	
Al-Infitar	82	x	x	
Al-Insan	76	x	x	
Al-Inshiqaq	84	x	x	
Al-Israa	17	x	x	
Al-Kafirun	109	x	x	
Al-Ma'arij	70	x	x	
Al-Ma'un	107	x	x	

Al-Muddathir	74	x	x	
Al-Naba	78	x	x	
Al-Najm	53	x	x	
Al-Qalam	68	x	x	
Al-Sajda	32	x	x	
Al-Tur	52	x	x	
Al-Waqi'a	56	x	x	
Ya-Seen	36	x	x	
Al-An'am	6		x	x
Al-A'raf	7		x	x
Al-Jathiya	45		x	x
Al-Jumu'a	62		x	x
Al-Kahf	18		x	x
Al-Muminun	23		x	x
Al-Naml	27		x	x
Al-Qasas	28		x	x
Al-Ra'd	13		x	x
Al-Rum	30		x	x
Al-Shura	42		x	x
Al-Tahreem	66		x	x
Hud	11		x	x
Mohammed	47		x	x
Yunus	10		x	x
Al-Bayyina	98	x		x
Al-Buruj	85	x		x
Al-Nasr	110	x		x
Al-Ikhlas	112	x	x	x
Al-Jinn	72	x	x	x
Al-Muzzammil	73	x	x	x

Table 23: Distribution of *suras* across clusters A, B, C

The 16 hierarchical analyses therefore agree that the *suras* fall into three main clusters, and that each of a substantial majority of the *suras* belongs to one and only one of these clusters. They differ, however, in how they assign the remaining *suras* to the main clusters. For ease of reference in subsequent discussion, the three main clusters and their membership are designated A, B and C, and the remainder as AB, BC, AC, and ABC in accordance with how the *suras* in question are distributed by the 16 analyses across the main clusters.

6.2.2 SOM cluster analysis

This section analyzes the Q1 - Q4 data matrices using a SOM.

6.2.2.1 SOM parameters

As noted earlier, SOMs require a range of parameters to be user-specified. Numerous combinations were tried for the analyses that follow, and, except for very small output lattices which gave insufficient space for spatial distribution of the 114 data points to be shown clearly, all gave very similar results; this was expected, since the SOM is known from numerous empirical results to be fairly insensitive to all but extremes of parameter selection, as noted earlier. The selected parameters were:

- Various map dimensions were tried, from 6 x 6. to 50 x 50. At the higher dimensionalities an effect analogous to overfitting in statistical regression occurred: map resolution was so detailed that it was difficult to see any clusters in the midst of the detail. The lowest resolutions, on the other hand, crowded the *suras* together to such an extent that it was, again, difficult to make out any clusters. The optimal size was found to be 12 x 12, which gave sufficient scope for separation among

suras on the surface of the map, and at the same time prevented 'overfitting', thus allowing clusters to emerge.

- Neighbourhood shape was square, as in the foregoing discussion of SOM architecture. Hexagonal neighborhoods were tried, but made no discernible difference to the results; see comments in Ultsch & Herrmann (2005), and compare to Kohonen (2001, 259).
- Neighbourhood decrement function was linear, initially 6, then monotonically decreasing in steps of 1 every 1000 iterations. On neighbourhood function issues see Kohonen (2001, 145-6).
- Learning rate decrement function was linear, initially 0.5, then monotonically decreasing in steps of 0.01 every 100 iterations. On learning rate issues see Kohonen (2001, 143-5).
- Connection initialization was random in range 0.01 .. 0.2. Random initialization is not necessarily the best strategy, and various approaches have been proposed that bring the initial connection vectors closer to the data vectors, thereby predisposing the SOM to learn the data more easily (see discussion in Kohonen 2001, 142-3), but in the present instance random initialization gave consistent convergence and results in numerous trials, and it was therefore used on account of its simplicity.
- Number of training iterations: 7000
- Training was online rather than 'batch', that is, connections were updated after each data vector input.

6.2.2.2 SOM training

For each of the matrices Q1 - Q4, row vectors were randomly selected and presented as input to the SOM, and the training algorithm described earlier was applied.

6.2.2.3 Map generation

Once training was complete for one of the matrices Q1 - Q4, each row vector was taken in turn from the matrix and presented to the SOM as input. The input vector was propagated through the connections, and the resulting map activation was added to the existing activation. At each presentation, the most active unit in the map was identified, and the corresponding *sura* name attached to it. After all 114 rows were presented, the map was a composite of activations from all 114 vectors, and each of the most-active units had one or more *suras* associated with it. This was the final cluster map.

6.2.2.4 Map display

The foregoing discussion mentioned the problem of knowing how to partition the map into clusters in a non-subjective way, that is, based on some principled criterion of cluster membership. Various options were discussed; the best results came from the U-matrix display method, and it was consequently adopted. U-matrix boundaries were calculated, and the gradients were then interpolated and displayed using Matlab's three-dimensional plotting facility. To retain interpretatively adequate resolution, it was found that colour had to be retained, since gray-scale obscured both gradient details and sometimes labelling. In the colour coding scheme, the extremes of map activation are dark blue and red, with dark blue representing greatest activity, red the least activity, and gradations through lighter blue - yellow - orange representing intermediate activation. The most active regions of the map therefore show up as blue areas which are bounded by lighter blue and / or orange-yellow regions.

For each of the Q1 - Q4 analyses, two map views are presented:

- The U-matrix map oriented two-dimensionally with viewpoint directly above, showing regions of relative map activation by means of the colour coding scheme just described and *sura* name labels indicating the location of each *sura* vector on the map. The *sura* labels are 'anchored' on the left: the unit associated with any given *sura* is located immediately to the left of the first letter of the *sura* name
- A three-dimensional view that combines colour-coding with relative elevation to give a colour-coded activation landscape, the aim being thereby to augment the two-dimensional view. The map is tilted towards the viewer so as to give an informative view of the landscape. Note, however, that the *sura* labels suffer from perspective displacement as a result, and can therefore mislead --they should always be cross-referenced with the two-dimensional map, where relative *sura* placement is accurately represented.

Finally, a brief note on the *sura* name labelling. It sometimes happens that, where two or more vectors are very similar, SOM training will assign them to the same unit, with the result that the Matlab labelling procedure overlaps and thus obscures the names. It is possible to mitigate this effect by staggering the labels horizontally to a relatively small degree, but in some cases this is still insufficient. Further mitigation was provided on the one hand by rotating the map 45 degrees. There are, however, still a few instances were the labels overlap, in which case they were manually separated.

As with the hierarchical analyses, the SOM ones are first presented without comment, and subsequently discussed.

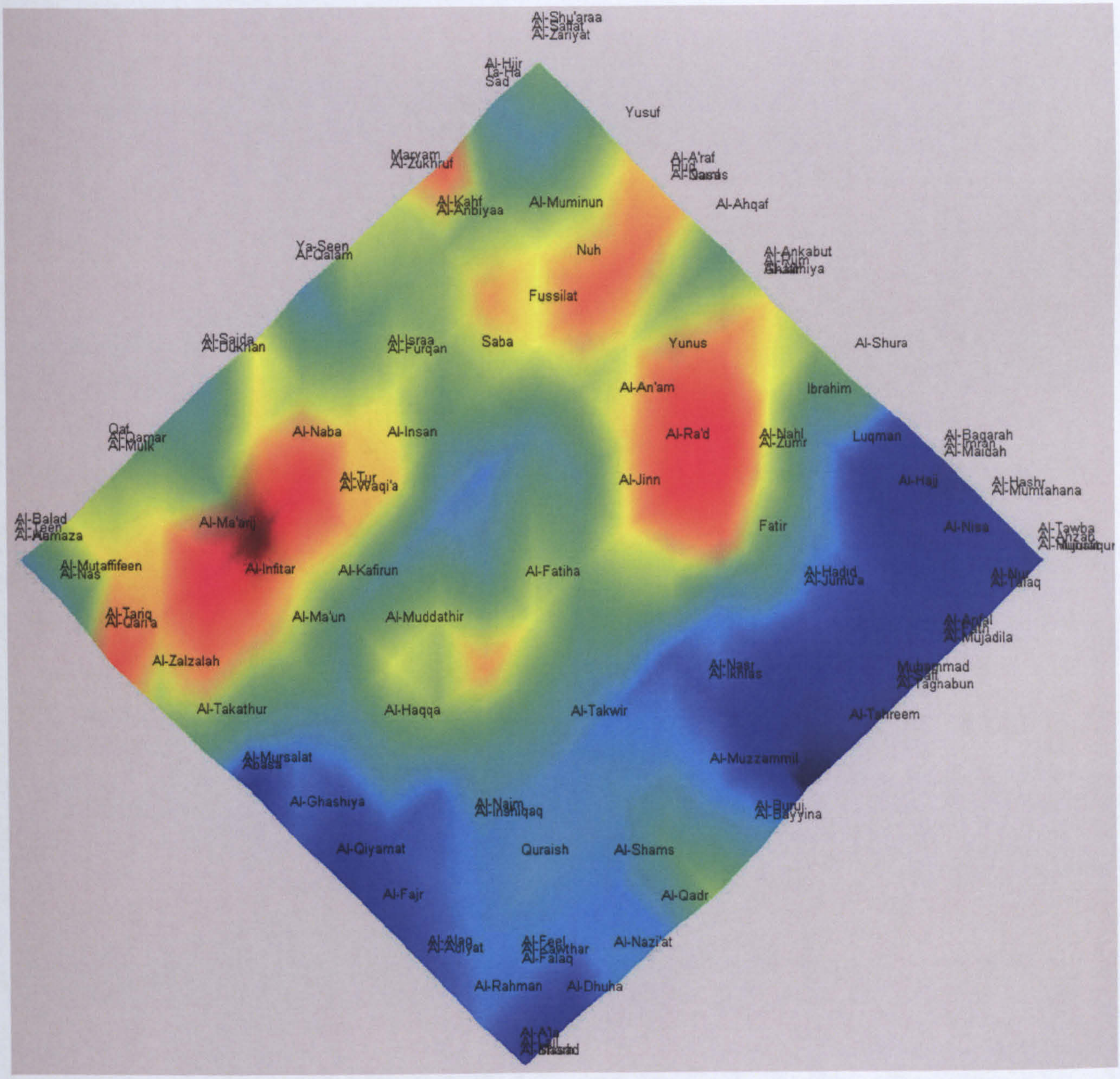


Figure 98: Two-dimensional SOM map of Q1

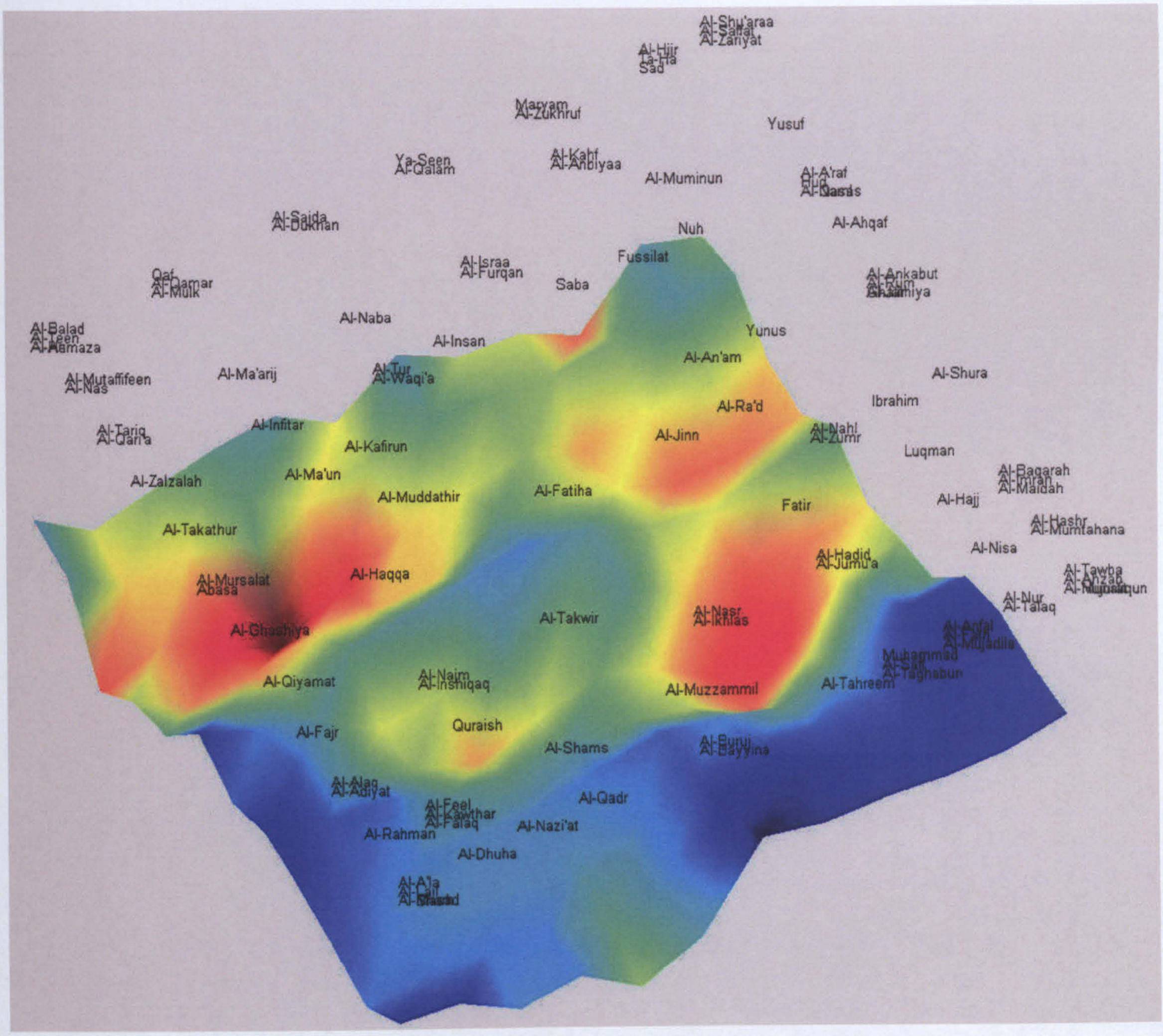


Figure 99: Three-dimensional SOM map of Q1

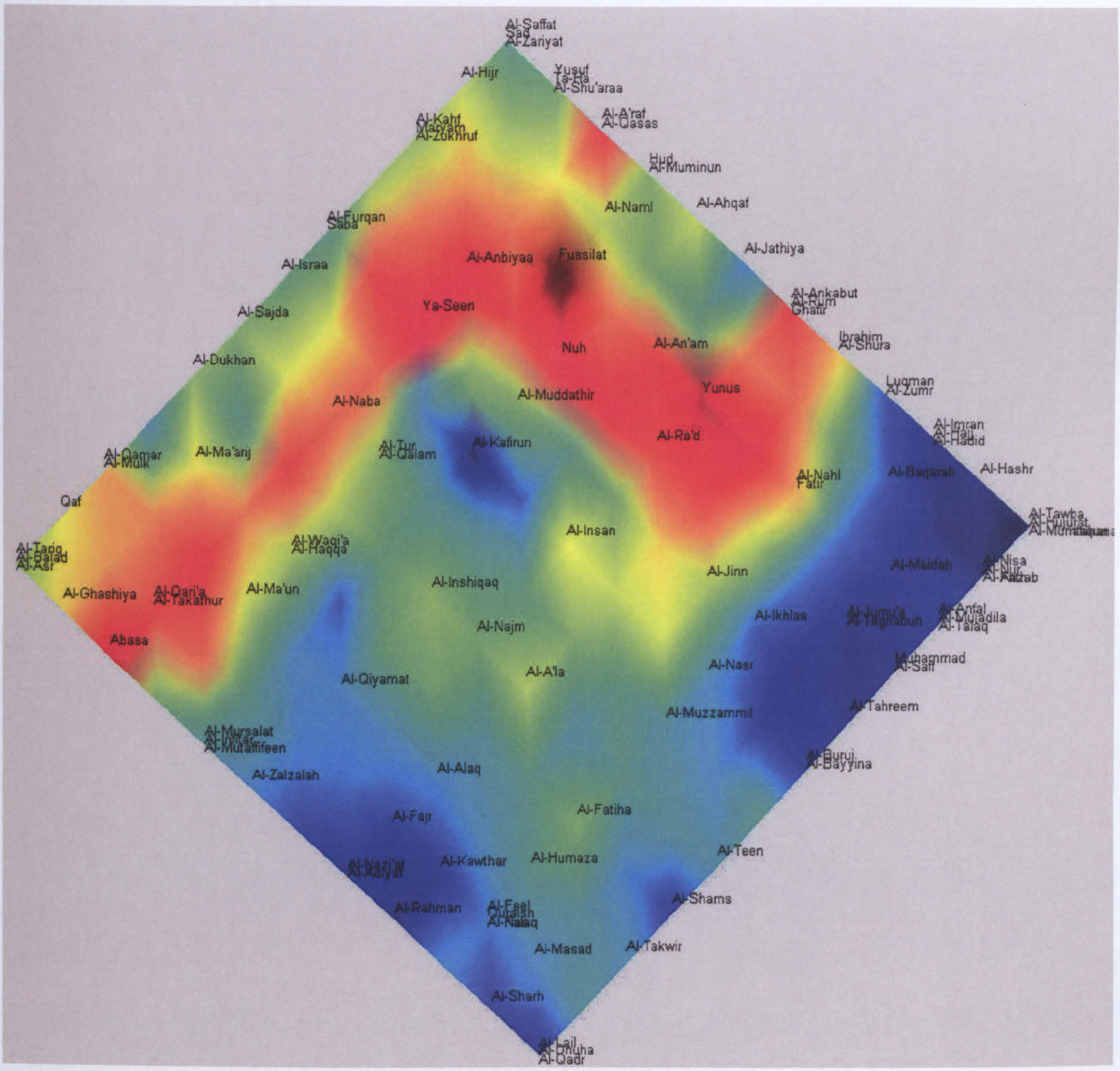


Figure 100: Two-dimensional SOM map of Q2

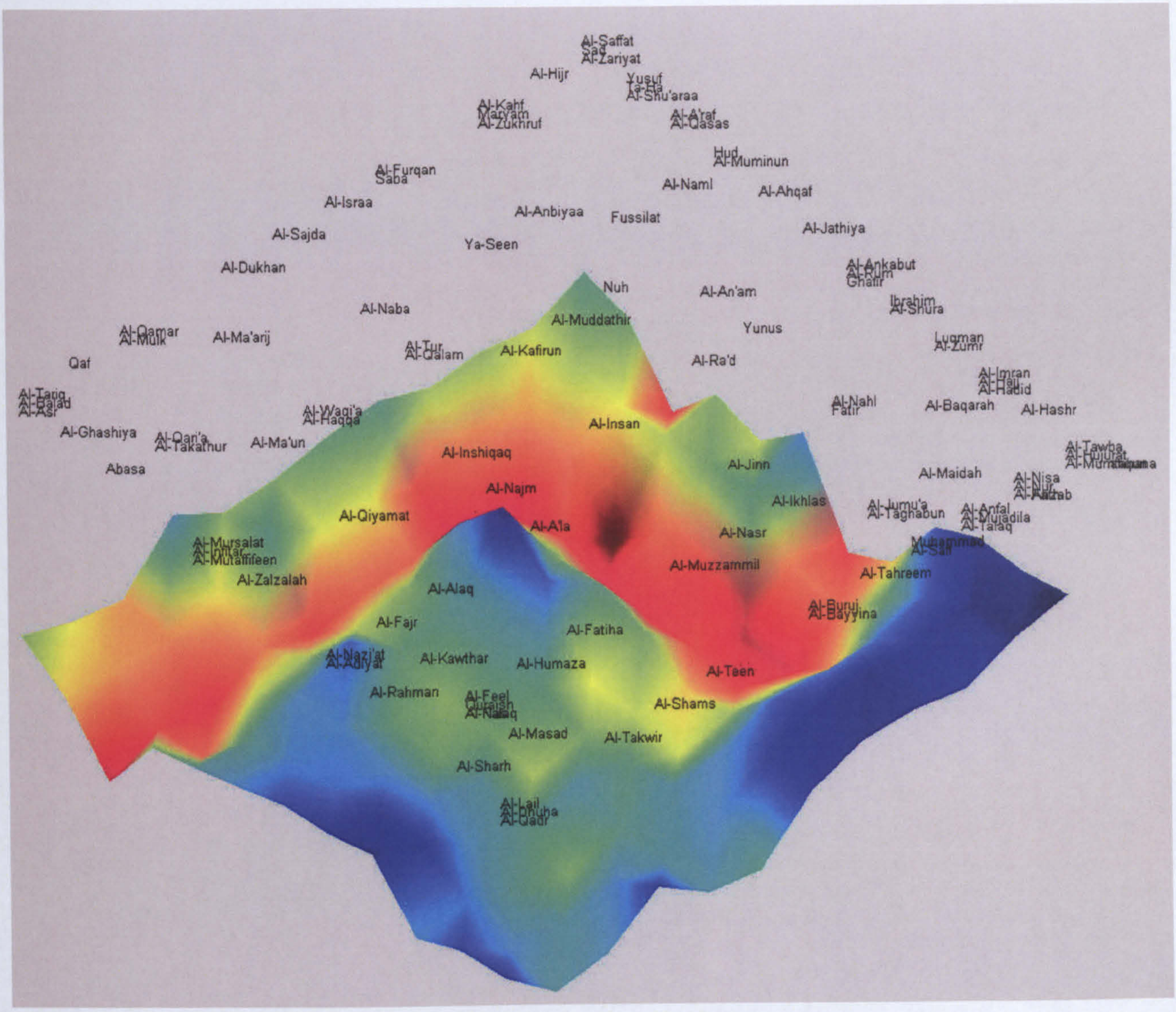


Figure 101: Three-dimensional SOM map of Q2

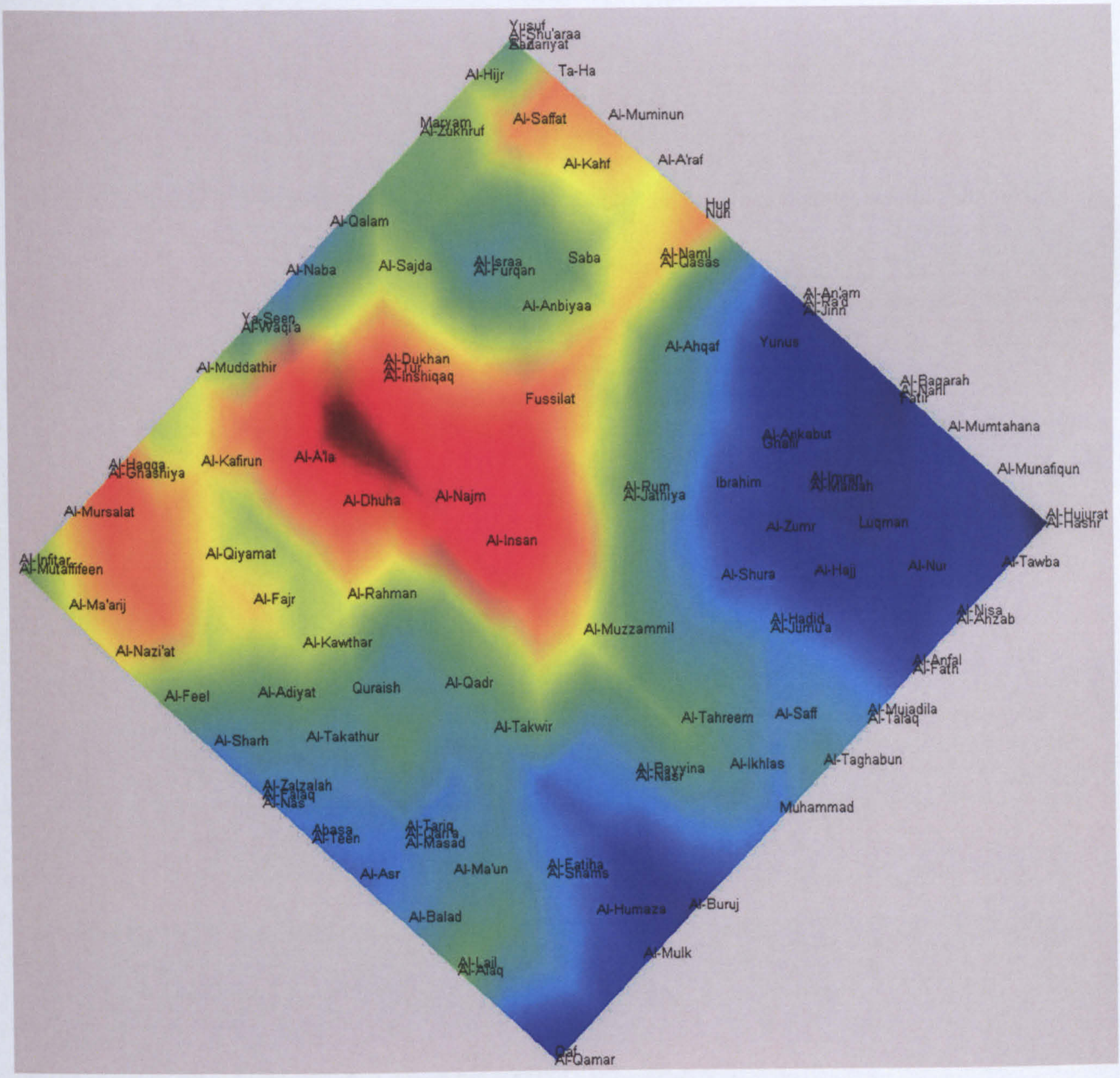


Figure 102: Two-dimensional SOM map of Q3

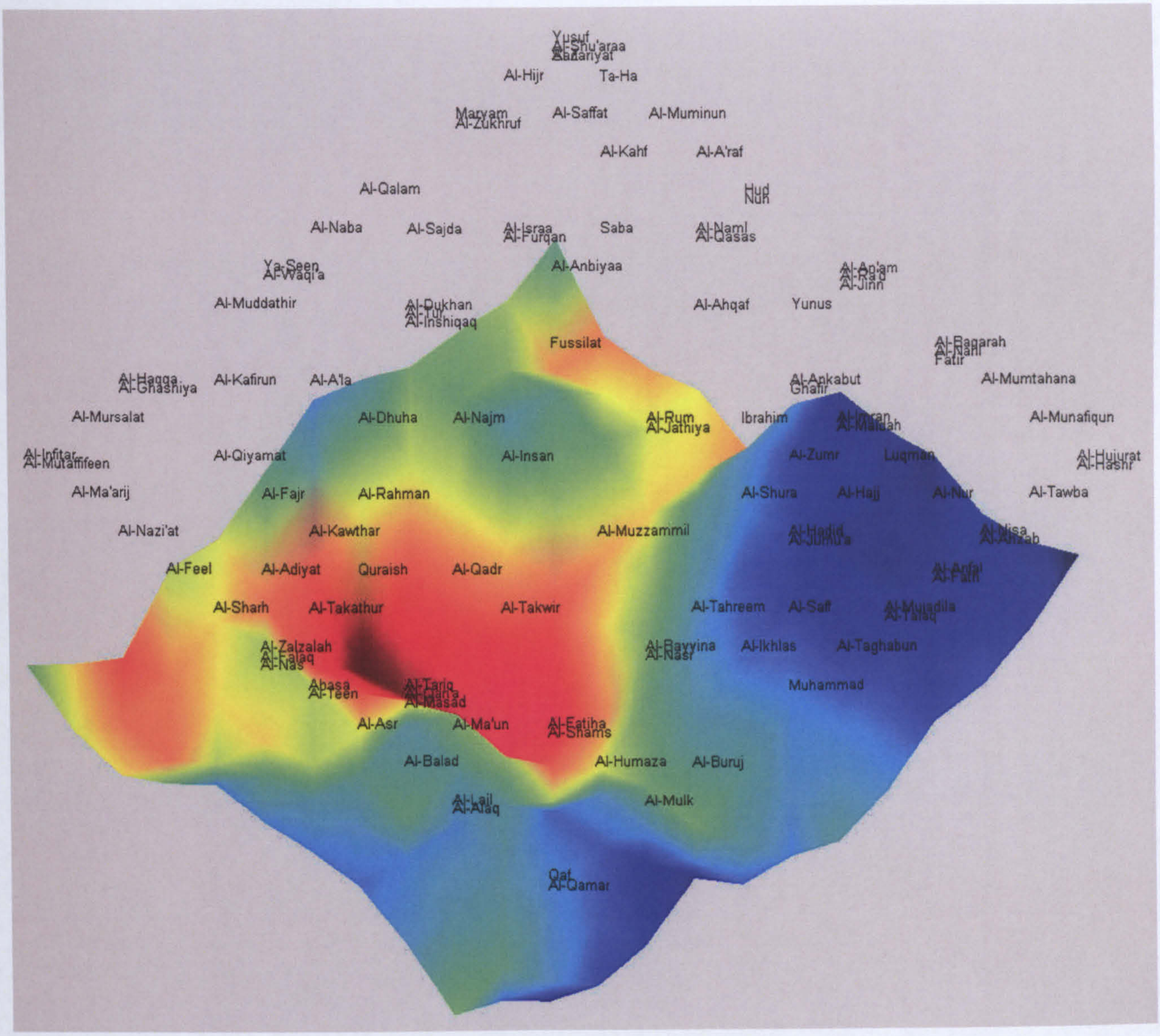


Figure 103: Three-dimensional SOM map of Q3

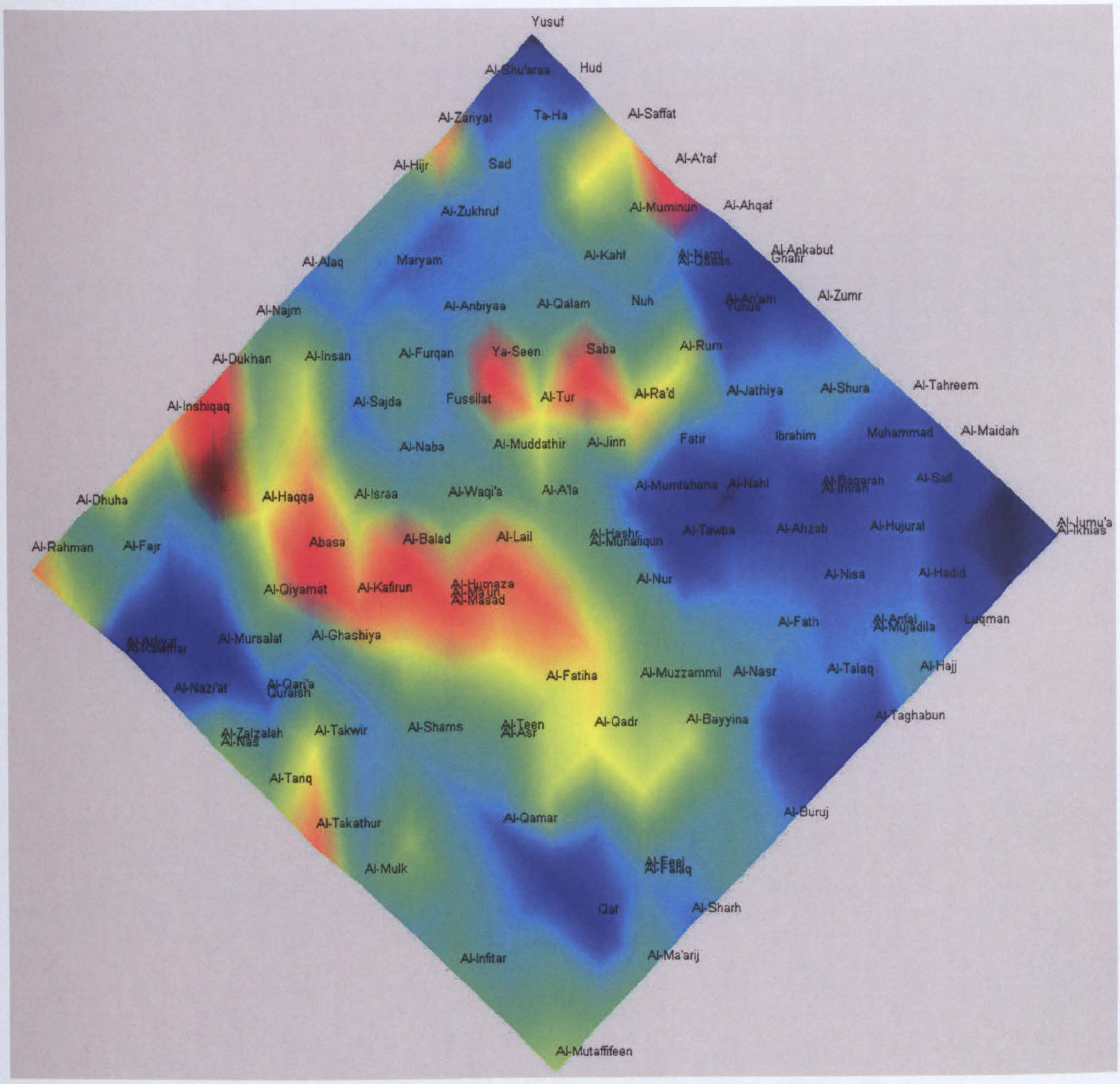


Figure 104: Two-dimensional SOM map of Q4

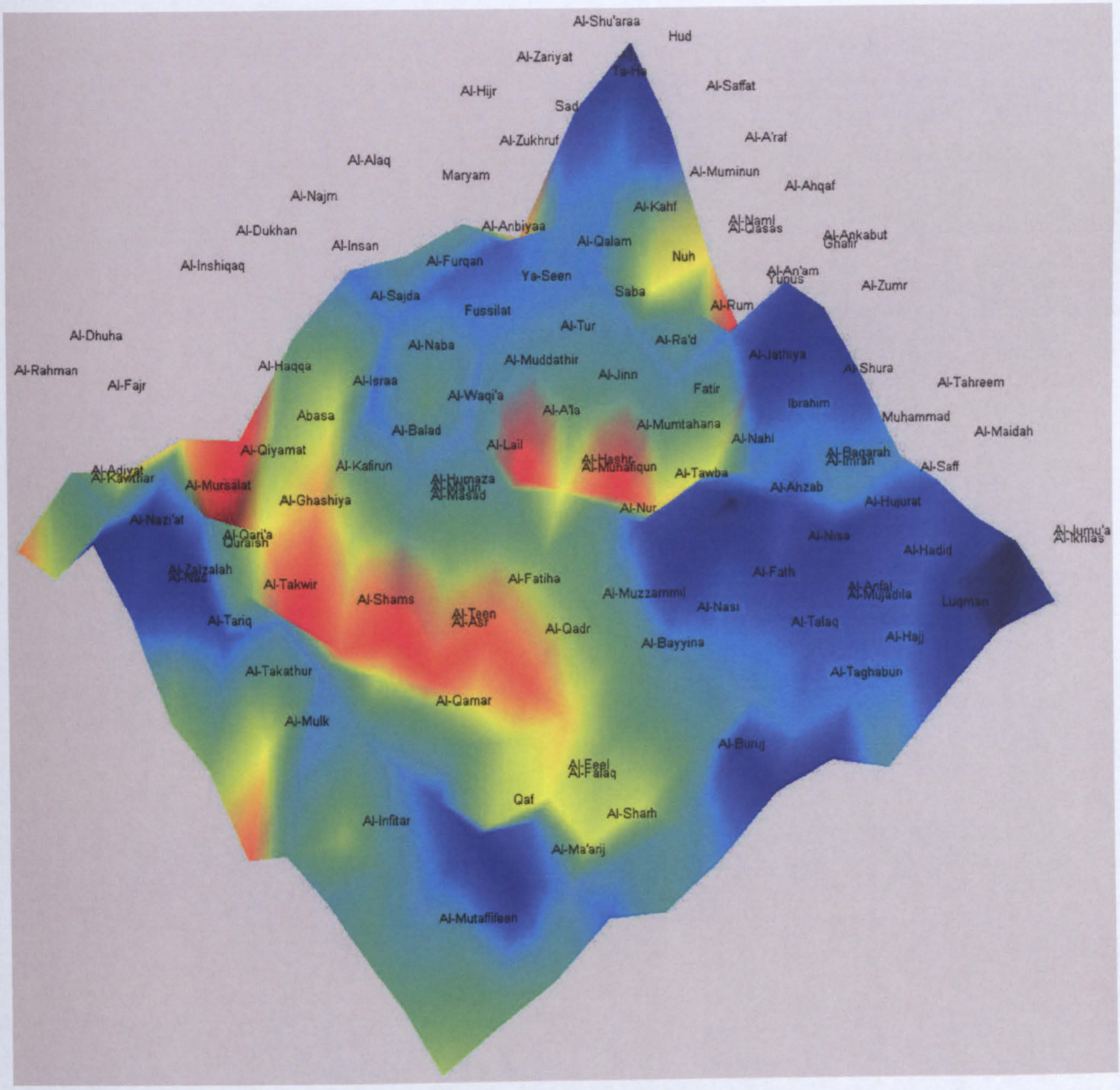


Figure 105: Three-dimensional SOM map of Q4

6.2.2.5 Discussion

In comparing the results for Q1 - Q4, the foregoing discussions of SOM topology preservation and of U-matrix representation have to be kept in mind. Specifically, the blue / high areas of the maps are the regions where the *suras* are topologically close, that is, where they cluster, and the yellow - orange - red / low areas are where they are topologically far apart.

The most consistent feature across all four maps is the blue area at the right-hand corner, which is approximately indicated by curved lines in the reduced versions of the foregoing SOM maps below, and labelled 'Cluster 1':

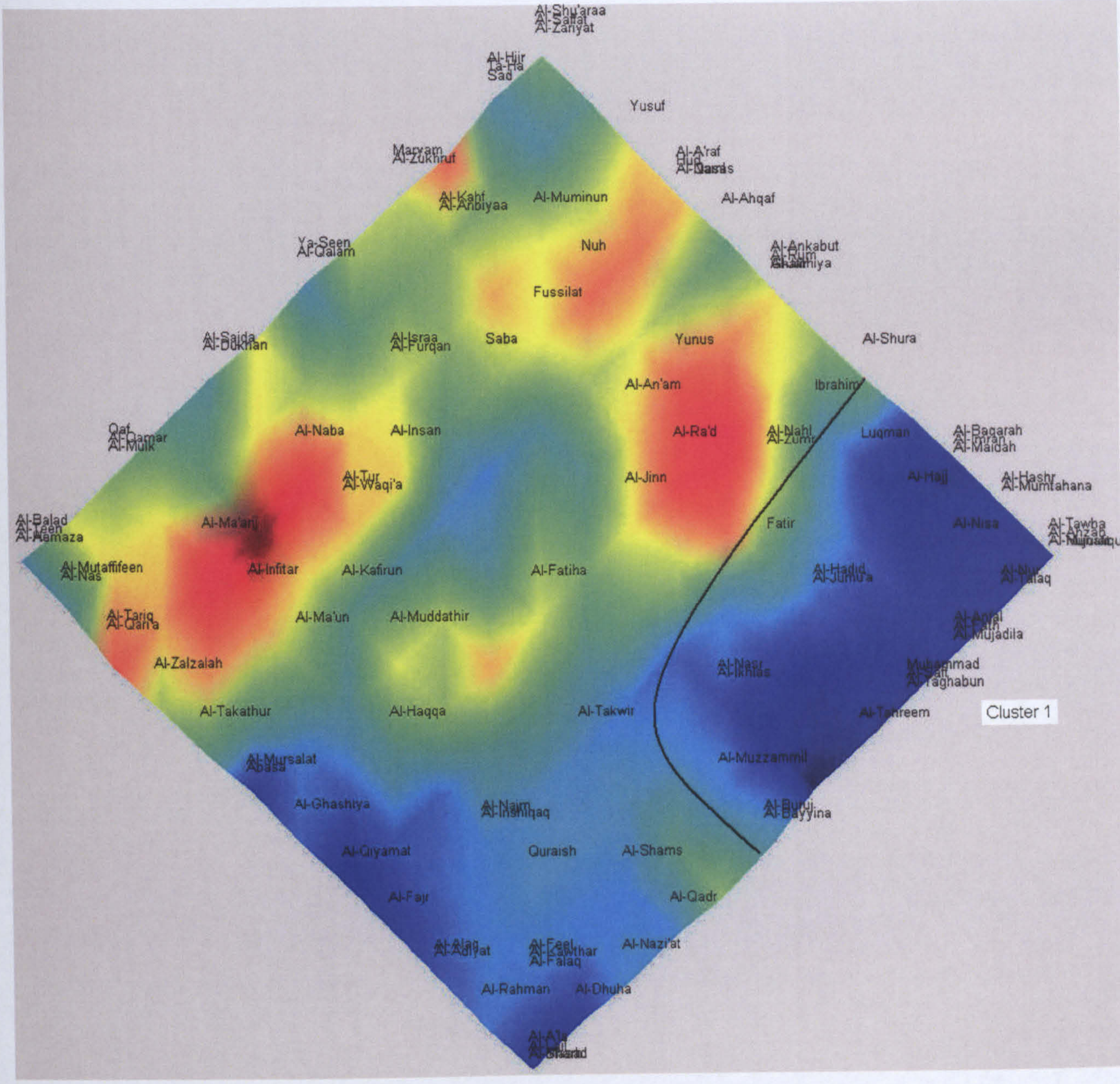


Figure 106: Two-dimensional SOM map of Q1 with cluster 1 shown

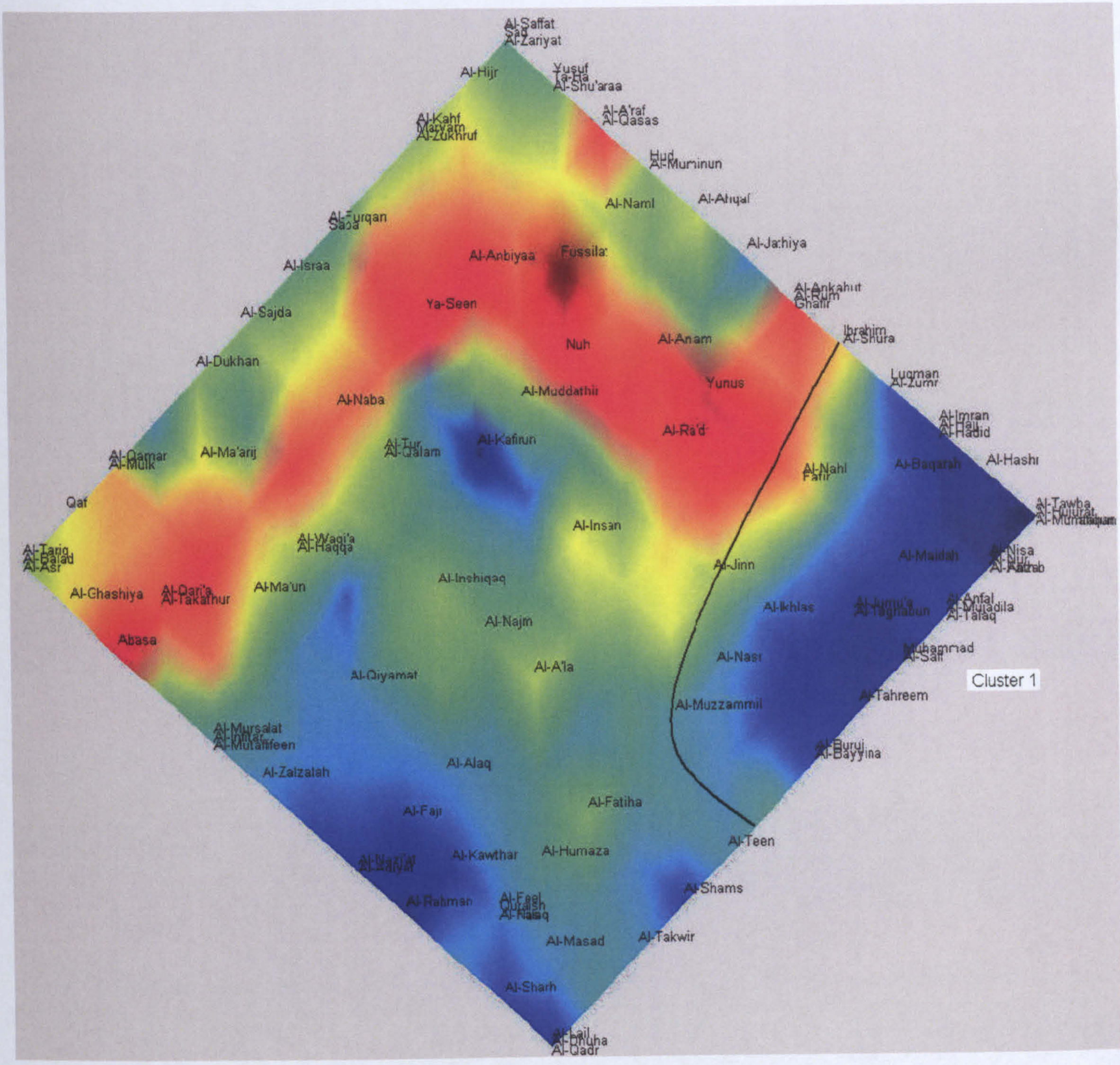


Figure 107: Two-dimensional SOM map of Q2 with cluster 1 shown

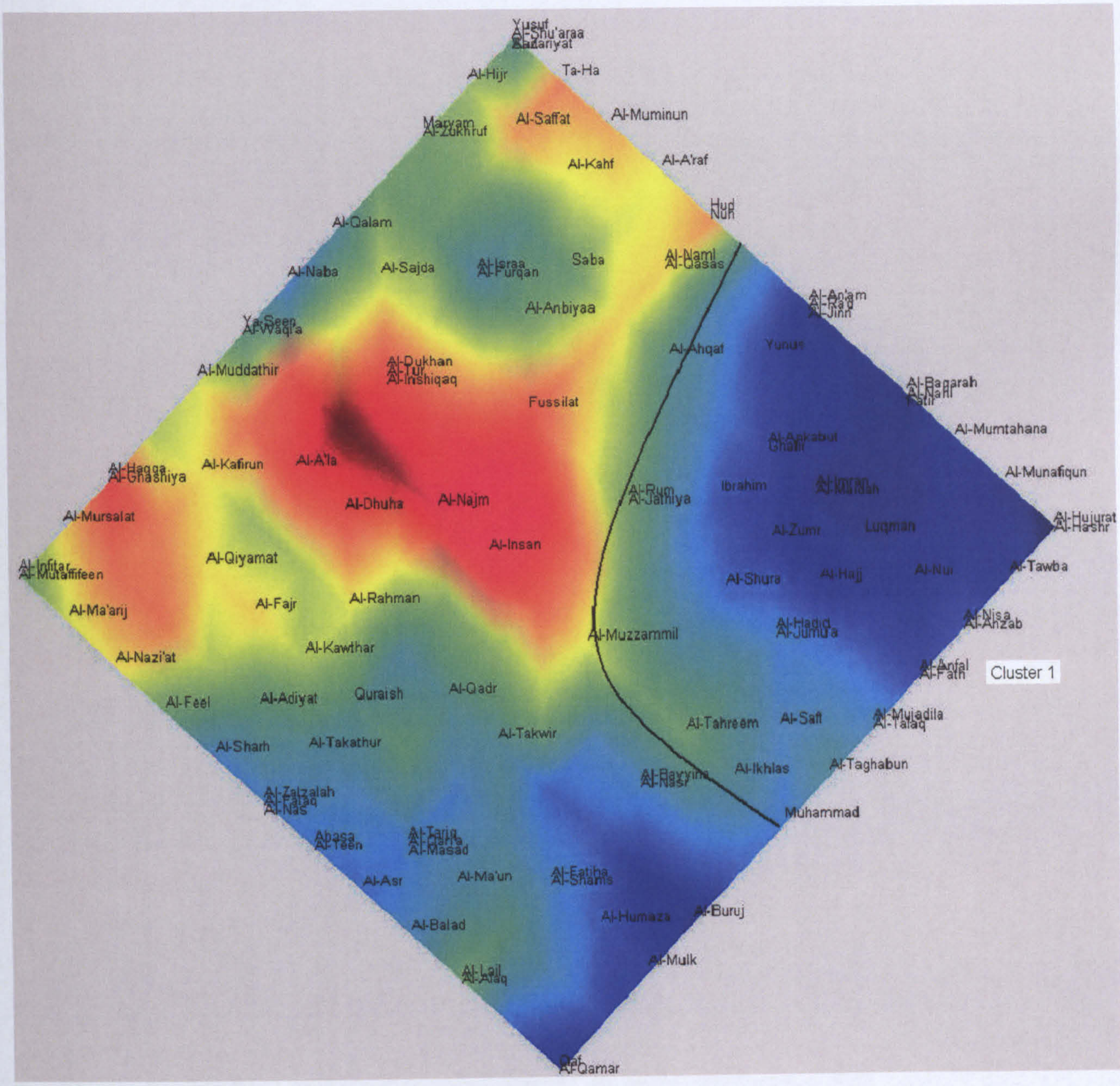


Figure 108: Two-dimensional SOM map of Q3 with cluster 1 shown

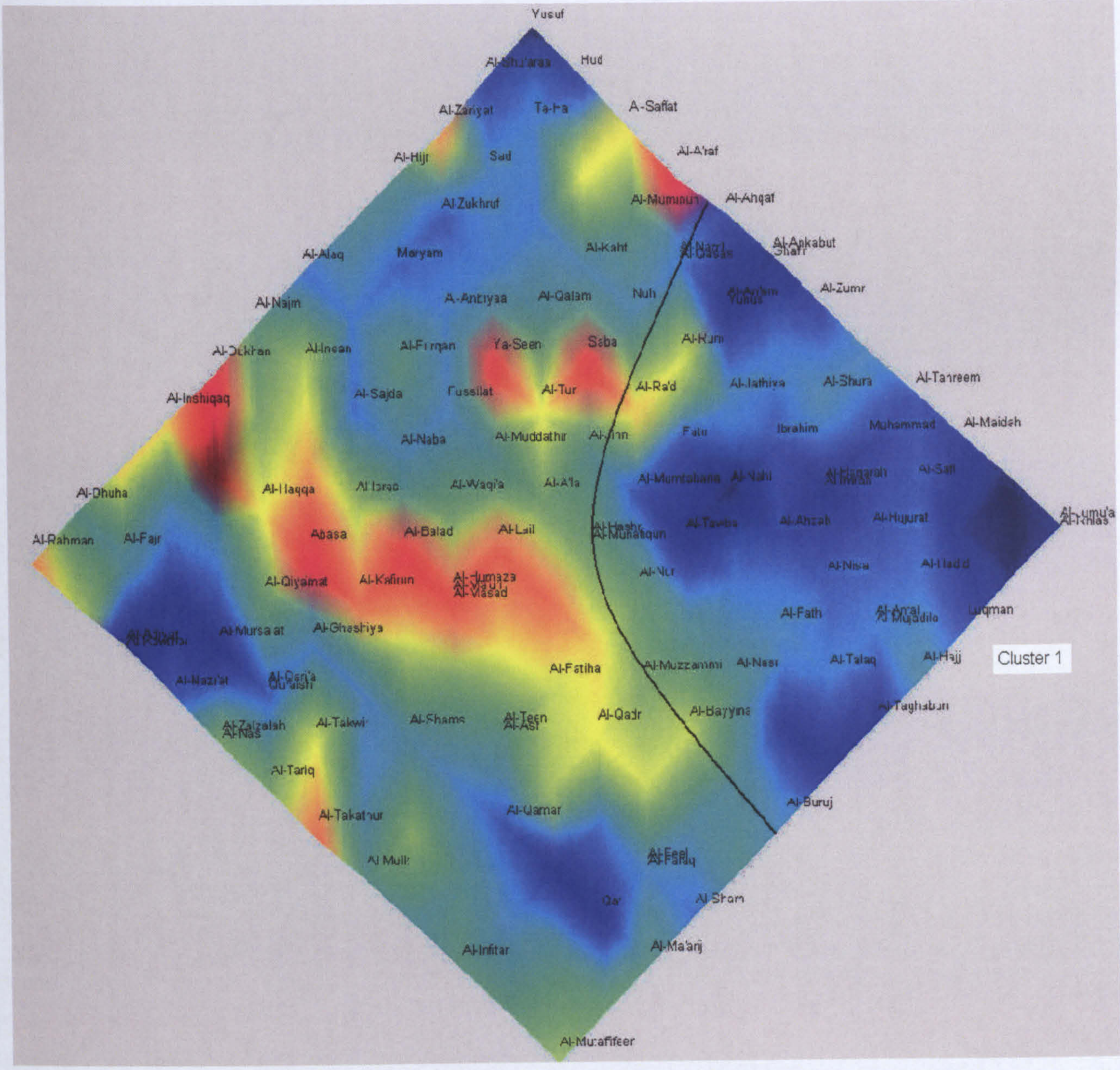


Figure 109: Two-dimensional SOM map of Q4 with cluster 1 shown

The *suras* in this cluster are shown in the table that follows. In relating this table to the maps, recall that (i) *sura* names are 'anchored' on the left, so that the start of any given *sura* name indicates its position on the map, and (ii) that there is unavoidable subjectivity at the boundary of the cluster. The table is divided into two parts: the first shows the *suras* which all the maps are unanimous in assigning to this cluster, and the second the *suras* on which the maps differ. In the first part, the *suras* are listed in alphabetical order for ease of reference in later discussion; in the second they are arranged in descending order of agreement among maps, and subsidiary to that ordering in alphabetical order:

Unanimous cluster membership	**Q1**	**Q2**	**Q3**	**Q4**
Al Ahzab	x	x	x	x
Al Anfal	x	x	x	x
Al Baqarah	x	x	x	x
Al Fath	x	x	x	x
Al Hadid	x	x	x	x
Al Hajj	x	x	x	x
Al Hashr	x	x	x	x
Al Hujurat	x	x	x	x
Al Ikhlas	x	x	x	x
Al Imran	x	x	x	x
Al Jumua	x	x	x	x
Al Maidah	x	x	x	x
Al Mujadila	x	x	x	x
Al Mumtahana	x	x	x	x
Al Mundafiqun	x	x	x	x
Al Nahl	x	x	x	x
Al Nisa	x	x	x	x
Al Nur	x	x	x	x
Al Saff	x	x	x	x
Al Taghabun	x	x	x	x

Al Tahreem	x	x	x	x
Al Talaq	x	x	x	x
Al Tawba	x	x	x	x
Al Zumr	x	x	x	x
Fatir	x	x	x	x
Ibrahim	x	x	x	x
Lugman	x	x	x	x
Mohammed	x	x	x	x
Three out of four				
Al Buruj	x	x		x
Al Muzzammil	x	x		x
Al Nasr	x	x		x
Al Shura		x	x	x
Two out of four				
Al Bayyina	x	x		
Al Ahqaf			x	x
Al Anam			x	x
Al Ankabut			x	x
Ghafir			x	x
Yunus			x	x
One only				
Al Jinn			x	
Al Rad			x	
Al Jathiya				x
Al Naml				x
Al Qasas				x
Al Rum				x

Table 24: *Sura* membership of cluster 1

Of the *suras* in Table 24 for which Q1-Q4 are not unanimous with respect to cluster membership, the majority are either immediately adjacent in the boundary region of the cluster, or nearby in an immediately adjacent cluster:

Al Ankabut	Al Muzzammil
Al Bayyina	Al Nasr
Al Buruj	Al Rum
Al Jathiya	Al Shura
Al Jinn	Ghafir

Table 25: *Suras* adjacent to cluster 1

Only the following, relatively few *suras* in Table 24 for which Q1-Q4 are not unanimous with respect to cluster membership occur in parts of the map distant from the main cluster:

Al Ahqaf	Al Qasa
Al Anam	Al Rad
Al Naml	Yunus

Table 26: *Suras* distant from cluster 1

Slightly less consistent, but still in evidence across all four maps, is the blue area at the lower left. As before, this is approximately indicated by curved lines in the maps that follow, and labelled 'Cluster 2':

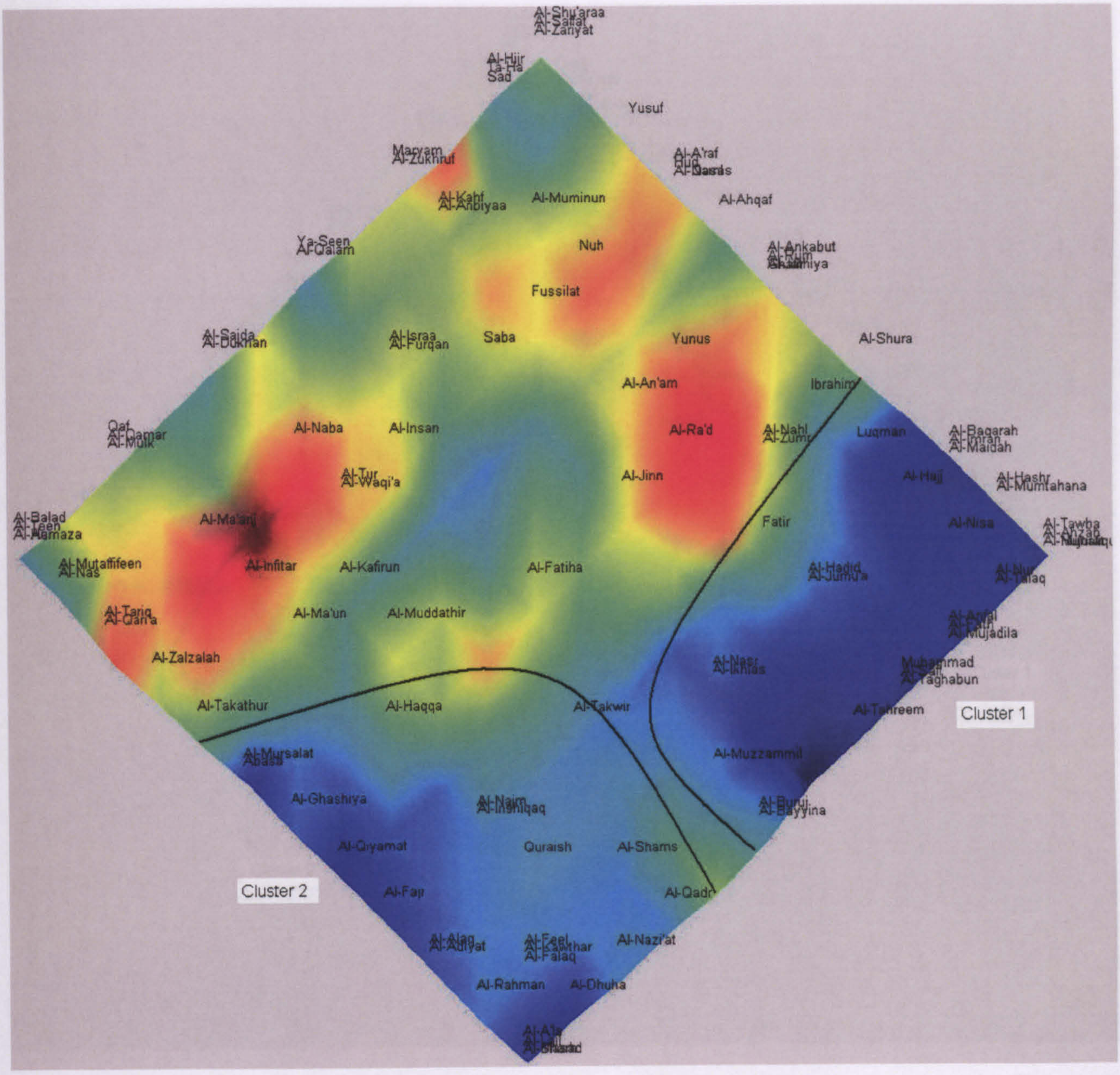


Figure 110: Two-dimensional SOM map of Q1 with clusters 1 and 2 shown

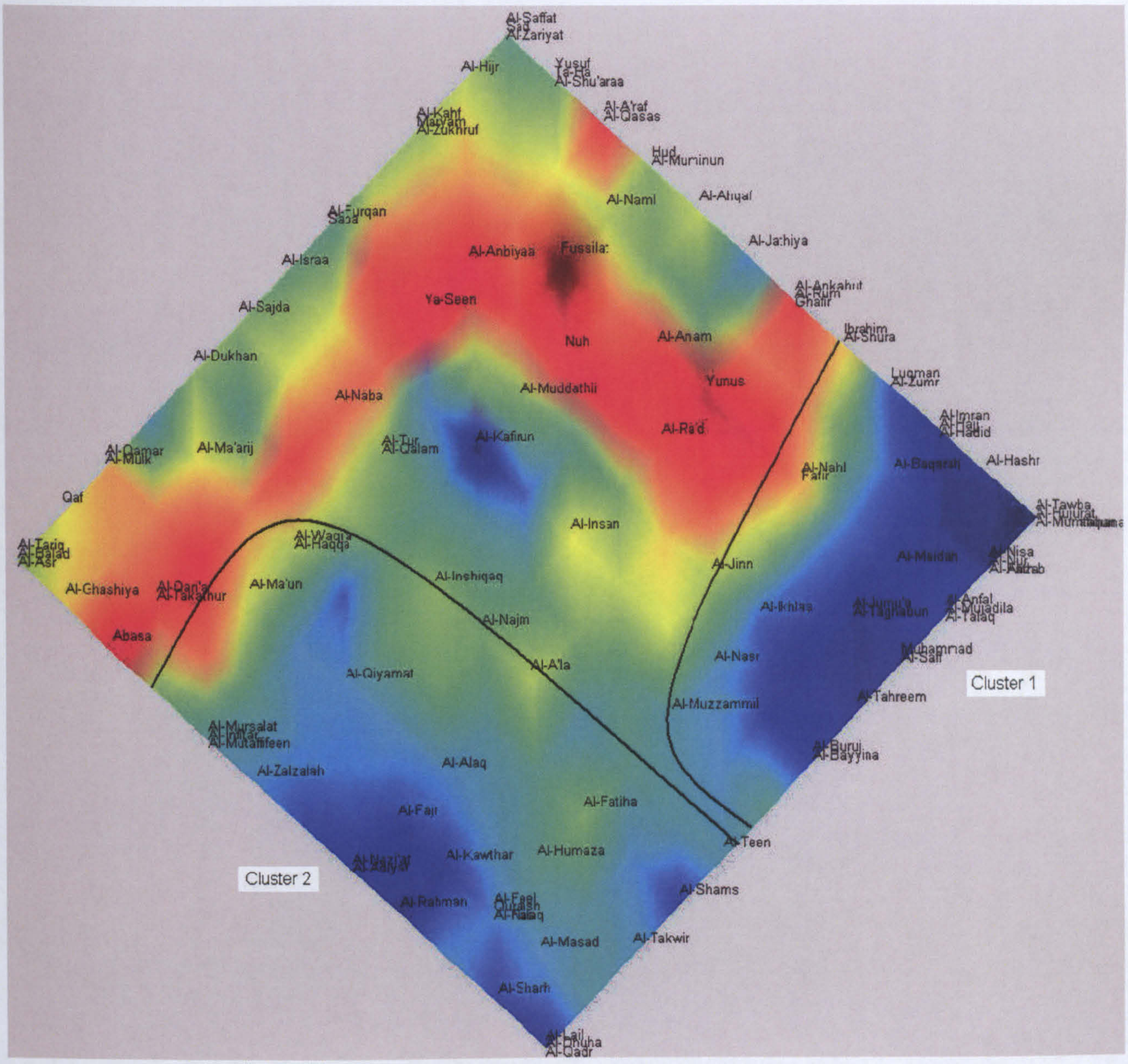


Figure 111: Two-dimensional SOM map of Q2 with clusters 1 and 2 shown

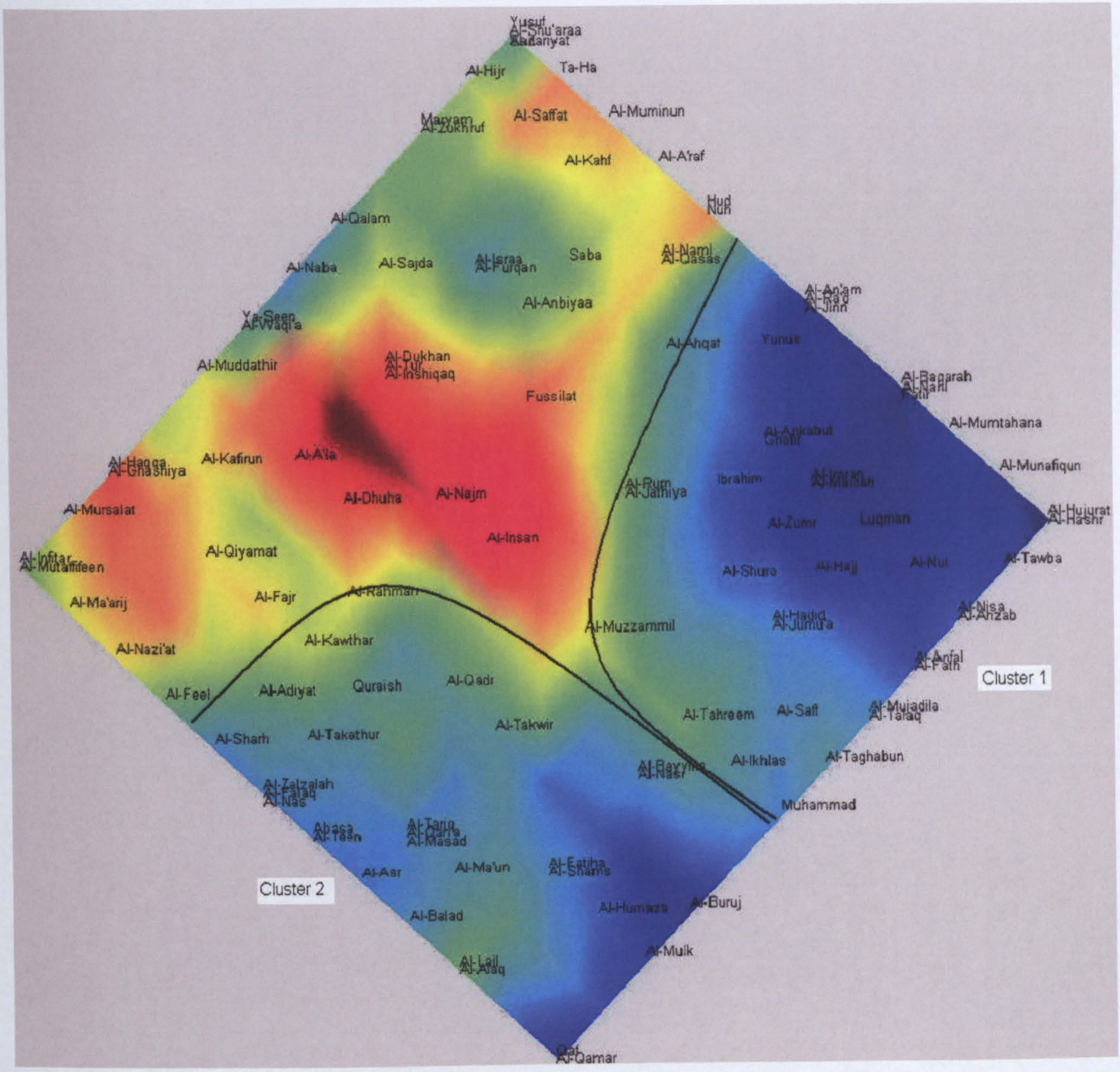


Figure 112: Two-dimensional SOM map of Q3 with clusters 1 and 2 shown

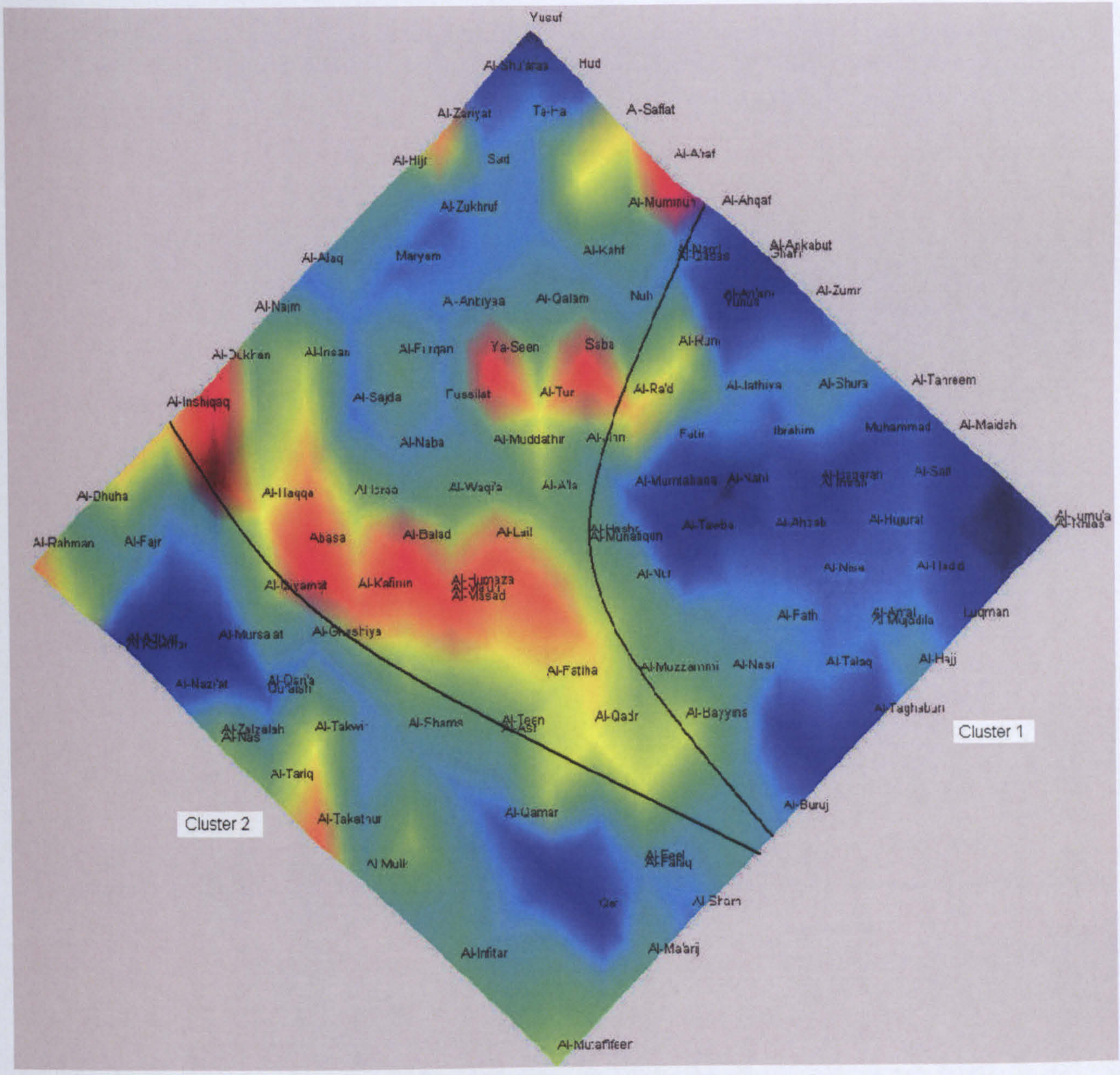


Figure 113: Two-dimensional SOM map of Q4 with clusters 1 and 2 shown

The *suras* in this cluster are shown in the table that follows; the observations made about the table for Cluster 1 apply to this one as well:

Unanimous	**Q1**	**Q2**	**Q3**	**Q4**
Al Adiyat	x	x	x	x
Al Fajr	x	x	x	x
Al Falaq	x	x	x	x
Al Feel	x	x	x	x
Al Kawthar	x	x	x	x
Al Rahman	x	x	x	x
Al Shams	x	x	x	x
Al Sharh	x	x	x	x
Al Takwir	x	x	x	x
Quraish	x	x	x	x
Three out of four				
Al Alaq	x	x	x	
Al Lail	x	x	x	
Al Masad	x	x	x	
Al Qadr	x	x	x	
Al Dhuha	x	x		x
Al Mursalat	x	x		x
Al Naziat	x	x		x
Al Qiyamat	x	x		x
Al Nas		x	x	x
Al Teen		x	x	x
Al Zalzalah		x	x	x
Two out of four				
Al Haqqa	x	x		
Abasa	x		x	
Al Ghashiya	x			x

Al Inshiqaq	x			x
Al Humaza		x	x	
Al Infitar		x		x
Al Mutaffifeen		x		x
Al Asr			x	x
Al Mulk			x	x
Al Qamar			x	x
Al Qaria			x	x
Al Takathur			x	x
Al Tariq			x	x
Qaf			x	x
One only				
Al Ala	x			
Al Najm	x			
Al Waqia		x		
Al Balad			x	
Al Bayyina			x	
Al Buruj			x	
Al Fatiha			x	
Al Maun			x	
Al Nasr			x	
Al Maarij				x

Table 27: *Sura* membership of cluster 2

Of the *suras* in Table 27 for which Q1-Q4 are not unanimous with respect to cluster membership, the majority are either immediately adjacent in the boundary region of the cluster, or nearby in an immediately adjacent cluster:

Abasa	Al Infitar	Al Naziat
Al Asr	Al Inshiqaq	Al Qadr

Al Balad	Al Lail	Al Qaria
Al Bayyina	Al Masad	Al Qiyamat
Al Dhuha	Al Maun	Al Takathur
Al Fatiha	Al Mursalat	Al Tariq
Al Ghashiya	Al Mutaffifeen	Al Teen
Al Haqqa	Al Nas	Al Zalzalah
Al Humaza	Al Nasr	

Table 28: *Suras* adjacent to cluster 2

Only the following, relatively few *suras* in Table 27 for which Q1-Q4 are not unanimous with respect to cluster membership occur in parts of the map distant from the main cluster:

Al Ala	Al Qamar
Al Alaq	Al Waqia
Al Mulk	Qaf
Al Najm	

Table 29: *Suras* distant from cluster 2

Finally, there is a cluster that is less easy to detect visually than the preceding ones, in that it emerges clearly only on Q4. Once one knows what to look for, however, many of the same *suras* as in Q4 are also close to one another in Q1 - Q3, though the colouring indicates that these *suras* are not as topologically close as those in clusters 1 and 2. The proposed boundaries are once again shown by lines, and the cluster ID labelled 'Cluster 3':

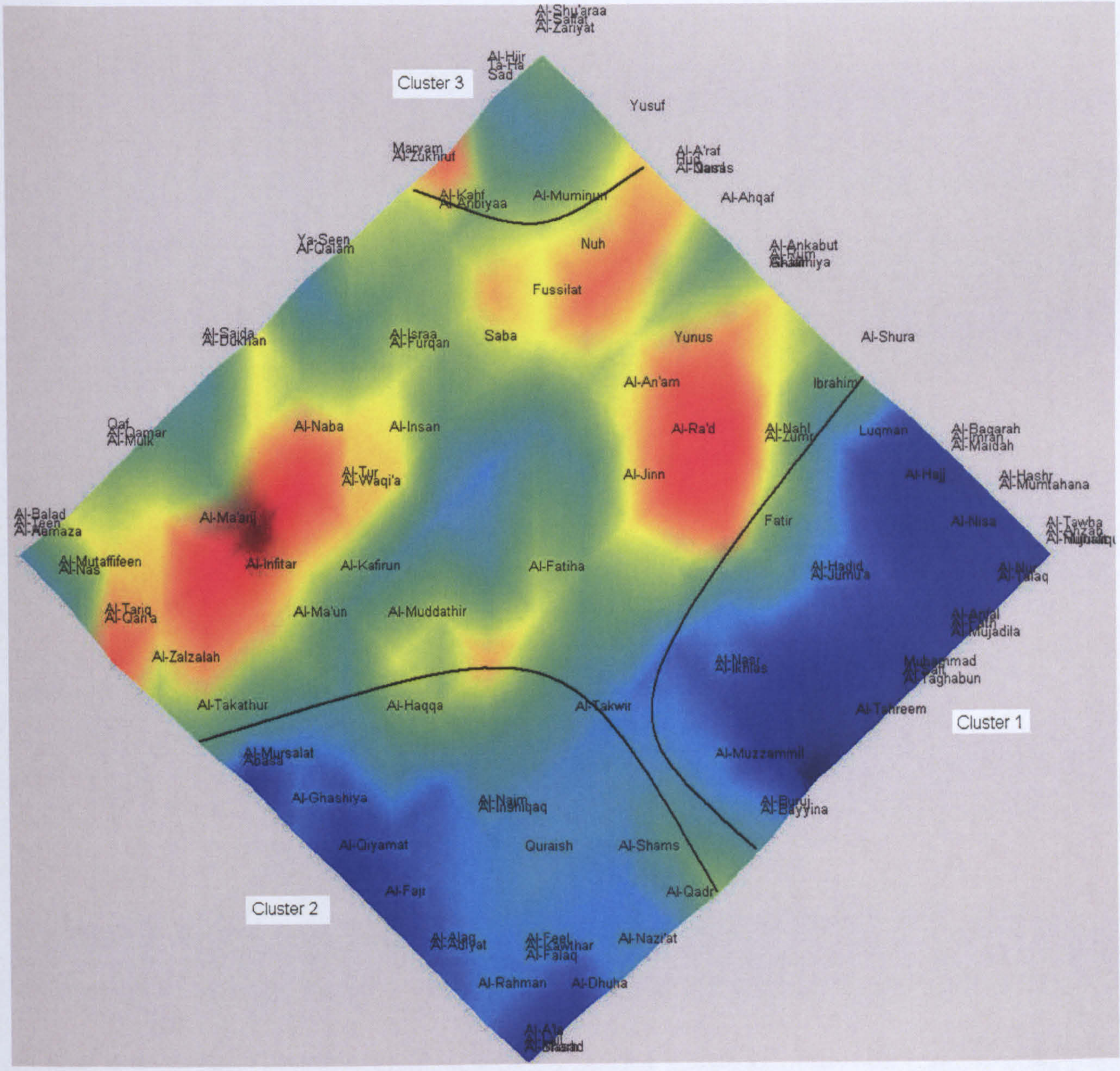


Figure 114: Two-dimensional SOM map of Q1 with clusters 1, 2, 3 shown

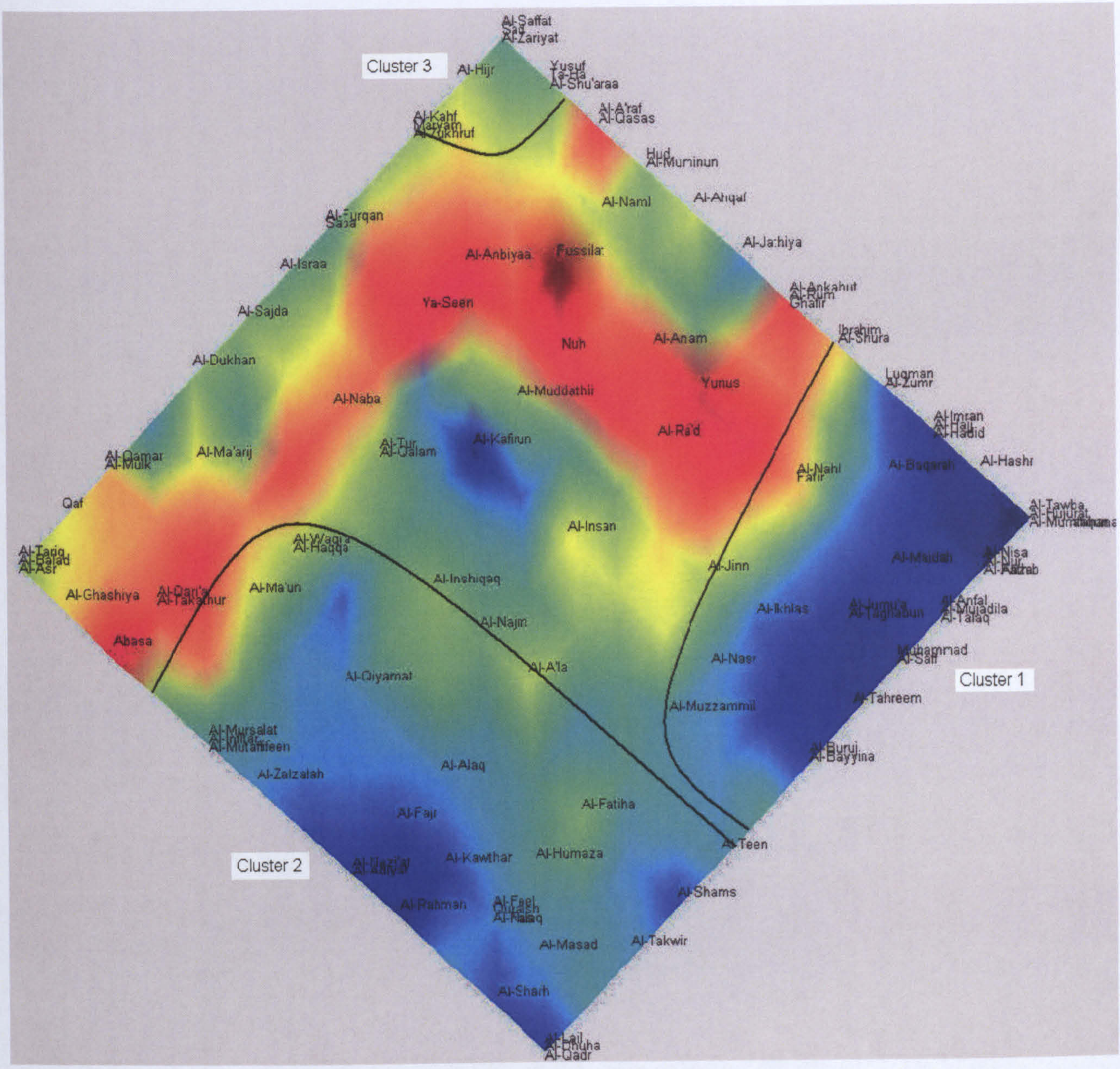


Figure 115: Two-dimensional SOM map of Q2 with clusters 1, 2, 3 shown

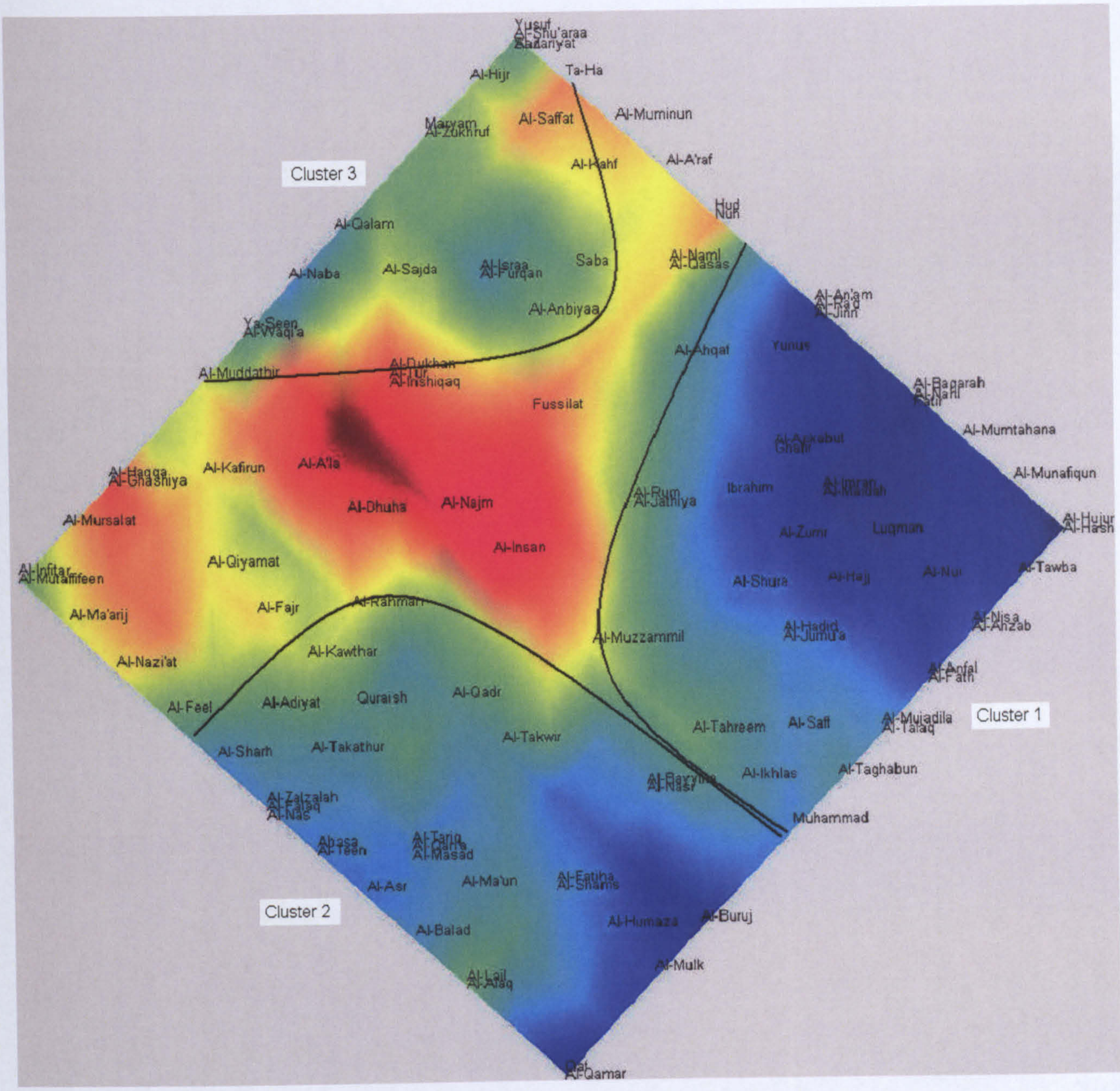


Figure 116: Two-dimensional SOM map of Q3 with clusters 1, 2, 3 shown

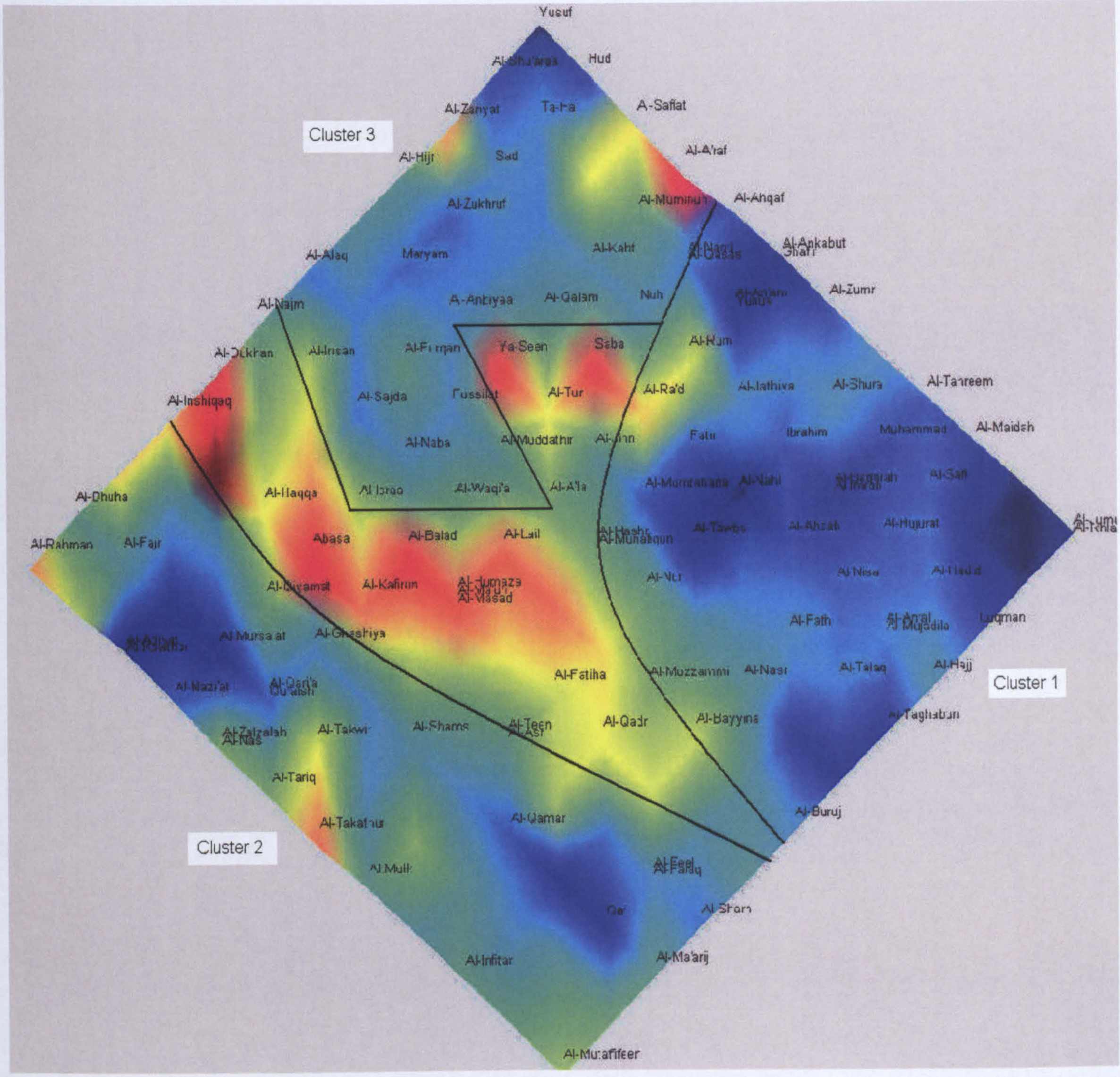


Figure 117: Two-dimensional SOM map of Q4 with clusters 1, 2, 3 shown

Unanimous	**Q1**	**Q2**	**Q3**	**Q4**
Al Hijr	x	x	x	x
Al Kahf	x	x	x	x
Al Saffat	x	x	x	x
Al Shuaraa	x	x	x	x
Al Zariyat	x	x	x	x
Al Zukhruf	x	x	x	x

Maryam	x	x	x	x
Sad	x	x	x	x
Ta Ha	x	x	x	x
Yusuf	x	x	x	x
Three out of four				
Al Anbiyya	x		x	x
Two out of four				
Al Muminun	x			x
Al Furqan			x	x
Al Muddathir			x	x
Al Naba			x	x
Al Qalam			x	x
Al Sajda			x	x
Al Waquia			x	x
One only				
Al Israa			x	
Saba			x	
Ya Seen			x	
Al Alaq				x
Al Insan				x
Al Najm				x
Fussilat				x
Hud				x
Nuh				x

Table 30: *Sura* membership of cluster 3

Of the *suras* in Table 30 for which Q1-Q4 are not unanimous with respect to cluster membership, the following are more or less closely adjacent in the boundary region of the cluster:

Al Anbiyya	Hud
Al Furqan	Nuh
Al Israa	Saba
Al Muminun	Ya Seen
Fussilat	

Table 31: *Suras* adjacent to cluster 3

The following, which Q4 definitely includes in the cluster, are either distant from it in a boundary region or in another --distant-- cluster:

Al Alaq	Al Najm
Al Insan	Al Qalam
Al Muddathir	Al Sajda
Al Naba	Al Waquia

Table 32: *Suras* distant from cluster 3

6.2.2.6 Summary

Though they differ considerably in detail, all four SOMs agree in their clustering of the data matrices Q1-Q4:

1. There are three main clusters separated by broad boundaries

2. The majority of the *suras* belong to one of the three main clusters, though a substantial minority occur in boundary regions between them

3. Each of the main clusters consists of a set of *suras* which is core in the sense that all the maps agree on assigning them to the cluster in question, and a set of *suras* which is peripheral in that either (i) all the maps locate them on the cluster boundary, or (b) one or more maps place them in the cluster, and one or more place them in the boundary region adjacent to the cluster.

4. The three main clusters are not equally well defined:

- Cluster 1 is the most clearly defined both visually and in terms of the high proportion of core to peripheral *suras.*
- Cluster 2 is reasonably clear visually, though its boundary is less uniformly distinct across the maps than that of cluster 1.
- Cluster 3 is hard to detect visually in that its boundary is both indistinct and varies considerably across the maps.

6.2.3 Comparíson of hierarchical and SOM analyses

To see how the SOM analyses compare with the hierarchical ones, the 2-dimensional SOM maps showing the clusters for each of Q1 - Q4 were augmented by the addition of labels which show, for each *sura*, the cluster to which the hierarchical analyses assigned that *sura*. This can be A, B, or C, or, where the hierarchical analyses assign a *sura* to more than one of the main clusters, AB, AC, BC, or ABC. These labels are based on table 23 in the discussion of the hierarchical analyses.

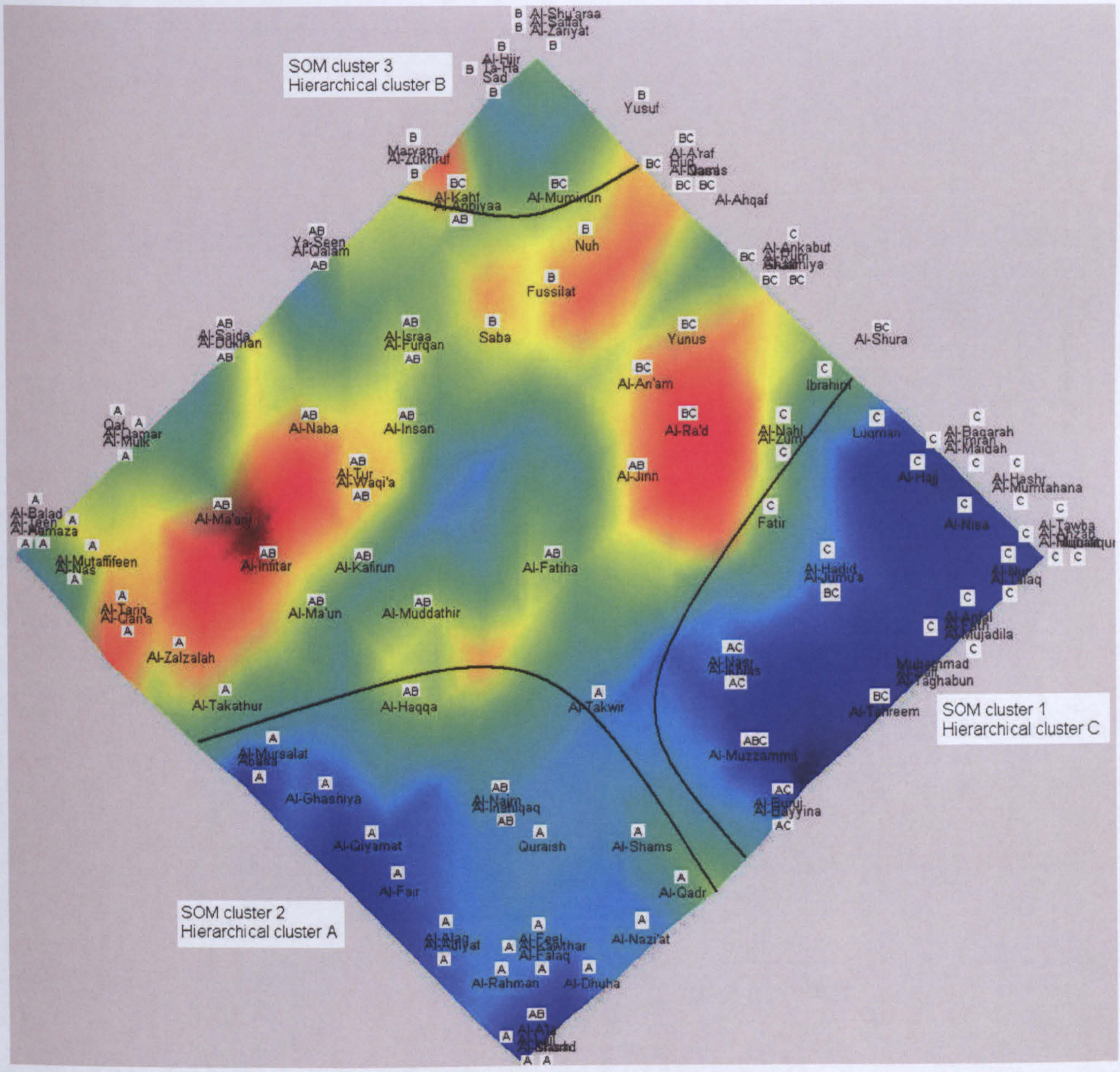


Figure 118: Two-dimensional SOM map of Q1 with clusters 1,2,3 and hierarchical cluster labels shown

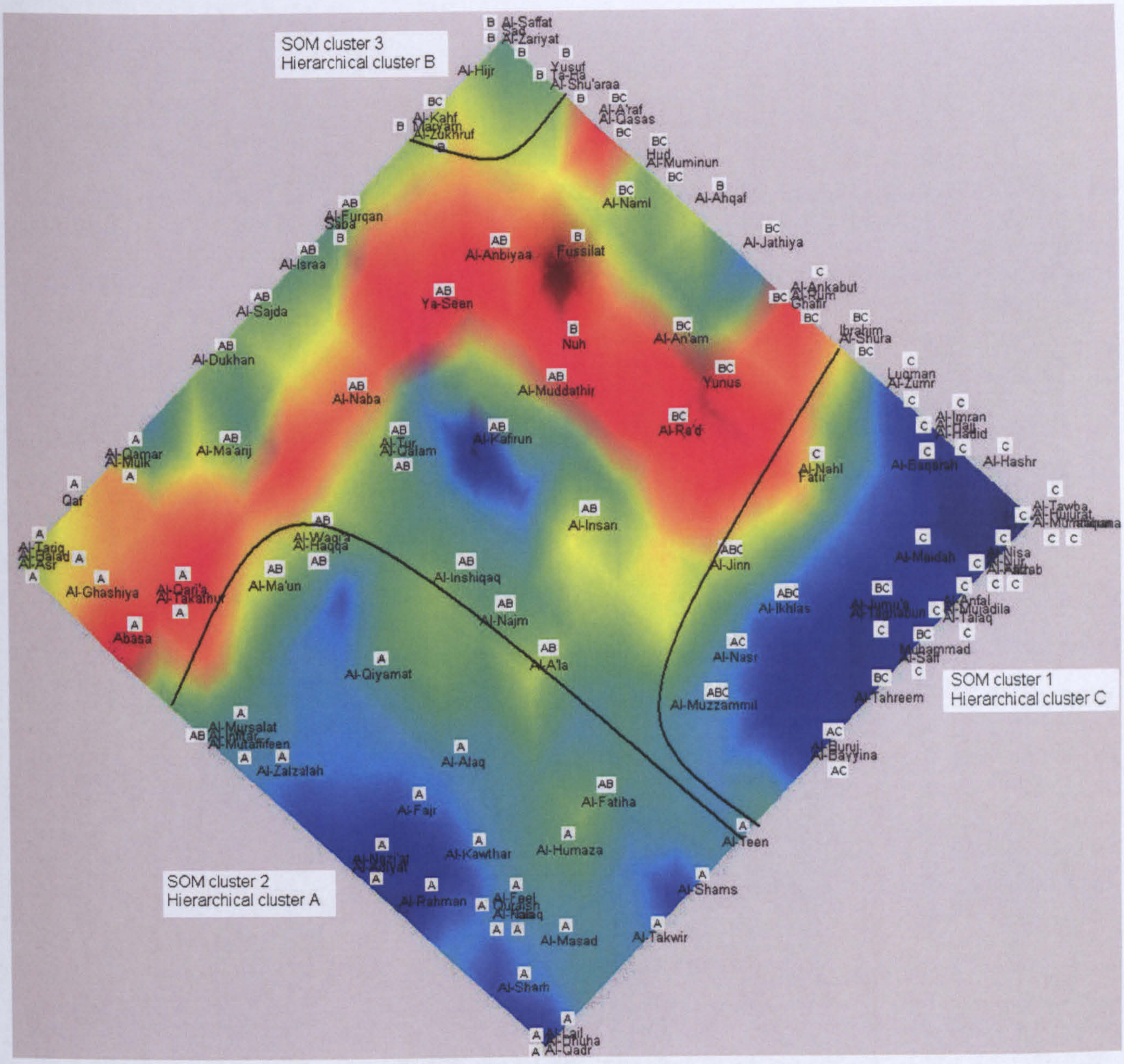


Figure 119: Two-dimensional SOM map of Q2 with clusters 1,2,3 and hierarchical cluster labels shown

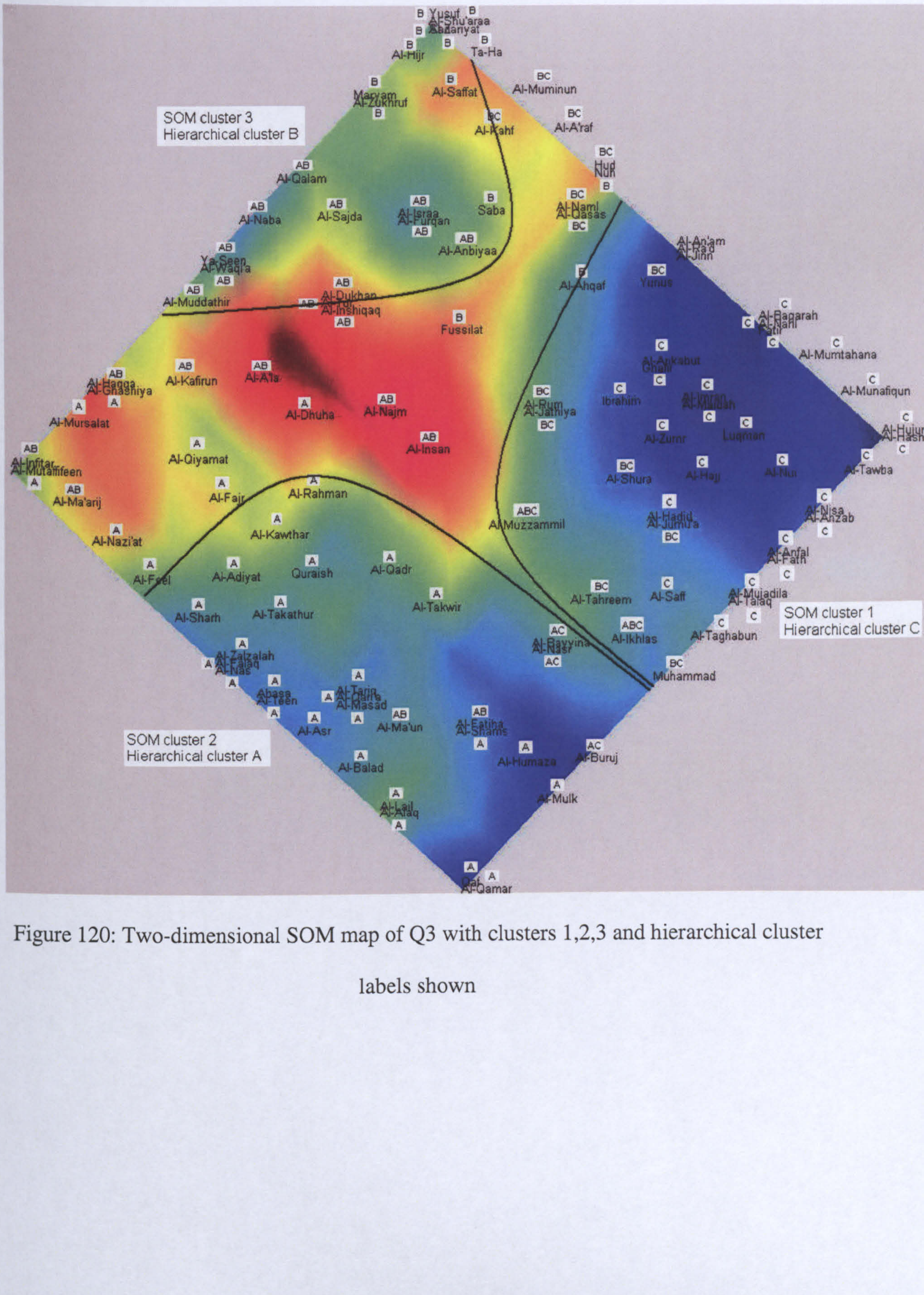


Figure 120: Two-dimensional SOM map of Q3 with clusters 1,2,3 and hierarchical cluster labels shown

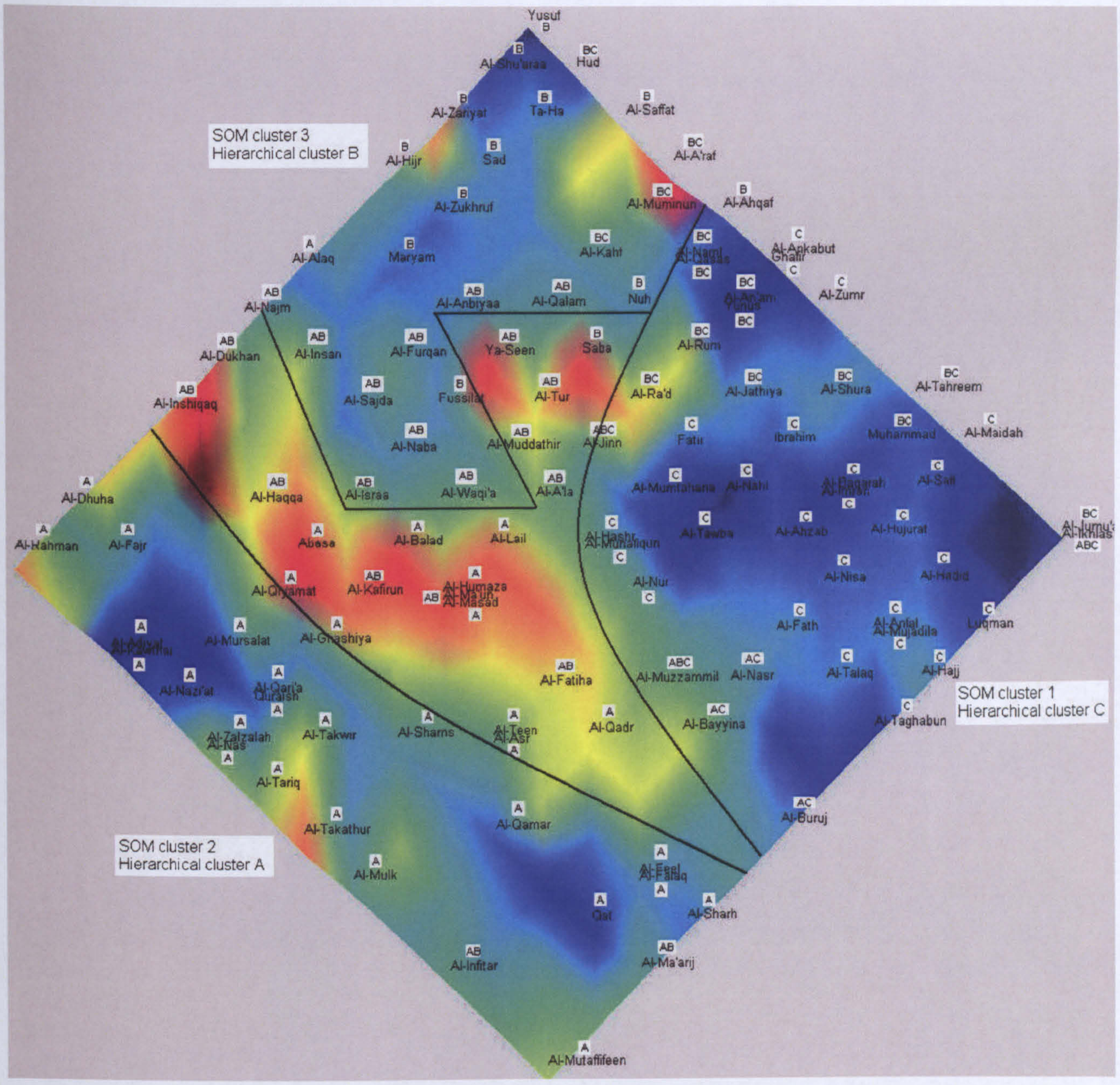


Figure 121: Two-dimensional SOM map of Q4 with clusters 1,2,3 and hierarchical cluster labels shown

Inspection shows a close fit between the results from hierarchical and SOM analysis. Specifically, there is a good degree of correspondence between the three main clusters A, B, and C of hierarchical analysis and the core *suras* in the three main SOM clusters:

- SOM cluster 1 corresponds closely to hierarchical cluster C, as shown in the following table. Where a *sura* in SOM cluster 1 is not in hierarchical cluster C an italicized indication of its occurrence is given, and where a *sura* in hierarchical cluster C is not in SOM cluster 1 the italicized word 'peripheral' is inserted to indicate that the *sura* occurs in the close topological periphery of cluster 1.

Hierarchical cluster C	**SOM cluster 1**
Al Ahzab	Al Ahzab
Al Anfal	Al Anfal
Al Ankabut	*Peripheral*
Al Baqarah	Al Baqarah
Al Fath	Al Fath
Al Hadid	Al Hadid
Al Hajj	Al Hajj
Al Hashr	Al Hashr
Al Hujurat	Al Hujurat
ABC	Al Ikhlas
Al Imran	Al Imran
BC	Al Jumua
Al Maidah	Al Maidah
Al Mujadila	Al Mujadila
Al Mumtahana	Al Mumtahana
Al Munafiqun	Al Mundafiqun

Al Nahl	Al Nahl
Al Nisa	Al Nisa
Al Nur	Al Nur
Al Saff	Al Saff
Al Taghabun	Al Taghabun
BC	Al Tahreem
Al Talaq	Al Talaq
Al Tawba	Al Tawba
Al Zumr	Al Zumr
Fatir	Fatir
Ibrahim	Ibrahim
Lugman	Lugman
BC	Mohammed

Table 33: Tabulation of *suras* in hierarchical cluster C and SOM cluster 1

- SOM cluster 2 is a subset of hierarchical cluster A. As above, where a *sura* in hierarchical cluster A is not in SOM cluster 2 the italicized word 'peripheral' is inserted to indicate that the *sura* occurs in the close topological periphery of cluster 2. Analogously, in the few cases where a *sura* in hierarchical cluster A does not occur in SOM cluster 2 and neither in the close periphery of the main SOM cluster but in a distant region of the map, the italicized word 'distant' is inserted.

Hierarchical cluster A	**SOM cluster 2**
Abasa	*Peripheral*
Al-Adiyat	Al Adiyat
Al-Alaq	*Distant*
Al-Asr	*Peripheral*

Al-Balad	*Peripheral*
Al-Dhuha	*Peripheral*
Al-Fajr	Al Fajr
Al-Falaq	Al Falaq
Al-Feel	Al Feel
Al-Ghashiya	*Peripheral*
Al-Humaza	*Peripheral*
Al-Kawthar	Al Kawthar
Al-Lail	*Peripheral*
Al-Masad	*Peripheral*
Al-Mulk	*Distant*
Al-Mursalat	Al-Mursalat
Al-Mutaffifeen	*Peripheral*
Al-Nas	*Peripheral*
Al-Nazi'at	*Peripheral*
Al-Qadr	*Peripheral*
Al-Qamar	*Distant*
Al-Qari'a	*Peripheral*
Al-Qiyamat	Al-Qiyamat
Al-Rahman	Al Rahman
Al-Shams	Al Shams
Al-Sharh	Al Sharh
Al-Takathur	*Peripheral*
Al-Takwir	Al Takwir
Al-Tariq	*Peripheral*
Al-Teen	Al-Teen
Al-Zalzalah	Al-Zalzalah
Qaf	*Distant*

Quraish	Quraish

Table 34: Tabulation of *suras* in hierarchical cluster A and SOM cluster 2

- SOM cluster 3 corresponds closely to hierarchical cluster B. Again as before, where a *sura* in SOM cluster 3 is not in hierarchical cluster B an italicized indication of its occurrence is given, and where a *sura* in hierarchical cluster B is not in SOM cluster 3 the italicized word 'peripheral' is inserted to indicate that the *sura* occurs in the close periphery of cluster 3, as discussed earlier.

Hierarchical cluster B	**SOM cluster 3**
Al-Ahqaf	*Peripheral*
Al Anbiyaa	Al Anbiyaa
Al-Hijr	Al Hijr
Al-Kahf	Al Kahf
Al-Saffat	Al Saffat
Al-Shu'araa	Al Shuaraa
Al-Zariyat	Al Zariyat
Al-Zukhruf	Al Zukhruf
Fussilat	*Peripheral*
Maryam	Maryam
Nuh	*Peripheral*
Saba	*Peripheral*
Sad	Sad
Ta-Ha	Ta Ha
Yusuf	Yusuf

Table 35: Tabulation of *suras* in hierarchical cluster B and SOM cluster 3

On the basis of these comparisons it is possible to define three core clusters, where a core cluster C_{core} consists of those *suras* which are assigned to C_{core} by all the hierarchical and all the SOM analyses:

Core: **SOM cluster 1 / Hierarchical cluster C**	**Core:** **SOM cluster 2 / Hierarchical cluster A**	**Core:** **SOM cluster 3 / Hierarchical cluster B**
Al Ahzab	Al-Adiyat	Al-Anbiyaa
Al Anfal	Al-Fajr	Al-Hijr
Al Baqarah	Al-Falaq	Al-Kahf
Al Fath	Al-Feel	Al-Saffat
Al Hadid	Al-Kawthar	Al-Shu'araa
Al Hajj	Al-Mursalat	Al-Zariyat
Al Hashr	Al-Qiyamat	Al-Zukhruf
Al Hujurat	Al-Rahman	Maryam
Al Imran	Al-Shams	Sad
Al Maidah	Al-Sharh	Ta-Ha
Al Mujadila	Al-Takwir	Yusuf
Al Mumtahana	Al-Teen	
Al Munafiqun	Al-Zalzalah	
Al Nahl	Quraish	
Al Nisa		
Al Nur		
Al Saff		
Al Taghabun		
Al Talaq		
Al Tawba		
Al Zumr		
Fatir		
Ibrahim		
Luqman		

Table 36: Tabulation of *suras* in core clusters

In addition, the foregoing comparisons identify *suras* which are peripheral to the core clusters in the sense that, for a given *sura* s_p and a given cluster C_{core}, all the hierarchical analyses assign s_p to C_{core} but not all the SOM analyses do, or vice versa:

Peripheral:	**Peripheral:**	**Peripheral:**
SOM cluster 1 / Hierarchical cluster C	**SOM cluster 2 / Hierarchical cluster A**	**SOM cluster 3 / Hierarchical cluster B**
Al Ankabut	Abasa	Al Ahqaf
Al Ikhlas	Al Alaq	Fussilat
Al Jumua	Al Asr	Nuh
Al Tahreem	Al Balad	Saba
Mohammed	Al Dhuha	
	Al Ghashiya	
	Al Humaza	
	Al Lail	
	Al Masad	
	Al Mulk	
	Al Mutaffifeen	
	Al Nas	
	Al Naziat	
	Al Qadr	
	Al Qamar	
	Al Qaria	
	Al Takathur	
	Al Tariq	
	Qaf	

Table 37: Tabulation of *suras* peripheral to core clusters

The remaining *suras* are those that the hierarchical analyses assign to more than one of the core clusters, and the SOM analyses either assign to different core clusters or locate them in the regions of the maps separating the core clusters from one another. Visual

examination of the SOM maps shows that there is a strong relationship between the way in which the hierarchical and the SOM analyses place these *suras* in relation to the core clusters:

- For the *suras* that the hierarchical analyses assign sometimes to cluster A and sometimes to B, the SOM analyses either locate them in the corresponding clusters 2 and 3 or in the region that separates cluster 2 from cluster 3:

Hierarchical A/B, SOM 2/3	
Al-A'la	Al-Ma'arij
Al-Dukhan	Al-Ma'un
Al-Fatiha	Al-Muddathir
Al-Furqan	Al-Naba
Al-Haqqa	Al-Najm
Al-Infitar	Al-Qalam
Al-Insan	Al-Sajda
Al-Inshiqaq	Al-Tur
Al-Israa	Al-Waqi'a
Al-Kafirun	Ya-Seen

Table 38: Tabulation of *suras* intermediate between hierarchical clusters A/B and SOM clusters 2/3

- For the *suras* that the hierarchical analyses assign sometimes to cluster B and sometimes to C, the SOM analyses either locate them in the corresponding clusters 3 and 1 or in the region that separates cluster 3 from cluster 1:

Hierarchical B/C, SOM 3/1	
Al-An'am	Al-Ra'd
Al-A'raf	Al-Rum

Al-Jathiya	Al-Shura
Al-Muminun	Hud
Al-Naml	Yunus
Al-Qasas	

Table 39: Tabulation of *suras* intermediate between hierarchical clusters B/C and SOM clusters 3/1

- For the *suras* that the hierarchical analyses assign sometimes to cluster A and sometimes to C, the SOM analyses locate them in the corresponding clusters 2 and 1:

Hierarchical A/C, SOM 2/1
Al-Bayyina
Al-Buruj
Al-Nasr

Table 40: Tabulation of *suras* intermediate between hierarchical clusters A/C and SOM clusters 2/1

- Finally, for the two *suras* that the hierarchical analyses spread over all three core clusters A, B, and C, the SOM analyses place them in or near cluster 1:

Hierarchical A/B/C, SOM 1
Al Jinn
Al-Muzzamil

Table 41: Tabulation of *suras* intermediate between hierarchical clusters A/B/C and in or near SOM cluster 1

On the basis of the foregoing findings, it can be concluded that the hierarchical and SOM cluster analyses of Q1 - Q4 are in substantial agreement: both assign the majority of the *suras* to three main clusters, and both agree that the substantial minority that remains are

intermediate between the three. Since hierarchical analysis and the SOM are fundamentally different clustering methods --the first is a linear method based on Euclidean distance between vectors and the second a nonlinear method based on topology preservation-- this agreement strongly supports the claim, now being made, that these clusters are not simply methodological artefacts but represent objectively real structure in the data.

Chapter Seven

Interpretation

7.1 Interpretation

The research question formulated at the outset of this study was to see whether the *suras* can be classified in a thematically coherent way on the basis of their lexical semantic content. Does the finding that the majority of the *suras* can be assigned to three main clusters bear usefully on this question and, if so, how? The answer is yes; the remainder of this chapter justifies that claim. The argument will be that each of these three clusters has a characteristic thematic unity that differentiates it from the others.

On the face of it, thematic coherence would appear to be an intrinsically subjective notion. It is, however, possible to objectify it to some degree using a quantitative criterion, which is now proposed. Cluster analysis classifies multidimensional vectors on the basis of their relative similarity: vectors in any given cluster are more similar to one another on some measurement criterion than they are to vectors in any other cluster. In the present application, the *suras* were clustered on the basis of lexical frequency vectors. The existence of distinct clusters therefore implies that each cluster has a characteristic lexical frequency profile which distinguishes it from the others. By comparing the lexical frequency profiles of the three *sura* clusters, therefore, it should be possible to determine the lexical items in which they differ most, and, on the basis of the lexical semantics of these items, to infer thematic characteristics of the respective *sura* clusters. What is a 'lexical frequency profile' for a cluster? It is an average vector constructed from the various *sura* lexical frequency vectors that constitute the cluster by adding the corresponding elements of each *sura* vector and taking the mean of the sum:

$$p_j = \left(\sum_{i=1..n} S_{i,j}\right) / n$$

where *j* is the index to the *j*th element of the profile vector *p*, *i* indexes the vectors of the *sura* set S that comprise the cluster, and *n* is the total number of vectors in the cluster. Such a profile vector is constructed for each of the three core clusters.

Before constructing lexical frequency profiles for the three clusters, a few technical issues have to be dealt with:

- As the foregoing discussion showed, the three clusters are 'fuzzy' in the sense that there is a core set of *suras* for each, a peripheral set closely associated with each core, and a set of *suras* that are intermediate between the clusters. The core *sura* sets can be expected to be maximally distinct from one another in terms of their lexical frequency profiles and thus to give the clearest results when compared. As such, cluster profiles are constructed only from the core *sura* sets, excluding the peripheral and intermediate *suras*.
- The construction of cluster lexical frequency profiles is complicated by the fact that the analyses were based on four different matrices Q1 - Q4: which should be chosen for the following analysis? Of these, Q3 and Q4 are unusable for the purpose because they are versions of Q1 and Q2 whose dimensionality has been reduced by PCA to 100, and therefore consist of *sura* vectors whose variables are derived and uninterpreted. In other words, for Q3 and Q4, the variables bear no obvious relationship to the original lexical variables of Q1 and Q2 from which PCA derived them, and, because they lack any obvious lexical semantics, they cannot be used for thematic inference. This leaves Q1 and Q2; for which of these to choose, see further below.

- If Q1 is to be the basis for semantic interpretation, then 761 lexical variables have to be considered, and if Q2 1236 variables. Neither case is practicable: there are simply too many variables. Their number must therefore be reduced to a reasonable level by selecting the semantically most important variables for consideration. This of course begs the question of how to select the semantically most important variables. The criterion used here is variance, for the same reason that variance was used as one of the criteria for variable selection in the data creation phase of the study: the higher the variance of a variable, the better is distinguishes the entities it describes. Since the aim is to distinguish the three *sura* clusters from one another in terms of variation in their lexical frequencies, the relative variance of the lexical variables in Q1 and Q2 is a good selection criterion for semantic importance. How many variables should be selected? Various possibilities were tried, and it was found that the 50 highest-variance variables were sufficient for the present purpose.

  These 50 variables turned out to be identical for Q1 and Q2, and as such it didn't matter which of the matrices is chosen as the basis for analysis. Q1 was arbitrarily selected.

Three length-100 cluster lexical frequency profiles were constructed, one for each of the core *sura* clusters. These were sorted in descending order of variance and then co-plotted, so that the highest-variance variables are on the left of the plot, the aim being to give an idea of the general shape of the distribution:

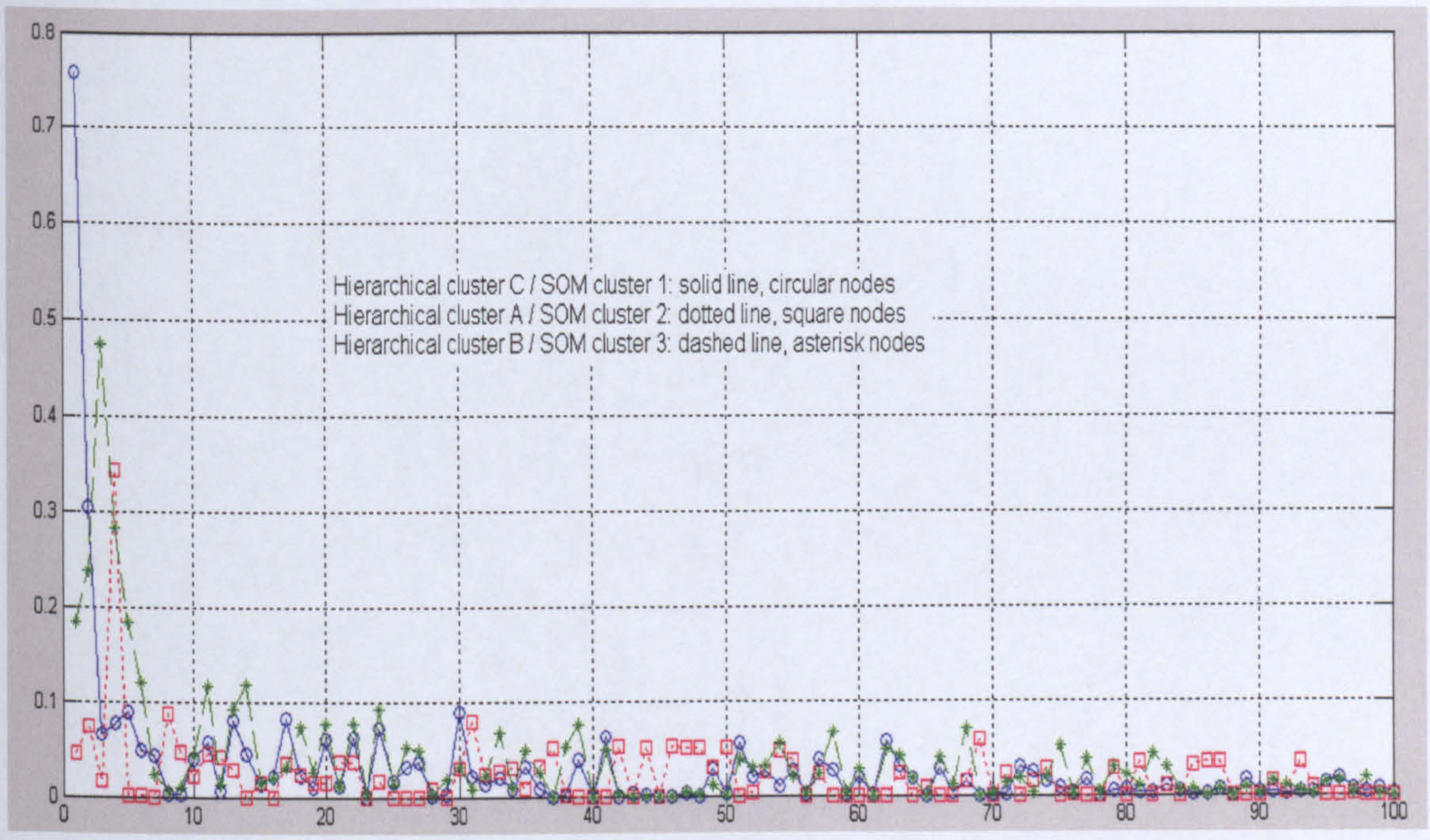


Figure 122: Co-plot of lexical frequency profiles for 3 core *sura* clusters

This plot was then truncated to the 50 variables selected for discussion and expanded for greater legibility. This truncated the highest values at the left of the plot, but the relative scaling of these can be seen in figure 123.

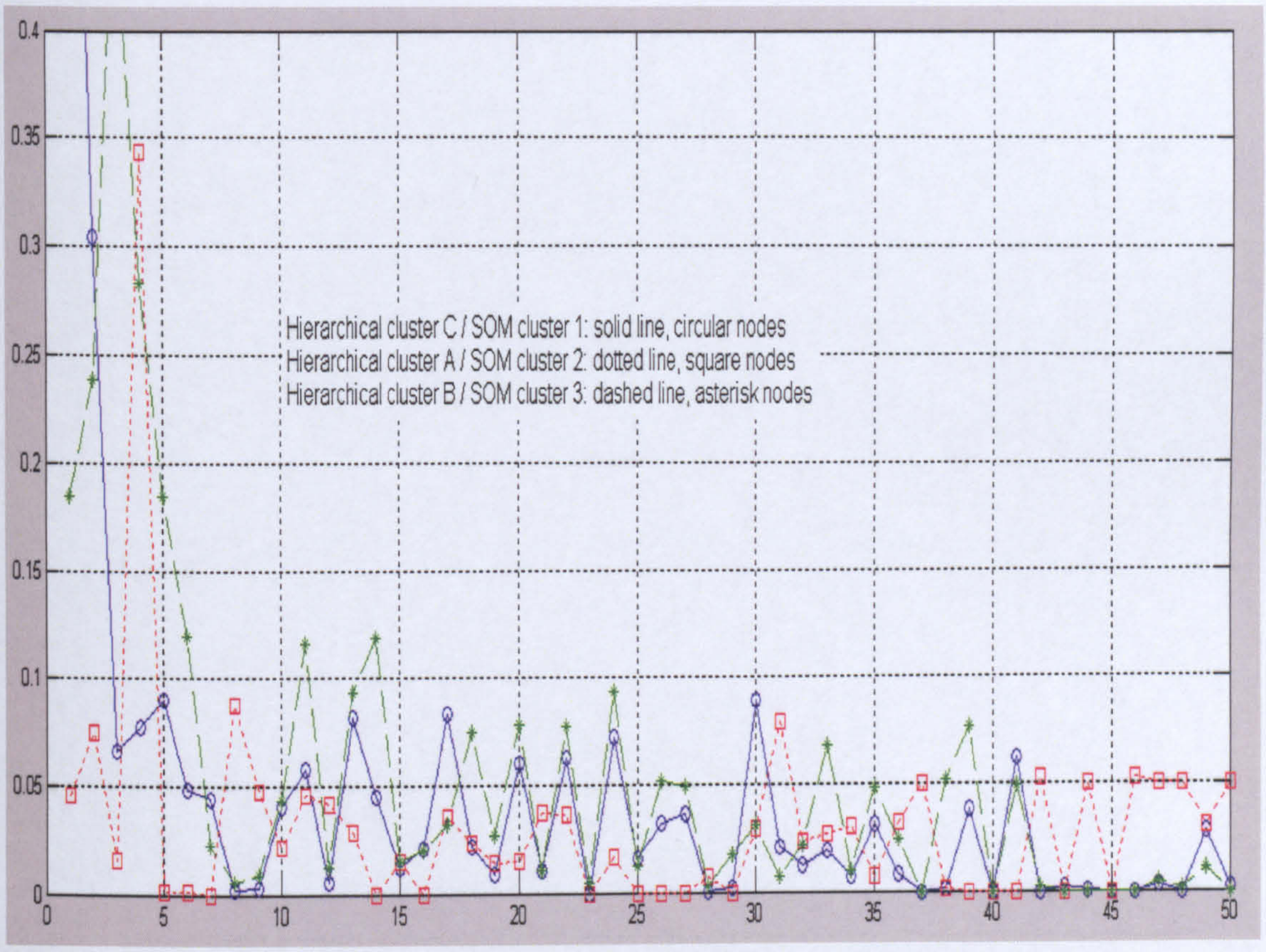


Figure 123: Figure 122 horizontally truncated to 50 variables

Each of the variables on the x-axis has a lexical interpretation, which is given in Table 42 below, together with the meaning of each Arabic word:

Variable nr	Variable name	Meaning	Variable nr	Variable name	Meaning
1	allAh	God	26	AyAt	signs
2	lA	no/not	27	kitAb	book
3	qAl	said	28	mAl	wealth
4	rabb	lord	29	wayl	woe
5	kAn	was	30	rasool	Messenger
6	yawm	day	31	alam	Did you not?
7	nAs	mankind	32	aS-HAb	companions
8	sharr	evil	33	khalaq	created
9	yawma-iZ	that Day	34	kayf	how

10	qul	say	35	jA	came
11	qad	already	36	kaZZab	lied
12	insAn	man	37	tabb	perish
13	arD	earth	38	raHmAn	merciful
14	qawm	people	39	Haqq	truth
15	aHad	anyone	40	qAri&at	calamity
16	deen	religion	41	mu'min	believer
17	Aman	believed	42	tukaZZib	deny
18	ja&al	made	43	qadr	estimate
19	ra	saw	44	Sall	prayed
20	&aZAb	torment	45	adrAk	realize
21	layl	night	46	AlA-	favours
22	samA	heaven/s	47	yusr	ease
23	tawAS	enjoin	48	a&T	gave
24	yA	O	49	tar	see
25	ta&lam	know	50	&usr	difficulty

Table 42: Lexical interpretation of 50 variables in Figure 123

Using the plots in conjunction with the x-axis labels, it is possible to determine which variables are most and least characteristic of each cluster, and which differentiate them most. In what follows, the lexical variables of most interest to each of the clusters are discussed. Supporting examples are taken from the translation of the Qur'an by Ghali (1997). They are quoted and then identified by the (*sura* number: *aya* number).

In the discussion that follows, familiarity with Chapter 2, 'Tradition' and with Chapter 4, 'Literature review' is assumed. The following interpretation of the *suras* is based on Abdul-Roaf (2001b), Al-Ghazali (2000) and Al-Ghazali (1998).

SOM cluster 1 / hierarchical cluster C

The *suras* of this cluster are most interested in the following words: *AllAh, Aman, mu'min, rasool*, and *nAs*.

- *AllAh* (God): The *suras* of cluster 1/C are strikingly more concerned with the denotation of *Allah*, the highest-variance variable in the Qur'an, than the *suras* of clusters A and B. The word *Allah* occurs 2176 times in the whole Qur'an, of which 1761 occurrences were in *suras* composed in Medina, which comprise the majority of this cluster; the only *sura* in which the word *Allah* is mentioned in every single verse is *sura Al-Mujadila* which occurs in this cluster. The disparity in the frequency of its occurrence across the three clusters is the first significant finding of this study. The word *Allah* seems to be absent from the earlier revelations and starts to appear towards the end of the Meccan period and then became increasingly frequent in the Medinan *suras* as opposed to the impersonal *rabb* ('god') in the earlier Meccan ones. Welch (1980, 754) mentions that in 29 of the 114 *suras* of the Qur'an the name *Allah* does not occur at all. He adds that a rough statistical survey shows that the name *Allah* occurs an average of once every two lines in the *suras* that are completely Medinan, while it occurs an average of only once in every 24 lines in *suras* he regards as completely Meccan, and once every six lines in those *suras* he regards as part Meccan and part Medinan. The Meccan *suras* are full of verses affirming Allah's oneness and refuting idolatry in all its forms. However, this is continued and expanded in the Medinan period with more clarification and detail. The emphatic denial and refutation of there being any god except Allah, and the equally emphatic affirmation of unity, oneness and uniqueness of Allah, is a

dominant theme in the *suras* of this cluster. Here, not only is the Person of Allah stated most clearly to be One, but the association of any other god with Him is refuted on all other possible grounds.

The name *Allah* soon became the accepted, and preferred, designation for Mohammed's Lord as is seen in numerous Medinan passages, such as the following:

Qur'an Verse	***Sura*:verse**
Allah. There is no god except He, The Ever-Living, The Sublime Self-Subsisting	(2:255)
Surely, your patron is only *Allah* and His Messenger, and the ones who have believed	(5:55)
And to *Allah* is the Kingdom of the heavens and the earth; and to *Allah* is the Destiny	(24:42)
And obey *Allah* and His Messenger; and *Allah* is Ever-Cognizant of whatever you do	(58:13)
Surely *Allah* is my Lord and your Lord; so worship Him. This is the straight path	(3:51)
Surely this is indeed the true narrative and in no way is there any god except *Allah*; and surely *Allah* is indeed He Who is the Ever-Mighty, The Ever-Wise.	(3: 62)

- *Aman* ('believed') / *mu'min* ('believer'): These two words occur frequently in cluster 1/C, the Medinan *suras*. These are *suras* in which prophet Mohammed addresses those who already believe in his message and hence focuses on introducing them to the social and ethical aspects of Islam. One of the ways of distinguishing between the two periods of Mecca and Medina is the manner of address. Whereas the passages in Meccan *suras* usually speak to Mohammed himself or people in general, the Medinan passages are often addressed to

Mohammed's followers as in *Ya ayyuha allatheena Amanoo* ('those who believe') or *mu/mineen* ('believers'). These pronouncements are followed by information, insights, teachings and directives to implement the command of Allah and go straight on His path. What follows is often of a legislative nature, and it is true to say that the laws of Islam (*shari'ah*) are found principally in the passages dating from Mohammed's migration to Medina. There are 89 instances in the Qur'an for the verse *Ya ayyuha allatheena Amanoo*, ('O you who have believed') all of which are Medinan revelations. Sixteen instances are in *sura Al-Maidah* which begins with this verse, as do *Sura Al-Hujurat* and *Al-Mumtahana*; all core members of cluster 1/C.

The following are some other examples containing these lexical items:

Qur'an Verse	***Sura*:verse**
O you who have believed, seek help in patience and prayer; surely Allah is with the patient ones.	(2:153)
O you who have believed, prescribed for you is the Fast, as it was prescribed for the ones who were even before you	(2:183)
O you who have believed, enter into peacefulness, the whole of you, and do not ever follow the steps of Satan; Surely he is an evident enemy to you	(2:208)
O you who have believed, do not void your donations with obliging reproach and hurt, as one who expends his wealth for the sake of showing off to mankind	(2:264)
O you who have believed, when you contract a debt one upon another for a stated term, then write it down; and let a writer write it down between you with justice, and let not any writer refuse to write it down, as Allah has taught him.	(2:282)
O you who have believed, it is not lawful for you to inherit women against their will; neither pose problems for them, that	

you may go away with some part of what you brought them	(4:19)
O you who have believed, do not draw near to prayer when you are drunken until you know what you are saying nor when you are ritually unclean	(4:43)
O you who have believed, obey Allah and obey the Messenger	(4:59)
O you who have believed, take your wary precautions; so march out in detachments, or march out altogether	(4:71)
O you who have believed, be ever upright with equity with others, witnesses for Allah, even if it be against yourselves, or your parents and nearest kin; in case the person is rich or poor, then Allah is the best patron for both. So do not ever follow prejudice, so as to do justice	(4:135)
And Allah will eventually bring the *believers* a magnificent reward.	(4:146)
O you who have believed, fulfil your contracts	(5:1)
O you who have believed, when you rise up for prayer then wash your faces, and your hands up to the elbows, and wipe your heads, and wash your legs to the ankles	(5:6)
O you who have believed, do not take to yourselves the ones that take your religion in mockery and as a plaything	(5:57)
O you who have believed, do not prohibit whatever good things Allah has made to you, and do not transgress	(5:87)
O you who have believed, surely wine and games of chance, and standards for idols and divining are only abomination of Satan's doing, so avoid it, that possibly you would prosper	(5:90)
O you who have believed, look after yourselves. He who errs cannot harm you	(5:105)
O you who have believed, when you meet an (enemy) community, then stand firm, and remember Allah much, that possibly you would prosper	(8:45)
Allah has promised the men *believers* and the women *believers* Gardens from beneath which the Rivers run, eternally abiding therein.	(9:72)

O you who have believed, bow down and prostrate yourselves, and worship your Lord, and perform charity that possibly you would prosper	(22:77)
O you who have believed, do not ever follow the steps of Satan	(24:21)
O you who have believed, do not enter houses other than your houses until you first announce your presence and salute the family thereof	(24:27)
O you who have believed, when you marry women *believers*, and thereafter divorce them before you touch them, then in no way do you have any fixed spell to calculate against them	(33:49)
And give good tidings to the *believers* that they will have great Grace from Allah.	(33:47)
O you who have believed, let not any people scoff at another people who may be more charitable than they; neither let a women scoff other women who may be more charitable than they. And do not defame one another, neither revile one another by nicknames	(49:11)
O you who have believed, avoid much surmise; surely some surmise is a vice; and do not spy on each other, neither backbite one another	(49:12)
O you who have believed when you confer privately together, then do not confer privately on hostility and disobedience to the Messenger; and confer privately together in benignancy and piety; and be pious to Allah	(58:9)

- *Rasool* ('messenger'): Most occurrences of *rasool* in the Qur'an were associated with the prophet Mohammed in these *suras*. One of the most significant distinctions between the Meccan and Medinan periods of revelation is the amount of attention paid to Mohammed himself as a Messenger to people in the Medinan *suras*. Although the Meccan *suras* are often addressed to him, he is very rarely the subject of the revelations, but in Medinan *suras* he comes regularly to the fore. Passages dealing with the Day of Judgement and the rising

from the dead give way to new Medinan revelations much concerned with immediate issues of Mohammed's private life in particular and the believers' code of life in general. Most of the passages of this period exhort believers to obey Allah and His Messenger, which lays the foundations for a Muslim's life by following his instructions. *Suras* from the Medinan period reflect Mohammed's new position as a political, economic, social, and military leader and so address a wider range of societal, historical, and legal issues. As ruler of a state, Mohammed was faced with an array of specific problems, some of which are answered with Qur'anic revelations.

Some *suras* begin with direct reference to the Messenger as in *Al-Anfal*, *Al-Tawba, Al-Hujurat, Al-Mumtahana,* and *Al-Munafiqun*. The following are a few other references to the Messenger from the *suras* of this cluster:

i. Instructions given to obey the Messenger:

Qur'an Verse	***Sura*:verse**
And obey Allah and obey the *Messenger*	(4:59), (5:92), (24:54), (47:33), (64:12)
Obey Allah and the *Messenger*	(3:32), (3:132)
Obey Allah and His *Messenger*	(8:20), (8:46)

ii. Those who obey the Messenger will be winners:

Qur'an Verse	***Sura*:verse**
And keep up the prayer, and bring the Zakat, and obey the *Messenger*, that possibly you would be granted mercy	(24:56)
And whosoever obeys Allah and the *Messenger*-then those are with the ones whom Allah has favoured	(4:69)
Those are the bounds of Allah; and whosoever obeys Allah and	(4: 13)

His *Messenger*, He will cause him to enter Gardens from beneath which Rivers run, eternally abiding therein; and that is the magnificent triumph.	

iii. Those who rejected and disobeyed the Messenger are promised severe punishment:

Qur'an Verse	***Sura*:verse**
And whosoever disobeys Allah and His *Messenger*, and transgresses His bounds, He will cause him to enter a Fire, eternally abiding therein, and for him is a degrading torment	(4:14)
That is for that they opposed Allah and His *Messenger*; and whosoever opposes Allah and His *Messenger*; then surely Allah is strict in punishment.	(8:13)

iv. There are several references to hypocrites and their hidden motives and plans against the Messenger in Medinan *suras*:

Qur'an Verse	***Sura*:verse**
And when it is said to them, Come to what Allah has sent down, and to the *Messenger*, you see the hypocrites barring the way to you forbiddingly	(4:61)
When the hypocrites come to you, they say, We testify that surely you are indeed the *Messenger* of Allah. And Allah knows that surely you are indeed his *Messenger*; and Allah testifies that surely the hypocrites are indeed liars.	(63:1)

v. One of the major themes of Medinan *suras* is the call to fight those who attacked the Messenger of Allah in his defence of faith and freedom. The *suras* deal with the code of practice during the state of war and in the battlefield. In these *suras*, instructions are given to the Messenger and believers about fighting, rules of prisoners of war, the booty, ceasefire and peace agreement:

Qur'an Verse	*Sura*:verse
And know that whatever thing you take as booty, then the fifth of it is for Allah and for the *Messenger*, and for a near kinsman, and the orphans, and the indigent, and the wayfarer	(8:41)
Whatsoever spoils Allah has conceded His *Messenger* from the population of the towns, then that is for Allah and for the *Messenger*, and for a near kinsman, and the orphans, and the indigent, and the wayfarer	(59:7)
Will you not fight a people who breached their oaths and designed to drive out the *Messenger*, and it was they who began the first time against you?	(9:13)

vi. The Messenger will be a witness on the Day of resurrection of the believers' acts:

Qur'an Verse	*Sura*:verse
Who has named you Muslims aforetime, and in this book that the *Messenger* may be an ever-present witness against you	(22:78)
And thus we have made you a middle nation to be witnesses over mankind, and for the *Messenger* to be a witness over you.	(2:143)

vii. There is a clear reference made to the Messenger setting a fair example to be followed and referred to by Muslims

Qur'an Verse	*Sura*:verse
Indeed you have already had a fair example in the *Messenger* of Allah for whosoever hopes for Allah and the Last Day, and remembers Allah much.	(33:21)

- *nAs* ('mankind'): Out of 241 occurrences of the word *nAs* in the Qur'an, 129 appear in Medinan revelations. As mentioned earlier, Medinan *suras* focused on establishing the foundations for people's Islamic life. A Muslim code of conduct was set up to regulate relations among Muslims themselves on the one hand, and

within the family on the other. Instructions are given to people regarding religious duties, inheritance, jihad and social and public relations. This great focus on people's life, attitude and behaviour is reflected in the frequent occurrence of *nAs*. For example:

Qur'an Verse	***Sura*:verse**
And speak fair to *mankind*, and keep up the prayer	(2:83)
And the ships that run in the sea with whatever profits *mankind*	(2:164)
O you *mankind*, eat of whatever is in the earth lawful and good	(2:168)
The month of Ramadan is the month wherein the Qur'an was sent down: a guidance to *mankind*	(2:185)
And when you judge among *mankind* that you judge with justice	(4:58)
And for their taking usury, and they were already forbidden it and eating up the riches of *mankind* untruthfully	(4:161)
And among *mankind* is he who disputes concerning Allah without knowledge or guidance	(22:8)
And announce to *mankind* the pilgrimage, and they shall come up hurriedly to you	(22:27)
Indeed We have already sent Our Messengers with the clear evidences, and We have sent down with them the Book and the Balance, that *mankind* may keep up equity	(57:25)
And We have sent down iron, wherein is strict violence, and various advantages for *mankind*	(57:25)

SOM cluster 2 / hierarchical cluster A

The *suras* of this cluster are most interested in the following words: *rabb, yawma-iZ, insAn, AlA-*.

- *Rabb* ('Lord'): According to Robinson (1996) the *suras* of this cluster belong to the early Meccan revelations. In the earliest revelations, the use of *rabb* is very common, in expressions such as 'your Lord' (*rabbuka* or *rabubukum)*, 'his Lord' (*rabbuhu),* 'our Lord' (*rabbana)*, etc. This term referred to the unnamed god that Mohammed called people to worship. After at first being unnamed in the Qur'an, Mohammed's Lord is for a while called *Al-Rahman*, and finally is identified with Allah. These identifications determined Mohammed's major tasks and mapped out the necessary course of his mission. First he had to separate Allah from the other Arabian deities in the minds of the people, and then he had to convince all the people that Allah alone is to be worshipped. Here are some examples:

Qur'an Verse	***Sura*:verse**
So let them worship the *Lord* of this House	(106:3)
Have you not seen what your *Lord* has done with Ad	(89:6)
Then, as for man, as long as his *Lord* tries him, so He honors him, and showers His favors on him, then he says, "My *Lord* has honoured me "	(89:15)
Then to whichever of your *Lord*'s boons do you both cry lies?	(55: 13,16,18,2,23, 25,28,30,32,34 ,36,38,40,42,4 5,47,49,51,53, 55,57,59,61,63 ,65,67,69,71,7 3,75,77)
Surely upon that Day their *Lord* is indeed of them Ever-Cognizant	(100:11)
So pray to your *Lord* and slaughter the sacrifice	(108:2)

- *Yawma-iZ* ('that Day'): Throughout the whole Qur'an, this word is mentioned 70 times, of which 33 occurrences belong to the *suras* of cluster 2/A, the cluster that contains the shortest *suras* in the Qur'an and represents a very small portion of its volume. Most of the *suras* of cluster 2/A are associated with the occurrence of the Day of Judgement and the various scenes taking place on the day --of reckoning, fine reward and severe retribution. The believers will be the winners and will be granted rewards and blessings from Allah. The losers, on the other hand, together with the disbelievers, will be punished in fire. The word *yawma-iZ* makes a clear reference to the Day of Judgement and the horrors of that day which is marked by violent convulsions and horrific cosmic events. Here are some examples of verses containing this word:

Qur'an Verse	***Sura*:verse**
Upon *that Day* man shall be fully informed of whatever he forwarded or deferred	(75:13)
Upon *that Day* woe to the beliers	(77:15,19,24,28,34,37,40,45,47,49)
Some faces upon *that Day* shall shine	(80:38)
And some faces upon *that Day* shall be covered by resentment	(80:40)
A Day when no self shall posses anything to help another self; and the Command upon *that Day* belongs to Allah	(82:19)
And Hell is made to come face to face upon *that Day*, man will remember, upon that Day, and howsoever will the reminding avail him?	(89:23)
So, upon *that Day* none shall torment as he torments	(89:25)
Upon *that Day* mankind shall go forward in diverse groups to be shown their deeds	(99:6)
Surely, upon *that Day* their Lord is indeed of them Ever-Cognizant	(100:11)

- *InsAn* ('man'): *Insan* occurs 65 times in the Qur'an, of which 27 occurrences are in cluster 2/A. Throughout the *suras* of this cluster, emphasis is laid on the origin of man as well as the origin of life. Allah had a recurrent message for mankind: it has been a fault of human beings throughout history of the human race that their awareness of the Day of Judgement has been non-existent or very weak. Allah's messages in these *suras* are to prove that He Who causes human bodies to decay is capable of bringing them back again with exactly the same traits. Man will be brought to life again in order to receive the recompense for what he did. Another major theme in this cluster is man's ingratitude to Allah. In view of Allah's goodness man ought to be grateful to Him, however, man disbelieved in Him, rejected His Messenger and denied the Day of Judgement. Man's creation, guidance and his attitude towards the Day of Judgement are referred to in several passages:

Qur'an Verse	***Sura*:verse**
Surely *man* is indeed ungrateful to his Lord	(100:6)
Does *man* reckon that We shall never gather his bones?	(75:3)
No indeed, but *man* would like to act impiously in the life before him.	(75:5)
Upon that Day *man* shall say, where to flee?	(75:10)
Does *man* reckon that he shall be left in vain?	(75:36)
Surely We created *man* from a sperm-drop, a mingling, trying him	(76:2)
Slain be *man*, how disbelieving he is	(80:17)
Created *man* from clot	(96:2)
He taught *man* what he did not know	(96:5)
So let *man* look into what he was created from; he was created from effusive water	(86:5,6)

- *AlA-* ('bounties / boons') and *tukaZZib* ('lies'): *AlA-* occurrs 34 times in the Qur'an, of which 31 are in s*ura Al-Rahman,* one of the core *suras* of this cluster. The second word *tukaZZib* occurrs 41 times in the Qur'an, of which 31 instances are also in *sura Al-Rahman.* The *sura* lists Allah's infinite favours bestowed on mankind. The verse 'Then to whichever of your Lord's *boons* do you both cry *lies*?" which comes first in verse 11 is repeated thirty-one times in the course of the *sura.*

SOM cluster 3 / hierarchical cluster B

The *suras* of this cluster are most interested in the following words: *qul, qAl, qawm, AyAt, arD, samA, khalaq, ja&al, kitAb, yawm, &aZAb, wayl and rahmAn.*

- *qul* ('say'), *qAl* ('said') and *qawm* ('people'): The *suras* of this cluster contain many narratives which illustrate important aspects of the Qur'anic message, remind of the earlier prophets and their struggles, and strengthen Prophet Mohammed's message of Islam. Other narratives refer to persons or events that took place during the lifetime of the Prophet. This usually features *qul* and its morphological variant *qAl.* Most of the passages of these Meccan *suras* start with the imperative *qul*, which is an instruction to Prophet Mohammed to address the words that follow to his audience in a particular situation, such as in reply to a question that has been raised, or an assertion of a matter of belief. Q*ul* and *qAl* are repeated well over 300 times in the Qur'an, and most of the occurrences are in the *suras* of this cluster.

The word *qawm* occurs 373 times in the Qur'an, mostly in connection with Meccan revelations where references are made to other nations and their prophets in expressions such as *qawm Nuh* 'the people of Nuh', *qawm Musa* 'the people of Musa", and so on.

The following are some examples featuring the above three lexical items:

Qur'an Verse	***Sura*: verse**
And *say*, Surely, I, even I, am the evident warner	(15:89)
And *say*, The Truth is from your Lord; so whosoever decides, then let him believe, and whosoever decides, then let him believe, and whosoever decides, then let him disbelieve.	(18:29)
Say, If the sea were a constant supply for the Words of my Lord, indeed the sea would be depleted before the Words of my Lord are depleted, even if We come with a replenishment the like of it	(18:109)
Say, Surely I am only a mortal the like of you: it is revealed to me that surely your God is only One God.	(18:110)
Yet, in case they disobey you, then *say*, Surely I am quit of whatever you do	(26:216)
As Yusuf *said* to his father, O my father, surely I saw eleven planets and the sun and the moon	(12:4)
He *said*, Surely I am the bondman of Allah; He has brought me the book and made me a Prophet	(19:30)
So *Musa* returned to his *people*, angry and sorrowful. He *said*, O my *people*, did your Lord not promise you a fair promise?	(20:86)
And as Ibrahim *said* to his father and his *people*, surely I myself am completely quit of whatever you worship	(43:26)
Indeed We have already sent Nuh to his *people*; so he *said O* my *people*! Worship Allah! In no way do you have any god other than He	(7: 59)

- *AyAt* ('signs'), *samA* ('heavens'), *arD* ('earth'), *khalaq* ('created'), *ja&al* ('made'): There are many references in the Qur'an to *AyAt* which are normally to be understood as 'signs' in a variety of connected senses. Watt & Bell (1970) identify four different usages for the word: (1) natural phenomena which are signs of God's

power and bounty, (2) events or objects associated with the work of a messenger of God and tending to confirm the truth of the message, (3) signs which are recited by a messenger, and (4) signs which are part of the Qur'an or of the Book. The first two usages of the word are the most popular in the Qur'an and thus the scope of interest here.

The main focus of Meccan *suras* was to call people to Islam, belief in the oneness of Allah, and abandonment of the worship of idols. This oneness of Allah was verified through a series of cosmic signs and phenomena. There is a large number of passages in which phenomena of nature and human life are described as evidence of Allah's power or of the benefits he bestowed on mankind. The signs most frequently cited are: the creation of the heavens and the earth, the creation or generation of man, the various uses and benefits man derives from the animals, the alternation of day and night, the shining of the sun, moon and stars, the changing winds, the sending of the rain from the sky, the movement of the ship on the sea and the stability of the mountains. Less frequently sited phenomena are: shadows, thunder, lightning, iron, fire, hearing, sight, understanding and wisdom.

These sign-passages, where natural phenomena are described as signs of God's power and goodness, were an important element in the Meccan revelations. Mohammed was asked to provide evidence of this to people in Mecca as verification for his message. Conversely, these sign-passages were relatively rare in Medinan revelations. The following are some examples:

Qur'an verse	***Sura*:verse**
And indeed We have already *made* in the heaven constellations and We have adorned it to the onlookers.	(15:16)

Surely in that are *signs* for the scrutinizers	(15:75)
And in no way did We *create* the *heavens* and *earth* and whatever is between them except with the Truth	(15:85)
Do you disbelieve in Him Who *created* you from dust	(18:37)
And who is more unjust than he who, being reminded of the *signs* of his lord, yet veers away from them	(18:57)
And indeed We already showed him all of Our *signs*, yet he cried lies and refused	(20:56)
And indeed in case you ask them, Who *created* the *heavens* and the *earth*? Indeed they will definitely say, The Ever-Mighty, The Ever-Knowing *created* them	(43:9)
The ones who believed in Our *signs* and were Muslims	(43:69)
We *made* for them hearing and beholdings and heart-sights	(46:26)
And in the *earth* are *signs* for the ones having who have certitude	(51:20)

- *Yawm* ('day'), *&aZAb* ('torment'), *wayl* ('woe'): One of the main themes of the Meccan revelations was to remind people of the coming resurrection on the Day of Judgement, when Allah will grant rewards and blessings for his believers and severe punishment and torment for disbelievers. Several references are made to the disbelievers' denial of Mohammed's message, which will result in severe punishment. These ideas are reflected in the use of the above keywords. Examples:

Qur'an verse	***Sura*:verse**
As for him who did injustice, we shall eventually *torment* him, and thereafter he will be reverted to his Lord; then He shall *torment* him with a highly maleficent *torment*	(18:87)
Those are they who have disbelieved in the *signs* of the Lord, and the meeting with Him, so their deeds have been frustrated. Then on the *Day* of the Resurrection We shall not set up for them any weight.	(18:105)

And they say , O *woe* to us! This is the *Day* of Doom	(37:20)
Surely I fear you the *torment* of a tremendous *Day*	(26:135)
Yet they cried him lies; then the *torment* of the *Day* of the Overshadowing took them away; surely it was the *torment* of a tremendous *Day*	(26:189)
The *Day* when neither money nor sons shall profit anyone	(26:88)
So *woe* to the ones who did injustice from the *torment* of a Painful *Day*	(43:65)
And all of them shall come up to Him upon the *Day* of the Resurrection, every one a single person	(19:95)
Surely the ones who err from the way of Allah shall have a strict *torment* for that they have forgotten the *Day* of Reckoning	(38:26)
The *Day* the Trumpet will be blown; and We shall muster the criminals upon that *Day* blue with terror	(20:102)
And on the *Day* when the ones who disbelieved are set before the Fire: Is not this the Truth? They shall say, Yes indeed, by our Lord. He shall say, Then taste the *torment* for that you used to disbelieve	(46:34)

- *KitAb* ('The Book'): In most cases, the word *kitAb* refers to the Qur'an. The word 'Qur'an' was used regularly in the early revelations. However, this was replaced by *Al-KitAb* (the Book) in late Meccan passages. Watt & Bell (1970, 141) state that:

> Perhaps the contrast between 'the Book' and 'the Qur'an' or 'recitation' also implies that the revelations were now written down shortly after they came to Mohammed. Certainly his function is now represented not as that of warning people of punishment but as that of producing a book

Thus in *sura* 19 Mohammed is commanded:

Qur'an verse	***Sura*:verse**
And mention in the *Book* Maryam as she retires from her family	(19:16)

And mention in the *Book* Ibrahim; surely he was constantly sincere, a prophet	(19:42)
And mention in the *Book* Musa; surely he was ever-faithful and he was a Messenger, a Prophet	(19:51)
And mention in the *Book* Ismail; surely he was sincerely (true) this promise; and he was a Messenger, a Prophet	(19:53)
And mention in the *Book* Idris; surely he was constantly sincere, a Prophet	(19:56)

Other passages from the *suras* of this cluster provide a description of the Qur'an as a clear Book full of wisdom and truth from God:

Qur'an Verse	***Sura*:verse**
Alif, Lam, Ra'. Those are the ayat of the *Book* and an Evident Qur'an.	(15:1)
Praise be to Allah Who has sent down upon His slave the *Book*, and has not made to it any crookedness	(18:1)
And recite what has been revealed to you of the *Book* of your Lord; none can alter His words	(18:27)
Those are the signs (verses) of the evident *Book*	(26:2)
A *Book* We have sent down to you, Blessed	(38:29)
The successive sending down of the *Book* is from Allah, The Ever-Mighty, The Ever-Wise	(46:2)
A *Book* whose ayat have been expounded, an Arabic Qur'an for a people who know	(41:3)

Another sense of the word *kitAb* can be distinguished. The word can simply mean 'something written'. On the Day of Judgement, each person will be given his/her *Book* in which all good and bad actions are written. The Book details every single action to award the righteous and punish the wrong-doers accordingly:

Qur'an Verse	*Sura*:verse
And the Book shall be laid down; so you shall see the criminals (feeling) timorous about what is in it, and they say, "O woe to us! How is it with this Book, that it leaves out (nothing), small or great, except that it has enumerated it?" And they shall find whatever they did present, and your Lord does no injustice to anyone	(18:49)

- *RahmAn* ('All-Merciful'): A distinguishing feature of the *suras* of this cluster is the frequent reference to Allah's mercy. The name of Allah, *Al-Rahman* 'The All-Merciful' is repeated 46 times in cluster 3/B out of a total 57 in the entire Qur'an, of which 16 occurrences are in *sura Maryam*. This theme speaks about people to whom Allah shows mercy on the Day of Judgement as a reward for their belief in Allah and for their right-doing, as opposed to another group of people whom Allah will deny mercy as a punishment for their constant denial of the truth and their evil actions. One of the characteristics of later Meccan revelations is the frequent use of the name *Al-RahmAn*, but this was subsequently dropped in Medinan *suras*. As mentioned earlier, Allah was referred to as an unnamed 'God' in early Meccan revelations of cluster A, then as *Al-RahmAn* in later Meccan *suras*, and finally named *AllAh* in Medinan passages of cluster 1/C. Here are some examples of verses featuring the word *RahmAn*:

Qur'an Verse	*Sura*:verse
Gardens of Adn (Eden) that The *All-Merciful* promised His bondmen in the Unseen	(19:61)
Decidedly none is there in the heavens and the earth except that he comes up to The *All-Merciful* as slave	(19:93)
Surely the ones who have believed and done deeds of righteousness, for them The *All-Merciful* shall soon keep	(19:96)

affection	
And surely your Lord is The *All-Merciful*; so closely follow me, and obey my command	(20:90)
The Kingdom, upon that Day, the true Kingdom, shall belong to the *All-Merciful*; and it shall be ever a Day difficult for the disbelievers	(25:26)
And the bondmen of The *All-Merciful* are the ones who walk on the earth gently, and when the ignorant address them, they say, Peace	(25:63)

The foregoing discussions of the three main clusters can be summarized as follows:

- Cluster 1/C is mainly concerned with Muslims' social life and Islamic legal rulings, Allah's omnipotence and uniqueness, and the role of Mohammed as the Messenger of Allah.
- Cluster 2/A is mainly concerned with belief in God, rising from the dead, the horrors of the Day of Judgement, man's ingratitude towards Allah and His favours, and reward and punishment.
- Cluster 3/B is mainly concerned with stories of the prophets and the struggle with their nations, signs of God's unique creative power in the universe, the Day of Judgement, and reward and punishment.

These quantitatively-based findings are qualitatively both corroborated and elaborated by reading and subjective interpretation of the *suras* that comprise the three core clusters. Summaries of these are given below:

SOM cluster 1 / hierarchical cluster C

- *Sura Al-Ahzab* focuses on vital Islamic legal rulings related especially to family affairs. The greater part of the *sura* is devoted to social instructions for the rapidly expanding Muslim society. It deals with such things as the abolition of pagan social customs related to marriage and rules of divorce, adoption and modesty for women, conjugal rights and remarriage, as well as respect for family privacy.
- *Sura Al-Anfal* deals with the code of practice during a state of war and in the battlefield. Instructions are given about fighting, rules on the prisoners of war, the booty, ceasefire, and peace agreements. Belief and reliance on Allah and obeying the Messenger are the keys to success and victory.
- *Sura Al-Baqarah* deals with Islamic legal rulings required for the evolving new Muslim community in Medina. It also highlights a number of important matters which are directly related to people's social life. Most importantly, the *sura* elaborates on family affairs; other social aspects include pilgrimage to the Ka'bah in Mecca, the law in cases of murder, the drawing up of a will by anyone who realizes that death is approaching, fasting, rules of signing contracts in the presence of witnesses, and so on. The *sura* frequently reminds people of the oneness of Allah and urges mankind to worship him alone.
- *Sura Al-Fath* focuses on fighting in the way of Allah, referring specifically to a political victory for Muslims over disbelievers. The *sura* also makes statements about male and female hypocrites who have ill thoughts about Allah but would still offer mercy to those who return to His path. The *sura* also stresses the role of the Messenger who was sent to provide guidance and act as a witness on the Day of Judgement.

- *Sura Al-Hadid* is directed to social rather than individual considerations. It starts by urging Muslims to give generously in the service of Allah. The *sura* also emphasizes that life of this world is a delusion, a passing delight, characterised by greed for more and more wealth and children. We should compete for forgiveness from our lord and to win the Hereafter. The *sura* stresses the importance of establishing justice among people and being fair.
- *Sura Al-Hajj* focuses on one of the five pillars of Islam: pilgrimage to Mecca. It gives details of Islamic legal rulings such as the legislation of pilgrimage, the animals brought for sacrifice during the pilgrimage, and fighting in the cause of Allah. The *sura* appeals to the believers to strive for Allah's sake, establish regular prayers, give in *sadaqah* ('charity'), and do good deeds for the pleasure of Allah and the happiness of mankind.
- *Sura Al-Hashr* urges Muslims to have strong belief in Allah, act righteously, and give preference to others in need over oneself. Hypocrisy and treachery are evils which should be avoided.
- *Sura Al-Hujurat* focuses on social relationships among the community. The *sura* provides a set of values related to courtesy and respectful behaviour towards each other in general and towards the Prophet as Messenger of Allah and leader of the community in particular. It establishes the principles of brotherhood among the Muslim community.
- *Sura Al-Imran* can be divided into two main parts, the first of which deals with the people of the Book, where a special appeal is made to them to accept the new revelation, and the second with Islamic precepts where fighting in the cause of Allah is encouraged. As for material on social and moral behaviour, this occurs in both parts when required by the context. The moral aspect of the *sura* focuses on

warning about the materialistic aspects of the worldly life, encouraging contentment with what one has, helping the needy, lying to be avoided, and being patient as the key to success and victory. The importance of interaction with people and establishing strong relations with friends and family is emphasized. The *sura* also urges Muslims to worship devoutly, pray for forgiveness and be grateful to Allah for His limitless bounties.

- *Sura Al-Maidah* focuses on Islamic legal rulings which mainly highlight the ethical responsibilities and obligations among people. The *sura* addresses believers on nineteen Islamic precepts to be followed: contracts should be fulfilled, laws of cleanliness before prayers and other legal matters related to wills and theft should be implemented.
- *Sura Al-Mujadila* focuses on Islamic legal rulings and manners. Pagan practices related to marriage are severely condemned in this *sura* and women's rights are to be respected. The *sura* refers to other unethical habits such as secret counsels and alliances with the enemies of Allah and His Messenger. Other social and moral values which should enhance the Muslim community are mentioned: hypocrisy is to be avoided, respect to be shown towards each other, and mistakes to be admitted. The *sura* also urges people to prayer, payment of the *zakat* ('almsgiving'), and obedience to Allah and His Messenger.
- *Sura Al-Mumtahana* focuses on social relations with disbelievers and on women converting to Islam. The sworn enemies of Allah who show bitter feelings against Muslims do not deserve friendship. However, we should show respect and fairness towards disbelievers who are not engaged in hostile activities against Islam. The *sura* urges believers to act fairly towards people in general. Women who convert to Islam should be supported and looked after.

- *Sura Al-Munafiqun* highlights the danger and mischief of hypocrites in the community. The *sura* also urges believers to give *sadaqah* ('charity') to attain the pleasure of Allah and not to let worldly gains divert one from the truth nor from remembering Allah.
- S*ura Al-Nahl* uses the metaphor of the bee which produces honey to represent the bounties and healing bestowed by Allah on mankind. Throughout the *sura* we are given evidence of Allah's creativity and omnipotence: He created the heavens and earth, created man form a drop of semen, created cattle for the benefit of people, sends water down from the sky for drinking and for the growth of the vegetation on which the cattle feed, regulates the night and the day, the sun and the moon, the stars and the sea, and the different colours of animals, insects, plants, metals, etc., all over the earth. This *sura* also refers to some legal rulings and moral and social behaviour --that justice should be practised, relatives should be treated with kindness, sexual misconduct is forbidden, arrogance is not compatible with right actions, and children should be loved equally whether they are boys or girls.
- *Sura Al-Nisa* is almost entirely devoted to social issues which were facing the Muslim community in Medina. It addresses Islamic law, the rights of women, and other social issues. It is also punctuated by a large number of moral themes which are vital for the establishment of a healthy society and human relations among people. The *sura* focuses on women and their rights in marriage, family and inheritance and for this reason the *sura* is called *Al-Nisa* (women). The *sura* also calls on believers to believe in Allah alone, His angels, previous revelations, the Messengers, and the Day of Judgement.
- *Sura Al-Nur* focuses on Islamic legal rulings and manners which are of great value to the individual and the community. The *sura* lays down behavioural principles

and concentrates on the family as the nucleus of the society. It addresses a number of moral values which are essential for a healthy society and which can eliminate social and moral decay, anarchy and insolent behaviour. Among these is the problem of sex offenders and slanderers who should be punished severely. The *sura* also tackles other problems related to good manners and moral social values: domestic privacy should be respected, men and women should lower their gazes and guard their private parts, mixing between men and women should be regulated, the chastity of women and men is stressed, men are encouraged to marry poor women to eliminate class barriers, prostitution and adultery are forbidden, marriage is encouraged to produce children and to enhance good social values. The *sura* also encourages the believers to obey and respect Mohammed's commands and instructions. Allah's creative power, which indicates his oneness, is highlighted through a series of phenomena and thus encourages those with intelligence to worship Him alone.

- *Sura Al-Saff* deals with self-sacrifice, physically and financially, in the way of Allah, and unity and bravery in the battlefield. The *sura* encourages dedication to Allah and backing up our belief with good deeds and practising what we preach.
- *Sura Al-Taghabun* focuses on belief in Allah's oneness and creative power. It also warns against the danger of becoming too involved in gaining worldly goods to the extent that devotion and remembrance of Allah are impaired. *Sadaqah* ('charity') and sincere devotion to Allah will increase our credit and enable us to attain Allah's forgiveness.
- *Sura Al-Talaq* focuses on the issue of divorce. The *sura* highlights the importance of family ties and urges believers to be very careful in making a decision to divorce in order to avoid abuse of rights and preserve the sanctity of marriage. It deals with

the regulation of sexual relations, and family and social life. Although permitted, divorce is disliked by Allah --it is permitted but has to follow strict legal rulings. The divorced couple should always establish a relationship based on mutual respect and cooperation.

- *Sura Al-Tawba* tackles the principles of Islamic social behaviour and principles of reform. *Zakat* ('almsgiving') should be given to the poor, the needy, the wayfarer, those who are employed to collect and distribute it, for freeing captors and debtors, and those possible converts whose hearts can be won over. The helpless, the sick and those who cannot find anything to give in the cause of Allah are exempted from joining the army for fighting provided that they are sincere in duty towards Allah and His Messenger. Hypocrisy is an evil in society and should be avoided. The *sura* also urges believers to worship Allah directly without any need for intermediaries.
- In *sura Al-Zumr* Allah's existence and oneness are central. Evidence of this theme is found throughout the *sura*: Allah's creative power manifests itself in the creation of the heavens and the earth, the sun and moon, man and animals. He knows the unseen and the visible and He is the creator of everything and is charge of everything. Therefore, worship and sincere devotion are due to Allah alone and any form of idolatry is forbidden.
- S*ura Fatir* highlights Allah's omnipotence and unique creative power. This *sura* resembles *Al-Zumr* and *Al-Nisa* in that is enumerates Allah's blessings and shows how Allah's favour is manifested by giving existence to everything in creation. He creates the heavens and the earth, sun and moon, day and night and man and animals. For these bounties, people need to be thankful to Allah and worship Him alone.

- S*ura Ibrahim* refers to the omnipotence of Allah. He is the Creator of the heaven and earth, sends down water from the sky to bring out fruits, has subjected the ships and rivers to people, and regulates the sun and moon and the night and the day for people. Allah's favours are infinite. One must be grateful, appreciative, remember the favours of Allah, and acknowledge His mercy in order to maintain and multiply these favours. A brief reference is made to one of the other common themes of Medinan *suras*, that believers are commanded to pray and give charity openly and secretly.
- S*ura Luqman* stresses Allah's unique creative power and omnipotence: Allah created the heavens and the earth, placed firm mountains on the earth, created all kinds of animals, sends down rain from the sky, merges the night into the day and the day into the night, regulates the sun and the moon, and knows what is in the wombs. The *sura* also highlights behavioural and social morals: look after and show gratitude to your parents, bear patiently whatever may afflict you, do not sneer at other people, do not walk insolently and be moderate in your pace when you walk, and lower your voice when you talk.

<u>SOM cluster 2 / hierarchical cluster A</u>

- *Sura Al-Adiyat* focuses on man's ingratitude to Allah and to all his bounties. Man's thanklessness and ingratitude are reflected in a host of actions and verbal statements which will serve as witness against him on the Day of Judgement. The *sura* presents a violent and frightening scene on the Day of Judgement in which we witness the 'scattering about' of the contents of the graves and the bringing out of the secrets of the hearts which were closely guarded.

- *Sura Al-Fajr* focuses on the Day of Judgement and reward and punishment. It portrays the horror of the Day of Judgement when the earth will be completely flattened, Allah will appear with the angels in rows, and Hell will be opened. On that Day man will remember his passion for worldly riches but it will be too late. Punishment will be awaiting the wrong-doers while the believers will be rewarded.
- *Sura Al Falaq* focuses on seeking refuge in Allah from every evil creature, evil plots, and envy of people.
- *Sura Al Feel* focuses on the annihilation of the rejectors of truth. The *sura* describes a historical event which took place prior to the birth of Mohammed when Abrah invaded Mecca with a large army of elephants but Allah defeated his evil plot.
- *Sura Al Kawthar* highlights the infinite bounties Allah bestows upon us. In return, we have to show gratitude and sincere devotion to Him for His mercy.
- *Sura Al-Mursalat* is emphatic that the rising from the dead will take place on the Day of Judgement. On this day, the stars will become dim, the sky split open, and the mountains will be scattered like dust. Also on that day, the good-doers will be rewarded in the garden of eternal bliss and the wrong-doers will be punished in the blazing fire. The *sura* also provides examples of Allah's omnipotence to show that Allah who created man is able to raise up the dead and that the rejectors of the truth will be proved wrong.
- *Sura Al-Qiyamat* focuses on the rising from the dead, the Day of Judgement, reward and punishment. The *sura* is emphatic that the rising from the dead is not an illusion: Allah who created us will be able to bring us back to life when we are dead. On the day of Judgement, eyesight will be lost due to the horrors of that day which is marked by the eclipse of the moon. People will then be brought for judgement.

- *Sura Al-Rahman*: It has been disputed whether this *sura* has been revealed in Mecca or Medina. Its subject matter bears closer resemblance with the Meccan *suras* than with the Medinan ones. It focuses on the revelation, Allah's infinite favours bestowed on mankind, Allah's unique creative power, and the Day of Judgement. The *sura* could be divided into four distinct parts: the first speaks about creation and origins, the second discusses annihilation, resurrection and reckoning, the third talks about the highest reward for those who excel in this existence and the fourth is about the reward of the ordinary believers. Again, all four themes are identical with the major themes of Meccan revelations.
- *Sura Al Shams* starts with a description of Allah's creative power. It then, puts man's fate in his own hands and makes him responsible for it. The *sura* highlights the fact that everyone is endowed with free will and that it is our decision to choose the right or wrong way. The wrath of Allah is the result of acting immorally. In conclusion, Allah is the Lord of man, the universe and fate.
- *Sura Al Sharh* lists Allah's bounties bestowed on the Prophet in particular and on mankind in general. It stresses that man should have patience and faith in Allah which helps overcome difficulties and hardship in life.
- *Sura Al Takwir* focuses on the horrors of the Day of Judgement and reward and punishment. It describes in detail the violent cosmic events that will take occur before the Day of Judgement and the division of people to face their final fates. The sun will implode, the stars will lose their light, the mountains will be swept away, the beasts will be herded together, the oceans will overflow and flood the land, souls will be reunited with bodies, the sky will be stripped bare and everyone will be presented with their book of actions.

- *Sura Al-Teen* emphasizes the elevated status of mankind over other creation. Allah created man in the finest form and is endowed with the best faculties. In return, man should be thankful to Allah. Deviating from His path will result in humiliation and punishment on the Day of Judgement which is inevitable.
- *Sura Al-Zalzalah* highlights the horror and convulsion of the Day of Judgement. This *sura*, as its name 'The Earthquake' suggests, features the Day of Judgement when the earth will be shaken and will throw up what is inside it, such as the dead. People will be shown their deeds for which they will be rewarded or punished.
- *Sura Quraish* focuses on the favours of Allah on mankind and specifically on the tribe of Quraish. People who are granted these favours should be grateful to Allah and their gratitude should be reflected in worshipping Him alone.

<u>SOM cluster 3 / hierarchical cluster B</u>

- *Sura Al-Anbiyaa* ('The prophets') was one of the last *suras* to be revealed in the Meccan period. Its name derives from the fact that sixteen prophets are mentioned together with brief reference to their lives. The prophets are: Musa, Harun, Lut, Nuh, Ayyub, Ismail, Idris, Zakaria, Yahya, Yunus, Isa, Ibrahim, Dawud, Sulayman, Dhul-kifl and Mohammed. The stories of the previous prophets tell of their struggles against unbelief and different forms of evil and the rejection of truth by their communities. The *sura* also describes the horror of the Day of Judgement and reminds those who disputed its occurrence of the painful torment awaiting them on that Day. It also refers to Allah's omnipotence, which is evidenced through a series of signs in the cosmic system: the creation of the sky and earth, day and night, sun and moon and mountains and water.

- *Sura Al-Hijr* gives details of the arrogance and opposition of previous nations who rejected the revelation and prophethood and sneered at their prophets. References are made to some prophets such as Ibrahim, Lut, Shu'ayb, and Salih whose nations were eventually afflicted by the wrath of Allah while their prophets and their supporters were saved by Allah. It also refers to the story of Adam and Shaytan which symbolises guidance and misguidance. The *sura* provides details of the cosmic signs which are meant as a representation of the omnipotence of Allah and His unique creative power.
- *Sura Al-Kahf* presents a number of stories. First it introduces the miraculous story of a group of disbelievers who slept in a cave for 309 years and thought that they had only slept for a few hours. It then recounts a story of Musa and his eagerness for knowledge. The third story in the *sura* is that of the 'two-horned' man Dhul-Qarnayn, who was a powerful and just ruler. A brief reference is also made to the story of Shaytan and his rebellious arrogance. As is the case in every *sura*, the stories mentioned in this *sura* have the same objective, namely exhortation to right conduct. The *sura* also refers to the signs Allah provided as proof of His oneness. Those signs, prophets and messages were all mocked by the disbelievers and hence will be punished in fire on the Day of Judgement. The believers on the other hand will be rewarded in the Garden.
- *Sura Al-Saffat* recounts six earlier stories of prophets and their messages to Prophet Mohammed to comfort him and strengthen his heart. The first is that of Nuh who endured long affliction for the sake of Allah. Then there is the story of Ibrahim and his relationship with his son and which proves that faith is not simply words but means steadfastness and submission. This is followed by the story of Musa who received the Book which presented the religion as creed, law, and governance

which gives it a certain resemblance to the message of Mohammed. The *sura* also refers to Lut, Ilyas and Yunus. The Day of Judgment and reward and punishment are major themes in this *sura*. It deals with disbelievers' scepticism about the rising from the dead. On the Day of Judgement, they will admit their mistake and will taste painful torment. The believers, however, will be rewarded with the Garden.

- *Sura Al-Shu'raa* ('the poets') derives its title from the brief reference to poets and the allegations made by the disbelievers against Prophet Mohammed, that he was a poet and that the Qur'an was poetry. The *sura* also presents stories about earlier nations: Musa and Pharaoh, Ibrahim with his people, and also the stories of Ad and Thamud, the people of Lut and the people of Madyan. The *sura* is emphatic that the Qur'an is a revelation from Allah representing truth and guidance despite the disbelievers' continuous opposition. They are warned about the painful torment awaiting them on the Day of Judgement, while the Gardens will be brought near for the believers.
- *Sura Al-Zariyat* presents the story of Ibrahim as evidence of Allah's power. Reference is made to earlier generations and their fates are given: the people of Sodom and Gomorrah and Firawn, and the tribes of Ad and Thamud. The *sura* also makes reference to numerous signs of Allah's unity and omnipotence: the ships which sail in rivers and seas, the formation of rain from the wind and clouds, the construction of the universe and the creation of mankind. The theme of reward and punishment on the Day of Judgement is highlighted; believers will be rewarded in Gardens while disbelievers will face painful torment.
- *Sura Al-Zukhruf* reassures the Prophet that he is on the straight way and reminds him of how the prophets of old struggled with their nations and how they were rejected and persecuted. Prophets such as Ibrahim, Musa, and Isa were mortal men

chosen by Allah to serve as His messengers, to advise their nations to worship Allah alone. However, most of them were ridiculed and rejected. The *sura* also describes the Qur'an as a clear book full of wisdom, yet the disbelievers still reject it. Evidence and signs of Allah's unity and creative power are illustrated in this *sura*: He created the heavens and earth, He sends rain from the sky to revive dead land, He created everything in pairs and made ships and cattle. Despite these clear signs, the disbelievers reject the truth and will therefore receive a painful torment in Hell while the believers will enjoy the bliss of the Garden.

- *Sura Maryam* is based on the story of Maryam and Isa. Through the story of Maryam and that of Zakaria, the *sura* provides an argument for Allah's uniqueness, omnipotence and perfection. We are also told of the story of Ibrahim and the details of the persecution he had undergone at the hands of his unbelieving people including his own father. The *sura* also briefly refers to other prophets such as Nuh, Musa, Harun, Ismail, and Idris. The recurring theme of reward and punishment of believers and disbelievers is stressed in this *sura*.
- *Sura Sad* explores the stories of some of the ancient prophets whose people disbelieved and rejected their messages. We are told about Nuh, Hud, Musa, Salih, Lut, Ibrahim, Ishaq, Ya'qub, Dawud, Sulayman and Ayyub. The *sura* also emphasizes that the Qur'an is a revelation from Allah to Mohammed and condemns the disbelievers, rejection to the truth. Detailed descriptions of the delights of the Garden and the horrors of Hell are given as proof of the occurrence of the Day of Judgment and the reward and punishment process.
- *Sura Ta-ha* provides two stories of previous prophets as consolation to the Prophet Mohammed who was challenged by the disbelievers to show them some miracles that could prove is prophethood. The first is the story of Musa and Firawn and the

second is that of Adam and his enemy Satan. The *sura* presents a vivid scene from the Day of Judgement, when believers will be rewarded for their right actions and disbelievers will be punished in Fire for rejecting the truth and associating others with Allah. The *sura* also emphasizes that the Qur'an is a revelation from Allah.

- *Sura Yusuf* presents in detail the full story of Yusuf's life including the severe trials and calamities he had to go through. The story of Yusuf is the only Qur'anic story that is discussed in detail in one single *sura*.

Combining the quantitative and qualitative evidence, the thematic characteristics of the three core clusters can be summarized as follows:

SOM cluster 1 / hierarchical cluster C: The oneness and beneficence of Allah, social relationships in Islam, Islamic legal rulings, rights of women, matters of war, good manners and moral values, divorce and marital affairs and the role of Mohammed as a Messenger of Allah. These themes are specific to Medinan revelations, and the majority of the *suras* in this cluster are Medinan. Five out of the twenty-four core *suras* of this cluster are Meccan (*Al-Nahl, Al-Zumr, Fatir, Ibrahim* and *Luqman*), but, as we have seen, they share Medinan themes.

SOM cluster 2 / hierarchical cluster A: A concise topical indexing of the basic Islamic tenets such as belief in God, belief in the Day of Judgement, man's ingratitude towards Allah and His bounties, the coming resurrection, and reward and punishment of deeds. The main objective of these *suras* is to drive home the reality of the hereafter, its inevitability, and its awesome and serious nature, and to stress its importance to the divine planning of man's life in this world. Such planning culminates in man's death and subsequent resurrection for the life to come. The *suras* of this cluster belong to the First Meccan

Period according to Nöldeke's classification of Qur'anic *suras*[13]. Although they vary in length, they are all quite short and have a distinctive style in terms of their spoken rhythm and the images they evoke.

SOM cluster 3 / hierarchical cluster B: Some of the major themes of these *suras* overlap with those of SOM cluster 2 / hierarchical cluster A--belief in God, revelation, prophethood, rising from the dead, Day of Judgement, reward and punishment-- but this cluster also has one theme unique to itself: stories of earlier prophets. One of the main objectives of these stories is to provide admonition and support to Prophet Mohammed through the outlines of the narratives. They draw comparisons between the resistance of the people of Mecca to the revelation and the way that past nations rejected their Messengers. In addition to the narrative material, the *suras* of this cluster exhibit several signs sections. In particular, 'emphasis is placed on the signs of God's power both in nature and in the events which befell former prophets. The latter are described in a way which brings out their relevance to what was happening to Mohammed and his followers' (Watt & Bell 1970, 110). The core *suras* of this cluster belong the Second Meccan period according to Nöldeke's classification.

Finally, it remains to consider the relationship of the remaining non-core *suras* to the thematic groupings just proposed.

Peripheral *suras*

The foregoing discussion identified some *suras* as peripheral to the core clusters in the sense that, for a given *sura* S_p and a given cluster C_{core}, all the hierarchical analyses assign

[13] The German scholar, Theodor Nöldeke, in his *Geschichte des Qorans* (History of the Qur'an), first published in 1860 and revised by Friedrich Schwally in 1909, classified the Qur'anic suras into four periods; three Meccan and one Medinan. He followed the Islamic tradition in recognizing a division into suras mainly revealed at Mecca and those mainly revealed at Medina, but also paid careful attention to the style and content of the revelations.

S_p to C_{core} but not all the SOM analyses do, or vice versa. Unsurprisingly, these *suras* are thematically very close to the core clusters relative to which they are peripheral, and in most cases are not easily distinguishable from them, though occasionally some thematic admixture from another cluster can be found.

a) *Peripheral to SOM cluster 1 / hierarchical cluster C*

- *Sura Al-Ankabut:* Deals with characteristic 1/C themes like belief in God and Allah's omnipotence and creative power, but also with themes characteristic of the other clusters: rising from the dead, reward and punishment, the Hereafter (2/A), and belief in prophets and their stories (3/B).
- *Sura Al Ikhlas:* stresses one major 1/C theme, namely, the oneness of Allah.
- *Sura Al-Jumu'a* focuses on characteristic 1/C themes like the Friday prayer and the spiritual, social and ethical values of this weekly meeting. It also highlights the importance of remembering Allah and common public worship over worldly gains and business.
- *Sura Al Tahreem* stresses the value of confidentiality among people and of establishing mutual understanding between husband and wife, both of which are characteristic 1/C themes.
- *Sura Mohammed* focuses on 1/C themes like the establishment of a strong moral and social practice for Muslims, fighting, self-sacrifice and charity for Allah's cause and in defence of faith and freedom. It also, however, deals with 2/A themes: the life of this world is a passing delight, guarding against evil, being heedful and prepared to win eternal happiness in the Hereafter, and the punishment awaiting the disbelievers. A brief reference is made to pervious prophets and their nations (3/B).

b) *Peripheral to SOM cluster 2 / hierarchical cluster A*

These *suras* are relatively numerous, and it would be tedious and repetitive to survey their contents individually, so representative examples are given. In all cases the *suras* are thematically close to 2/A, with occasional admixtures of topics characteristic of other core clusters. *Sura Al-Ghashiya* is one of the names of the Resurrection, and it describes how people will be overwhelmed and stunned by the horror of this event. *Sura Al-Naziat* is emphatic that the rising from the dead is true and beyond doubt. We are told about the horror of the Day of Judgement; a rumbling will be felt, hearts will be pound, and eyes will be downcast. *Sura Al-Qamar* refers to some of the events on the Day of Judgement when the moon will split marking the hour of Judgement; when the trumpet is blown, people will answer that call and come out of their graves like scattered locusts, running in confusion and overwhelmed by shock. *Sura Al-Qari'ah* highlights the horrors of that Day by describing how the mountains will be scattered like bits of carded wool and people will be at total loss and helpless as scattered moths. *Sura Al-Tariq* refers to the creation of man, Allah's ability to bring everyone back to life after death on the Day of Judgement. *Sura Qaf* deals mainly with the disbelievers, scepticism and denial of the Day of Judgement. The *sura* is emphatic that Allah who created man in the first place is able to return him to life after death. The *sura* also provides scenes from the Day of Judgement when every person is accompanied by an angel and a witness and brought forward for reward or punishment. And so on.

c) *Peripheral to SOM cluster 3 / hierarchical cluster B*

- *Sura Al Ahqaf* discusses the debate which took place between the Messenger and the idolaters when the latter were trying to justify their beliefs and conduct. The *sura* later refers to the story of the Jinn (the 'invisible beings') who listened to the

reading of the Qur'an with respect, were impressed by its wisdom and believed in it. The *sura* also lists signs of Allah's creative power and unity. Both are 3/B themes.

- *Sura Fussilat* presents stories of two previous prophets, Hud and Salih, and their struggle against evil and disbelief, and also stresses Allah's unique creation by referring to signs of His power; these are 3/B themes. 2/A themes are scenes from the Day of Judgement, and the reward and punishment.
- *Sura Nuh* is focused on the story of the Prophet Nuh, who was sent by Allah to advise his people to worship Allah alone and seek His forgiveness. The *sura* also mentions the struggles and methods Nuh employed in his prophetic mission (3/B).
- *Sura Saba* refers to the story of Saba, a city in Yemen. The people of Saba were very skilful, prosperous and rich. However, they were constant wrongdoers. They were afflicted by Allah's wrath and were destroyed by a great flood. The *sura* also refers briefly to the story of Dawud and Sulayman. The use of narratives in *suras* is common in 3/B *suras*. This *sura* also glorifies Allah's omniscience and refers to Mohammed as Allah Messenger to mankind. (1/C), and describes the rising from the dead and the occurrence of the Day of Judgement (2/A)

Intermediate *suras*

The foregoing discussion identified the *suras* that the hierarchical analyses assign to more than one of the core clusters, and the SOM analyses either assign to different core clusters or locate in the regions of the maps separating the core clusters from one another. It also noted a strong relationship between the way in which the hierarchical and SOM analyses place these *suras* in relation to the core clusters and to one another. This placement indicates that they are thematically intermediate between the core clusters, and examination

of them shows that this is indeed the case. This is not demonstrated in detail here, however, for two reasons:

- These *suras* are quite numerous, and it would be tedious and repetitive to survey their contents individually.
- Some of them are very short, and the grounds for clustering them are correspondingly slender, as discussed earlier. The suspicion is that these *suras* are 'intermediate' in the sense that they are simply too short to classify reliably.

Chapter Eight

Conclusion

8.1 Conclusion

The research question formulated at the outset of this study asked whether the *suras* of the Qur'an could be classified in a thematically coherent way on the basis of their semantic content. The question was approached using a range of quantitative data preparation and exploratory multivariate analytical techniques, and the result was that a large majority of the *suras* can indeed be classified into three thematically coherent groups. Since exploratory multivariate analysis generates hypotheses, this result constitutes a hypothesis about the thematic structure of the Qur'an. In the currently-dominant paradigm of scientific method (Chalmers 1999; Ladyman 2002), a hypothesis is not a statement of truth in any objective sense, but a collection of substantive claims about some domain of inquiry that can be falsified, and whose validity depends on (i) explaining the domain without self-contradiction, (ii) being supported by evidence from the domain, and (iii) having predictive power in the sense that its claims are compatible with evidence not known when the hypothesis was formulated. The result of this study appears to be valid in terms of (i) and (ii), but what of (iii)? Because the analysis was based on the entire Qur'an and no further evidence can, therefore, be forthcoming, the predictive power of our result would appear to be untestable. This is not the case, however. The clusters identified in this study correspond closely to the classification of the *suras* in the philological tradition of Qur'anic scholarship described earlier in this discussion. In particular:

- The *suras* of core cluster SOM1 / hierarchical cluster C are those that the philological tradition says were composed in the Medinan phase of Mohammed's activity.
- The *suras* of core clusters SOM2 / hierarchical cluster A and SOM3 / hierarchical cluster B are those that the philological tradition says were composed in the Meccan phase of Mohammed's activity.

Because the philologically-based classification was independently developed over many centuries, and because it was not included in the derivation of the results of the present discussion, this close agreement shows that the hypothesis also has predictive power, thus further supporting its validity.

Given that the results of the present study largely corroborate those of traditional philology, what has been gained? The most obvious gain is confirmation: that a completely independent, quantitatively-based methodology arrived at substantially the same result as the philologically-based one strongly supports that on which the two agree. The quantitative results can also be useful in resolution of issues in the philological tradition. There is some disagreement in the latter about whether certain *suras* should be assigned to the Meccan or the Medinan groups, and the quantitatively-based result presented here provides an additional decision criterion for this. The present study has identified the core *suras* of the two groups, and has also shown the relative thematic closeness of the remaining *suras* to the core ones.

In general, therefore, the present discussion can be said to have shown that traditional philological and current quantitative methods can work together fruitfully in interpretation of the Qur'an.

There are three main ways in which the quantitative analysis presented in the foregoing study can be extended:

1. Detailed examination of clusters

The result of this discussion partitions the *suras* into three main thematic clusters. There is no reason in principle why the methodology underlying these partitions should not be used for a more detailed examination of each main cluster with a view to determining the thematic interrelationships of its constituent *suras.* This would, however, require someone with specialist knowledge of Qur'anic interpretation which the present writer does not possess, and as such was not undertaken here.

2. Methodological refinement

- Decisions made at the data creation stage of this study had substantial effects on subsequent analyses. Probably the most important development of the methodology used in this study would be to gain a better understanding of the consequences of normalization for text length, keyword selection, and dimensionality reduction for analysis. A particular issue is the presence of numerous very short *suras*, which were the most unstable in terms of cluster membership across the various analyses.
- The range of analyses can be extended to other exploratory, specifically nonlinear techniques such as multidimensional scaling (Borg & Groenen 2005), Isomap (Tenenbaum *et al.* 2000), and locally linear embedding (Roweis & Saul 2000), and the results compared to those of the present study.
- There is scope for using various analytical methods in conjunction. The quantized vectors generated by SOM training constitute a noise-reduced representation of the data manifold topology, and they can thus be expected to yield more consistent

results across hierarchical analyses than original data vectors (Vesanto & Alhoniemi 2000; Kaski 1997).

3. Diversification

Sura classification is only one of a range of possible quantitative analyses of the text of the Qur'an. Of particular interest would be analyses which allow the semantics of keywords that are of particular thematic significance or of special general interest to be derived from their lexical context, as for example in latent semantic analysis (Landauer *et al.* 1998) and in the SenseClusters and WordNet systems (for example Pedersen & Patwardhan 2006).

References

Abd-Albaqi, M. F. (1987). *Al-Mu'jam al-mufahras li-alfaz al-Qur'an al-karim*, Cairo: Dar Al-Hadith.

Abdel-Haleem, M.A.S. (1993). Context and internal relationships: keys to quranic exegesis: a study of surat al-Rahman (Qur'an chapter 55). In G.R. Hawting and A. A. Shareef (eds.), *Approaches to the Qur'an*, 71-98. London, New York: Routledge.

Abdul-Haleem, M. (1999). *Understanding the Qur'an: themes and style*. London: I.B. Tauris Publishers.

Abdul-Jaleel, N. and Larkey, L. S. (2003). *Statistical transliteration for English-Arabic cross language information retrieval.*. In the Conference on Information and Knowledge Management (CIKM-03), 139-146. New Orleans, USA.

Abdul-Raof, H. (2001a). *Qur'an translation: discourse, texture and exegesis.* Richmond: Curzon.

Abdul-Raof, H. (2001b). *The Qur'an outlined: theme and text*. London: Ta-Ha Publishers.

Abdul-Raof, H. (2003*). Exploring the Qur'an*. Dundee: Al-Maktoum Institute Academic Press.

Abu-Salem, H., Al-Omari, M., and Evans, M. (1999). Stemming methodologies over individual query words for an Arabic information retrieval system. *Journal of the American Society for Information Science* 50 (6), 524-529.

Ahmad, F., Yusoff, M., and Sembok, T. M. T. (1996). Experiments with a stemming algorithm for Malay words. *Journal of the American Society for Information Science* 47 (12), 909-918.

Akbar, M. (1978). *The meaning of the Qur'an*, Lahore: Islamic Publications Ltd.

Al-Alusi, S. M. (1855). *Ruh Al-Ma'ani*. Volumes 1-15. Beirut: Dar Al-Kutub Al-Ilmiyyah.

Al-Andalusi , A. (1993). *Al-Bahr Al-Muhit.* Also called 'Tafsir Abu Hayyan', 8 volumes. Beirut: Dar Al-Kutub Al-Ilmiyyah.

Al-Baghawi, A. (1985). *Ma'alim Al-Tanzil.* Volumes 1-4. Beirut: Dar Al-Ma'rifah.

Al-Fedaghi, S.S. and Al-Anzi, F.S. (1989). *A new algorithm to generate root-pattern forms.* In Proceedings of the 11th National Computer Conference, 391-400. King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Al-Ghazali, M. (1998). *A journey through the Qur'an: themes and messages of the Holy Qur'an.* London: Dar Al-Taqwa.

Al-Ghazali, M. (2000). *A thematic commentary of the Qur'an.* Institute of Islamic Thought. Herdon, Va: Richmond.

Ali, A. Y. (1983). *The Holy Qur'an: Text, Translation, and Commentar.* Maryland: Amana Corp.

Aljlayl, M. and Frieder, O. (2002). *On Arabic search: Improving the retrieval effectiveness via a light stemming approach.* In Proceedings of the 11th ACM International Conference on Information and Knowledge Management, (CIKM-02), 4-9. Mclean, VA., USA.

Al-Kharashi, I. A. and Evans, M. W. (1994). Comparing words, stems, and roots as index terms in an Arabic information retrieval system. *Journal of the American Society for Information Science* 45 (8), 548-560.

Al-Kharashi, I. and Al-Sughaiyer, I. (2002). *Data set for designing and testing an Arabic stemmer*. In Proceedings of Arabic Language Resources and Evaluation: Status and Prospects, Spain.

Allinson, N., Yin, H., Allinson, L., and Slack, J. (2001). (eds.). *Advances in self organising maps*. New York: Springer Verlag.

Al-Mahalli, J. and Al-Suyuti, J. (1989*). Al-Jalalayn*. Beirut: Dar Ibn Kathir.

Al-Qurtubi, A. (1997). *Al-Jami' Liahkam Al-Qur'an*. Volumes 1-20. Beirut: Dar Al-Kitab Al-Arabi.

Al-Razi, M. (1981). *Mafateeh Al-Ghayb*. Volumes 1-16. Beirut: Dar Al-Kutub Al-Ilmiyyah.

Al-Shalabi, R. (1996). *Design and implementation of an Arabic morphological system to support natural language processing*. PhD thesis, Computer Science, Illinois Institute of Technology, Chicago.

Al-Shinqiti, M. (1996). *Athwa' Al-Bayan*. Volumes 1-10. Beirut: Dar Al-Kutub Al-Ilmiyyah.

Al-Sughaiyer, I. and Al-Kharashi, I. (2004). Arabic morphological analysis techniques: a comprehensive survey. *Journal of the American Society for Information Science and Technology* 55 (3), 189-213.

Al-Suwaynea, A. S. (1994). *Information Retrieval in Arabic* [In Arabic]. Riyadh, Saudi Arabia, King Fahh National Library.

Al-Tabari, A. (1968). *Jami' Al-Bayan fi Ta'wil Ayi Al-Qur'an*. Volumes 1-30. Beirut: Dar Al-Qalam.

Al-Zamakhshari, A. (1995). *Al-Kashshaf*. Volumes 1-4. Beirut: Dar Al-Kutub Al-Ilmiyyah.

Al-Zarkashi, B. (1988). *Al-Burhan fi Ulum al-Qur'an*. Volumes1-4. Beirut: Dar al-Kutub al-Ilmiyyah.

Arabie, P., Hubert, L., and de Soete, G. (1992). (eds.) *Clustering and classification*. River Edge, New Jersey: World Scientific Press.

Arberry, A. (1980). *The Koran interpreted*. London: George Allen & Unwin.

Asad, M. (1980). *The message of the Qur'an*. Gibraltar: Dar Al-Andalus.

Asian, J., Williams, H., and Tahaghoghi, S. (2005). *Stemming Indonesian.* In Proceedings of the twenty-eighth Australian Computer Science Conference, ACS, 307-314, Newcastle, Australia. CRPIT, 38. Estivill-Castro, V., ed.

Baayen, R. (2001). *Word frequency distributions*. Dordrecht: Kluwer

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*. Addison-Wesley.

Bauer, H., Herrmann, M., and Villmann, T. (1999). Neural maps and topographic vector quantization. *Neural Networks* 12, 659-676.

Beesley, K.(1996). *Arabic finite-state morphological analysis and generation.* In Proceedings of COLING-96, the 16th International Conference on Computational Linguistics, vol. (1), 89-94, Copenhagen, Denmark.

Belew, R. (2000). *Finding out about: A cognitive perspective in search engine technology and the WWW*. Cambridge: Cambridge University Press.

Bell, R. (1937). *The Qur'an: Translated with a critical re-arrangement of the surahs*. Vol. 1 & 2. Edinburgh: T. & T. Clark.

Bellmann, R. (1961). *Adaptive control processes: a guided tour*. Princeton University Press.

Berg, H. (2001). Computers and the Qur'ān. In J. McAuliffe (ed.), *Encyclopaedia of the Qur'ān.* Vol. 1, 391-395. Leiden-Boston-Köln: Brill.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press.

Bishop, C., Svensen, M., and Williams, C. (1998). The generative topographic mapping. *Neural Computation* 10(1), 215-234.

Bookstein, A. and Kraft, D. (1977). Operations research applied to document indexing and retrieval decisions. *Journal of the Association of Computing Machinery* 24 (3), 410-427.

Bookstein, A. and Swanson, D.R., (1974). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science* 25, 312-318.

Borg, I. and Groenen, P. (2005). *Modern multidimensional scaling: Theory and applications*. 2nd ed. New York: Springer-Verlag.

Buckley, C. (1993). The importance of proper weighting methods. In M. Bates (ed.) *Human Language Technology.* San Mateo, CA: Morgan Kaufmann. Alternative ref: In ARPA Workshop on Human Language Technology, Morgan Kaufmann, 1993, 349-52. Princeton, New Jersey.

Buckwalter, T. (2002). *Buckwalter Arabic Morphological Analyzer.* Version1.0, Linguistic Data Consortium (LDC), catalog number LDC2002L49 and ISBN 1-58563-257-0.

Carstairs, M. (2002). *An Introduction to Englich Morphology: words and their structure*. Edinburgh: Edinburgh University Press.

Chalmers, A. F. (1999). *What is this thing called Science*? 3rd edition. UK: OpenUniversity Press.

Chen, A. and Gey, F. (2002). *Building an Arabic stemmer for information retrieval*. In Proceedings of the 11th Text Retrieval Conference (TREC-02), Gaithersburg, Maryland.

Church, K. and Gale, W. (1995a). Poisson mixtures. *Natural Language Engineering* 1, 163-90.

Church, K. and Gale, W. (1995b). *Inverse Document Frequency (IDF): A Measure of Deviation from Poisson*. In Proceedings of the Third Workshop on Very Large Corpora, 121-130.

Clarke, G. M. and Cooke, D. (1998*). A basic course in statistics*. 4th ed., London: Arnold.

Cottrell, M., de Bodt, E., Verleysen, M. (2001) A statistical tool to assess the reliability of self-organizing maps, In: N. Allinson, H. Yin, L. Allinson, J. Slack (eds.), Advances in Self-Organizing Maps, Springer, 7-14

Croft, W. and Harper, D. (1979). Using probabilistic models of information retrieval without relevance information. *Journal of Documentation* 35, 285-295.

Dale, R., Moisl, H. and Somers, H. (2000). (eds.) *Handbook of Natural Language Processing*. New York:Marcel Dekker.

Darwish, K. (2002a). *Al-Stem: A light Arabic stemmer* [Online]. Available at: http://www.glue.umd.edu/~kareem/research.

Darwish, K. (2002b). *Building a shallow morphological analyzer in one day*. In Proceedings of the Association of Computational Linguistics (ACL-02), 47-54, University of Pennsylvania, Philadelphia.

de Bodt, E., Verleysen, M., and Cottrell, M. (1997). *Kohonen maps versus vector quantization for data analysis*. In Proceedings of the 5th European Symposium on Neural Networks, ESANN-97, 187-193, D-Facto public, Brussels.

Dror, J., Shaharabani, D., Talmon, R., and Wintner, S. (2004). Morphological analysis of the Qur'an. *Literary and Linguistic Computing*, 19(4), 431-452.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification*. 2nd ed. New York: John Wiley & Sone.

El-Awa, S. (2002). *Textual relations in the Qur'an*. PhD thesis, London: University of London.

Everitt, B. S. and Dunn, G. (2001). *Applied multivariate data analysis*. London: Arnold.

Everitt, B. S., Landau, S., and Leese, M. (2001). *Cluster Analysis*. 4th ed. London: Arnold.

Fodor, J. (2001). Language, Thought, and Compositionality. *Mind and Language* 16, 1–15.

Fodor, J. and Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critique. *Cognition,* 28, 3–71.

Frakes, W. (1992). Stemming Algorithms. In W. Frakes & R. Baeza-Yates (eds.), *Information Retrieval*, 131-60.

Frakes, W. and Baeza-Yates, R. (1992). (eds.) *Information retrieval: Data structures and algorithms*. Prentice Hall.

Fraleigh, J. and Beauregard, R. (1995). *Linear Algebra*. 2nd ed. Menlo Park, CA: Addison-Wesley.

Fuller, M. and Zobel, J. (1998). *Conflation-based comparison of stemming algorithms*. In Proceedings of the third Australian Document Computing Symposium, Sydney, Australia.

Ghali, M. (1997). *Towards understanding the ever-glorious Qur'an*. Cairo: Publishing House for Universities.

Gordon, A. (1987). A review of hierarchical classification. *Journal of the Royal Ststaistical Society Series*, A 150(2), 119-37.

Gordon, A. (1992). Hierarchical classification. In P.Arabie, L. Hubert and G. de Soete (eds.), *Clustering and Classification*. River Edge, New Jersey: World Scientific Press.

Gordon, A. (1999). *Classification*. 2nd ed., London: Chapman & Hall.

Gore, P. (2000). Cluster Analysis. In Tinlsey Brown (2000), 297-321

Goweder, A. (2004). *Stemming and Arabic information retrieval : the case of broken plurals*. PhD thesis, Department of Computer Science, University of Essex.

Goweder, A., Poesio, M., and De Roeck, A. (2004). *Broken plural detection for Arabic information retrieval*. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, 566-567, Sheffield, UK, July, 25-29.

Grimm, L. and Yarnold, P. (1995). *Reading and understanding multivariate statistics*. Washington DC: American Psychological Association.

Grimm, L. and Yarnold, P. (2000). *Reading and understanding more multivariate statistics*. Washington, DC: American Psychological Association.

Gurney, K. (1997). *An introduction to neural networks*. Routledge.

Hair, J., Anderson, R., Tatham, R., and Black, W. (1998). *Multivariate data analysis*. 5th ed. New Jersey, USA: Prentice-Hall International.

Hammer, B. and Villmann, T. (2001). Estimating relevant input dimensions for self-organizing algorithms. In N. Allinson, H. Yin, L. Allinson and J. Slack (eds.), *Advances in self-organising maps*, 173-80. New York: Springer Verlag.

Hastie, T. and Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association* 84, 502-16.

Haykin, S. (1999). *Neural networks: a comprehensive foundation*. Prentice Hall.

Hegazi, N. H. and El-Sharkawi, A. A. (1985). *An approach to a computerised lexical analyzer for natural Arabic text*. In Proceedings of the Arabic Language Conference, Kuwait.

Hegazi, N. H. and El-Sharkawi, A. A. (1986). *Natural Arabic language processing*. In Proceedings of the National Computer Conference, Volume 2 (pp.10-5-1-10-5-17), Riyadh, Saudi Arabia.

Hilborn, R. C. (2000). *Chaos and nonlinear dynamics*. Oxford: Oxford University Press.

Hollink, V., Kamps, C., Monz, C., and de Rijke, M. (2003). Monolingual document retrieval for European languages. *Information Retrieval* (6).

Hull, D. (1996). Stemming algorithms- a case study for detailed evaluation. *Journal of the American Society for Information Science* 47 (1), 70-84.

Ibn-Al-Jawzi, A. (1964). *Zad Al-Masir fi Ilm Al-Tafsir*. Volumes 1-9. Beirut: Al-Maktab Al-Islami.

Ibn-Attiyah, A. (1977). *Al-Muharrir Al-Wajiz*. Volumes 1-4. Duha: Wizarat Al-Awqaf.

Ibn-Kathir, I. (1970). *Tafsir Al-Qur'an Al-Adhim*. Volumes 1-7. Beirut: Dar Al-Fikr.

Irving, T. (1979). Terms and concepts: problems in translating the Qur'an. In K. Ahmead and Z. I. Ansari (eds.), *Islamic Perspectives: Studies in Honour of Mawlana Sayyid Abul A'la Mawdudi*, 121-134, Leceister: The Islamic Foundation.

Jackson, J. (2003). *A user's guide to principal components*. New York: Wiley-Interscience.

Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice-Hall.

Jain, A. Murty, M., and Flynn, P. (1999). Data clustering: a review. *ACM Computing Surveys* 31(3), 264-323.

Jolliffe, I. (2002). *Principal component analysis*. 2nd ed., Springer: New York.

Kachigan, S. (1991). *Multivariate statistical analysis: A conceptual introduction*. Radius Press. New York.

Kaski, S. (1997). *Data exploration using self-organizing maps*. PhD thesis, Helsinki University of Technology.

Kaski, S., Kangas, J., and Kohonen, T. (1998). Bibliography of Self-Organizing Map (SOM) Papers: 1981-1997, *Neural Computing Surveys* 1, 102-350.

Kaski, S. and Lagus, K. (1996). *Comparing self-organizing maps.* In Proceedings of the International Conference on Artificial Neural Networks (ICANN-96), 809-814.

Kaski, S., Nikkila, J., and Kohonen, T. (2000). *Methods for exploratory cluster analysis*. In Proceedings on the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, SSGRR 2000, Italy.

Khoja, S. and Garside, R. (1999). *Stemming Arabic text*. Computing Department, Lancaster University, Lancaster, U.K. Retrieved February, 9th 2004 from http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps

Kiraz, G. (1996). *Analysis of the Arabic broken plural and diminutive*. In Proceedings of the 5th International Conference and Exhibition on Multi-Lingual Computing, Cambridge.

Kohonen, T. (2001). *Self-organizing maps*. 3rd ed. Springer.

Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., and Saarela, A. (2000). *Self organization of a massive text document collection*. In IEEE Transactions on Neural Networks 11, 574-85.

Kraaij, W. and Pohlman, R. (1994). Porter's stemming algorithm for Dutch. In L. G. M. Noordman and W. A. M. de Vroomen (eds.), Informatiewetenschaap1994: Wetenschappelijke bijdragen aan de derde STINFON Conferentie, Tilburg, 167-180.

Kraaij, W. and Pohlman, R. (1996). *Viewing stemming as recall enhancements*. In Proceedings of ACM SIGIR-96, 40-48, Zurich, Switzerland.

Krovetz, R. (1993). *Viewing morphology as an inference process*. In Proceedings of ACM SIGIR-93, 191-203, Zurich, Switzerland.

Ladyman, J. (2002). *Understanding Philosophy of Science*. Routledge, London.

Lagus, K. (2002). Text retrieval using self-organized document maps. *Neural Processing Letters* 15(1), 21-29.

Lagus, K., Kaski, S., and Kohonen, T. (2004). Mining massive document collections by the WEBSOM method. *Information Sciences* 163(1-3), 135-156.

Landauer, T., Foltz, P., and Laham, D. (1998). Introduction to Latent Semantic Analysis. *Discourse Processes* 25, 259-284.

Larkey, L. S., Ballesteros, L., and Connell, M.E. (2002). *Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis.* In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-02), 275-282, Tampere, Finland.

Larkey, L. and Connell, M. (2001). *Arabic information retrieval at UMass in TREC-10.* In Proceedings of the 10th Text Retrieval Conference (TREC-01), 562-570, Gaithersburg, NIST.

Larson, R. and Segal, G. (1995). *Knowledge of meaning: An introduction to semantic theory*. MIT Press.

Lebart, L.and Rajman, M. (2000). Computing similarity. In R. Dale, H. Moisl and H. Somers (eds.), *Handbook of Natural Language Processing*, 477-505, New York: Marcel Dekker.

Lovins, J. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11 (1), 22-31.

Luhn, H. (1957). A statistical approach to mechanised encoding and searching of library information. *IBM Journal of Research and Development* 1, 309-317.

Luhn, H.(1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2, 159 -165.

MacKay, D. (2003). *Information theory, inference, and learning algorithms*. Cambridge: Cambridge University Press.

Malek, M., (1997). *A study of the Qur'an : the universal guidance for mankind.* Sutton.

Manning, C. and Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Mass: MIT Press.

Maududi, S. (1988). *The meaning of the Qur'an*. Lahore: Islamic Publications Ltd.

Maududi, S. (2005). *Introduction to the study of the Qur'an*. Retrieved April 19th, 2005, from http://www.islam101.com/quran/studyOfQuran.htm

McCarthy, J.and Prince, A. (1990). Foot and word in prosodic morphology: The Arabic broken plural. *Natural Language and Linguistic Theory* 8, 209-282.

Mendelson, B. (1975). *Introduction to topology*. 3rd ed. Boston, MA: Allyn & Bacon.

Merkl, D. (1997). *Exploration of text collections with hierarchical feature maps*. In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR-97, 186-195, Philadelphia, PA, USA, July 27-31.

Merkl, D (1998). Text classification with self-organizing maps: some lessons learned. *Neurocomputing* 21, 61-77.

Merkl, D. (1999). Document classification with self-organizing map. In E.Oja and S. Kaski (eds.), *Kohonen Maps*,183-192. Amesterdam: Elsevier.

Merkl, D. (2000). Text data mining, In R. Dale, H. Moisl and H. Somers (eds.), *Handbook of Natural Language Processing*, 889-903, New York: Marcel Dekker.

Merkl. D. and Rauber, A. (1997a). *Cluster connections: a visualization technique to reveal cluster boundaries in self-organizing maps*. In Proceedings of the 9th Italian Workshop on Neural Nets WIRN-97, Vietri sul Mare, Italy.

Merkl, D. and Rauber, A. (1997b). *Alternative ways for cluster visualization in self-organizing maps*. In Proceedings of the Workshop on Self-Organizing Maps (WSOM-97), Espoo, Finland.

Merkl, D. and Rauber, A. (2000). Document classification with unsupervised artificial neural networks. In F. Crestani and G. Pasi, (eds.), *Soft computing in information retrieval: techniques and applications* 50, 102-121, Heidelberg, Germany: Physica Verlag.

Milton, J. and Arnold, J. (2003). *Introduction to probability and statistics: principles and applications for engineering and the computing sciences*. 4th ed. McGraw-Hill.

Munkres, J. (2000). *Topology*. 2nd ed. Prentice Hall.

Nanji, A. (1991). Islamic Ethics. In P. Singer (ed.), *A Companion to ethics*, 106-118, Oxford: Blackwell Publishers.

Nöldeke, T. (1909). *Geschichte des Korans.* Vol. (1) p.147. Leipzig.

Oakes, M. (1998). *Statistics for corpus linguistics*. Edinburgh: Edinburgh University Press.

Oja, E. and Kaski, S. (1999). (eds.). *Kohonen Maps*. Amsterdam: Elsevier.

Oja, M., Kaski, S., and Kohonen, T. (2001). Bibliography of self-organizing map (SOM) papers: 1998-2001, *Neural Computing Surveys* 3, 1-156.

Omran, E. (1988). Islam, the Qur'an and Arabic. *Al-Serat, Journal of Islamic Studies* 14, retrieved March 21st, 2005, from http://www.al-islam.org/al-serat/

Palmer, D. (2000). Tokenisation and sentence segmentation. In R. Dale, H. Moisl and H. Somers (eds.), *Handbook of Natural Language Processing*, 11-35, New York: Marcel Dekker.

Pedersen, T. and Patwardhand, S. (2006). *Using WordNet based context vectors to estimate the semantic relatedness of concepts*. In Proceedings of the EACL 2006 Workshop Making Sense of Sense - Bringing Computational Linguistics and Psycholinguistics Together, April 4, 2006, Trento, Italy.

Pickthall, M. (1969). *The meaning of the glorious Qur'an: an exploratory translation.* London: George Allen & Unwin Ltd.

Pierce, J. (1980). *An introduction to information theory*. 2nd ed. New York: Dover Publications.

Pölzlbauer, G., Dittenbach, M., and Rauber, A. (2005a). *A visualization technique for Self-Organizing Maps with vector fields to obtain the cluster structure at desired levels of*

detail. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'05) 3 (1558-1563), July 31-August 4, 2005, Montreal, Canada.

Pölzlbauer, G., Dittenbach, M., and Rauber, A. (2005b). *Advanced visualization techniques for self-organizing maps with graph-based methods*. In Proceedings of the Second International Symposium on Neural Networks (ISNN-05), 75-80, May 30 - June 1 2005, Chongquin, China, Springer Verlag.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program* 14 (3) 130-131.

Porter, E. and Gleick, J. (2001). *Nature's Chaos*. Little, Brown.

Pyle, D. (1999). *Data preparation for data mining*. San Francisco, CA: Morgan Kaufmann Publishers.

Qutub, S. (1982). *Fi Thilal Al-Qur'an (In the Shade of the Qur'an)*.. Volumes 1-6. Cairo: Dar Al-Shuruq.

Rietveld, A. and van Hout, R. (1993). *Statistical techniques for the study of language and language behaviour*. Berlin: Mouton de Gruyter.

Ritter, H., Martinetz, T., and Schulten, K. (1992). *Neural computation and self-organizing maps*. Addison-Wesley.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* 60, 503-520.

Robertson, S. and Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, 129-46.

Robertson, S. and Spärck Jones, K. (2004). IDF term weighting and IR research lessons. *Journal of Documentation* 60, 521-523.

Robinson, N. (1996). *Discovering the Qur'an: A contemporary approach to a veiled text*. London: SCM Press.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science* 290, 2323-2326.

Salton, G. and Buckley, C. (1988). Term weighting approaches in automatic text retrieval. *Information Processing and Management* 24, 513-523.

Salton, G. and McGill, M. (1983). *Introduction to modern information retrieval*. New York: McGraw-Hill.

Salton, G., Wong, A., and Yang, C. (1975). A vector space model for automatic indexing. *Comunications of the ACM* 18, 613-20.

Sammon, J. (1969). *A nonlinear mapping for data structure analysis*. In IEEE Transactions on Computers C-18 (5), 401-09.

Savoy, J. (1993). Stemming of French words based on grammatical categories. *Journal of the America Society for Information Science* 44, 1-9.

Schinke R., Greengrass, A., and Robertson, P. (1996). A stemming algorithm for Latin text databases. *Journal of Documentation* 52, 172-187.

Scott, D. and Thompson, J. (1983). Probability density estimation in higher dimensions. In J. Gentle (ed.), *Computer Science and Statistics*: Proceedings of the Fifteenth Symposium on the Interface, 173-79, Amesterdam, New York, Oxford: North Holland-Elsevier Science Publishers.

Singhal, A., Buckley, C., and Mitra, M. (1996a). *Pivoted document length normalization.* In Proceedings of the 19th ACM Conference on Research and Development in Information Retrieval (SIGIR-96), 21-29.

Singhal, A. Salton, G., and Buckley, C. (1995). *Length normalization in degraded text collections*. TR95-1507, Department of Computer Science, Cornell University, New York.

Singhal, A., Salton, G., Mitra, M., and Buckley, C. (1996b). Document length normalization. *Information Processing & Management* 32(5), 619-633.

Spärck Jones, K. (1972). Exhaustivity and specificity. *Journal of Documentation* 28, 11-21.

Spärck Jones, K. (1974). Automatic indexing. *Journal of Documentation*, 30(4), 393-432.

Spärck Jones, K. (2004). IDF term weighting and IR research lessons. *Journal of Documentation* 60, 521-523.

Spärck Jones, K., Robertson, S., Hiemstra, D., and Zaragoza, H. (2003). Language modelling and relevance. In W.B. Croft and J. Lafferty (eds.), *Language Modelling for Information Retrieval*, 57-71, Kluwer Academic Publishers.

Spärck Jones, K., Walker, S., and Robertson, S. (2000). A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management* 36, Part 1 779-808, Part 2 809-840.

Sprott, J. C. (2003). *Chaos and Time-Series Analysis*. New York: Oxford University Press.

Tabachnik, B.and Fidell, L. (2001). *Using multivariate statistics*. 4th ed. Boston, MA: Allyn and Bacon.

Thabet, N. (2004). *Stemming the Qur'an*. In Proceedings of Arabic Script-Based Languages Workshop, COLING-04, Geneva, Switzerland, August 2004.

Thabet, N. (2005). *Understanding the thematic structure of the Qur'an: an exploratory multivariate approach*. In Proceedings of the ACL-05 Student Research Workshop, 7-12. Ann Arbor, Michigan.

Tinsley, H. and Brown, S. (2000). (eds.). *Handbook of applied multivariate statistics and mathematical modeling*, Academic Press.

Tukey, J. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.

Turner, C. (1997). *The Qur'an: a new interpretation*. Surrey: Curzon Press.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 2319-2323.

Ultsch, A. (1993). Self-organizing neural networks for visualization and classification. In O.Opitz, B. Lausen, and R. Klar, (eds.), *Information and classification :concepts, methods, and applications*, 307-313, Berlin: Springer-Verlag.

Ultsch, A. (2003a). *U*-matrix: a tool to visualize clusters in high dimensional data.* Technical Report No. 36, Dept. of Mathematics and Computer Science, University of Marburg, Germany.

Ultsch, A. (2003b). *Maps for the visualization of high-dimensional data spaces*. In Proceedings of the Workshop on Self-Organizing Maps (WSOM'03*)*, 225-230, Kyushu, Japan.

Ultsch, A.and Herrmann, L. (2005). *The architecture of emergent self-organizing maps to reduce projection errors*. In Proceedings of the European Symposium on Artificial Neural Networks, ESANN-05, Bruges, Belgium.

Ultsch, A. and Mörchen, F. (2005). *ESOM-maps: tools for clustering, visualization, and classification with emergent SOM.* Technical Report No. 46, Dept. of Mathematics and Computer Science, University of Marburg, Germany.

Ultsch, A.and Siemon, H. (1990). *Kohonen's self organizing feature maps for exploratory data analysis*. In Proceedings of the International Neural Networks Conference (ICNN-90), 305-308, Paris: Kluwer Academic Press.

van Rijsbergen, C. (1979). *Information retrieval*. 2nd ed. London: Butterworths.

Verleysen, M. (1997). Feedforward models. In E. Fiesler and R. Beale (eds.) *Handbook of neural computation*, Oxford University Press, C2.1:1 - C2.1:15.

Verleysen, M. (2003). Learning high-dimensional data. In S. Ablameyko, L. Goras, M. Gori, and V. Piuri, (eds.), *Limitations and future trends in neural computation*, 141-62, IOS Press.

Verleysen, M., François, D., Simon, G., and Wertz, V. (2003). On the effects of dimensionality on data analysis with neural networks. In J. Mira (ed.), *International Work-Conference on Artificial and Natural Neural Networks*, 105-112, Mao, Menorca (Spain), June 3-6, 2003.

Vesanto, J. (1999). SOM-based data visualization methods. *Intelligent Data Analysis* 3, 111-126.

Vesanto, J. (2000). *Using SOM in data mining*, Licentiate's thesis, Helsinki University of Technology. Finland.

Vesanto, J. (2002). *Data exploration process based on the self-organizing map*. PhD thesis, Helsinki University of Technology.

Vesanto, J. and Alhoniemi, E. (2000). *Clustering of the self-organizing map*. In IEEE Transactions on Neural Networks 11(3), 586 – 600.

Villmann, T., Der, R., Herrmann, M., and Martinetz, T. (1994). *A new quantitative measure of topology preservation in Kohonen's feature maps*. In Proceedings of the IEEE International Conference on Neural Networks (ICNN-94), vol. 2, 645-648, Orlando, Florida.

Villmann, T., Der, R., Herrmann, M., and Martinetz, T. (1997). *Topology preservation in self organizing feature maps: exact definition and measurement*. In IEEE Transactions on Neural networks 8(2), 256-266.

Watson, J. (2002). *The phonology and morphology of Arabic*. Oxford: Oxford University Press.

Watt, W. and Bell, R. (1970). *Introduction to the Qur'an*. Edinburgh: Edinburgh University Press.

Welch, A. (1980). Allah and other supernatural beings: the emergence of the Qur'anic doctrine of Tawhid. *Journal of American Academy of Religion Thematic Studies* 47, 733-758.

Welch, A. (1990). The translatability of the Qur'an: literary and theological implications of what the Qur'an says about itself. In D. Goldenberg (ed.) *Translation of Scripture*, 249-285, Philadelphia: Annenberg Research Institute.

Woods, A., Fletcher, P., and Hughes, A. (1986). *Statistics in language studies*. Cambridge: Cambridge University Press.

Xu, J. and Croft, W. (1998). *Corpus-based stemming using co-occurrence of word variants*. In ACM Transactions on Information Systems 16 (1), 61-81.

Xu, J. Fraser, A., and Weischedel, R. (2002). *Empirical studies in strategies for Arabic retrieval.* In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-02), 269-274, Tampere, Finland.

Ziviani, N.and Ribeiro-Neto, B. (1999). Text operations. In. R. Baeza-Yates and B. Ribeiro-Neto (eds.). *Modern information retrieval*, ch.7 Addison-Wesley.

Appendices

Appendix 1: Purpose-built programs

Creation, manipulation, and analysis of the data matrices on which this study is based required a variety of computational tools. Commercial and third-party tools were used for the hierarchical and SOM analyses and for the various kinds of graphics that appear in the text, and these are identified where relevant in the course of discussion. For data creation and manipulation, however, application-specific programs were written by the author. This was done for two main reasons. On the one hand, part of the motivation for undertaking PhD-level research on a computationally-oriented topic was to learn computer programming, and thereby to become less dependent on existing software tools for research purposes; the data creation and manipulation programs described in what follows constitute the practical component of this learning process. And, on the other, the data creation and manipulation requirements for the present study were very application-specific, and it was arguably more convenient to write the requisite programs than to attempt to identify suitable commercial applications.

A fair number of these purpose-built programs were written in the course of the research documented in the foregoing discussion, most of them having to do with low-level manipulation of the Qur'an text files. These are of little interest either conceptually or in terms of program construction, and it was felt to be unnecessary to include them in this appendix. Only the three longest and most important programs are documented here: *StemQuram*, *CreateFreqMatrix*, and *EditMatrix*. Before proceeding to describe these, some general observations about them need to be made.

- All the programs described in what follows are written in the Borland Delphi implementation of object-oriented Pascal. In principle, any of the very numerous

programming languages currently available could have been used, since they are all theoretically equivalent in the sense that all are virtual machines formally equivalent to universal programmable Turing Machines. Where they differ is in their expressiveness and consequently their convenience relative to particular classes of application. For the text processing applications characteristic of disciplines like corpus and computational linguistics and natural language processing, so-called high-level languages that provide text-specific data structures and functions, such as PERL, are the natural choice. Why, therefore, use a relatively low-level, general purpose language like Pascal, and why, given that decision, not use a more 'modern' one like C++ or Java? Since part of the author's aim was to acquire a competence in computer programming, the dissertation supervisor felt that a relatively low level programming language would convey greater familiarity with fundamental data structures, control structures, and program construction principles than a higher-level one would. And why Pascal? Pascal achieved immediate and widespread popularity when it was introduced in the early 1970s for a variety of reasons, the main ones being (i) that it is economical in the sense that only a minimal set of data structure and control primitives are provided, (ii) it is strongly typed, and (iii) variables and subroutines must be declared before being used. The consequence of (i) - (iii) is that Pascal is both easy to learn and enforces structured programming practice; in fact, Pascal was long used as a teaching language. With the addition of contemporary features like object orientation, type inheritance, and an extensive library of graphical facilities, Borland Delphi Pascal seemed the obvious choice, and was therefore selected.

- As noted, Delphi is an object-oriented extension of Pascal, and, given the structuring advantages of object-oriented programming, the natural course would

have been to make use of this facility in the programs described below. That was the aim, but funding for the research reported in this dissertation imposed time constraints such that revision of the programs described below to incorporate object orientation could not be undertaken. As such, these programs have a traditional separation of data structures and subroutines that operate on those structures.

- Because these are research programs designed for use only in the present application, where designer and user are identical, error recovery mechanisms were not felt to be necessary. It is therefore very easy to cause program malfunctions by unexpected or incorrect user input, and if these programs were every developed for general use they would have to be made less fragile.

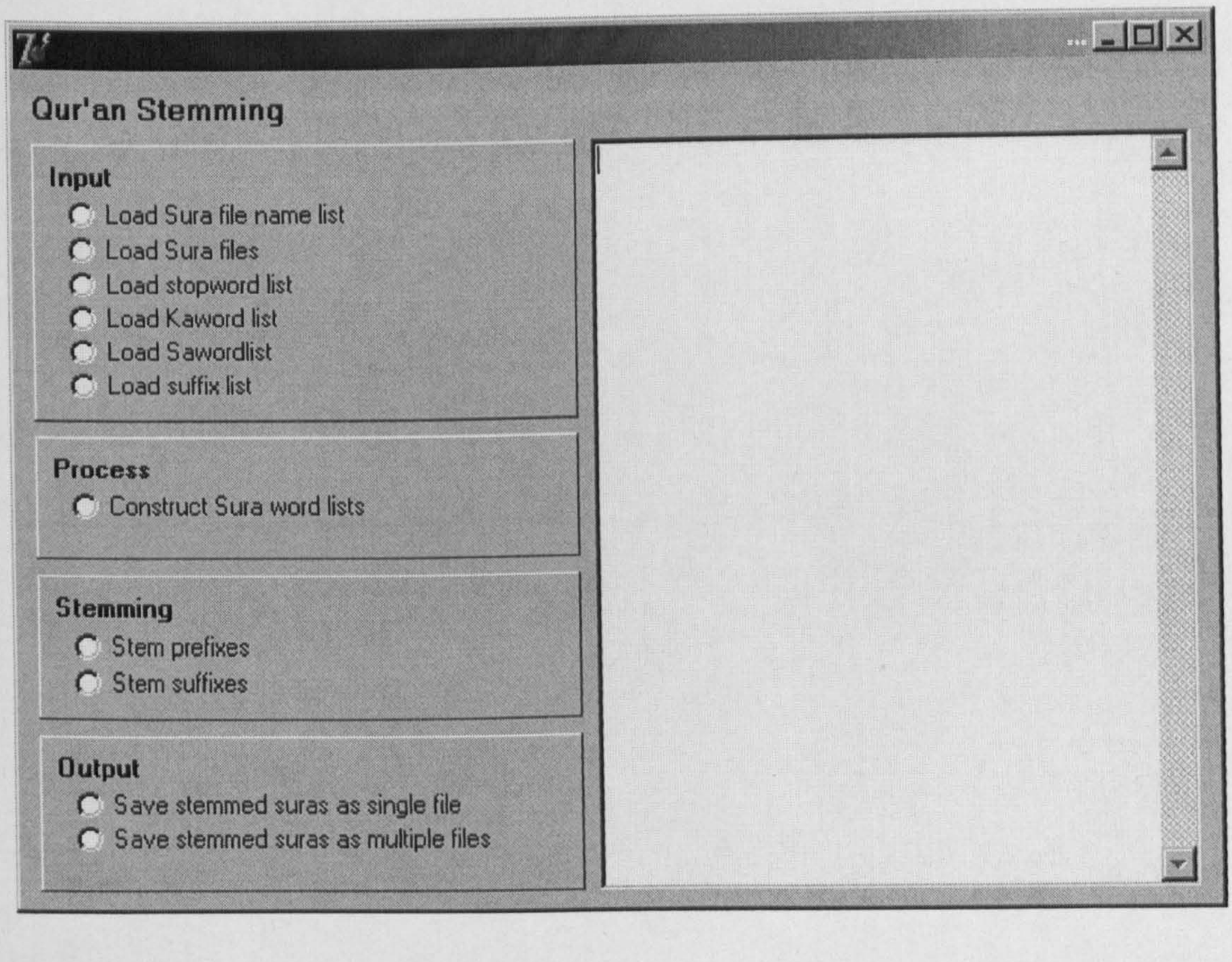
1.1 **StemQuran**

a) Purpose

StemQuran is a light stemmer that stems an electronic copy of the Qur'an in the format described in section 5.2 of the foregoing discussion.

b) Use

The user interface looks like this:



Radio buttons are intended to be used in strict sequence from top to bottom:

1. In the 'Input' box:

   - 'Load sura file name list' inputs an external text file containing a list of the names of the 114 suras in the Qur'an.

- 'Load sura files' inputs from external files and stores in memory the texts of the suras whose names were loaded immediately above.
- 'Load stopword list' inputs from an external text file a list of 'stopwords', that is, words that begin with letter sequences which are candidates for prefix stemming, but for which these letter sequences are in fact not prefixes but part of the stem.
- 'Load Kaword list' inputs from an external file a list of words for which the initial letter sequence 'ka' constitutes a prefix that can be removed.
- 'Load Saword list' inputs from an external file a list of words for which the initial letter sequence 'sa' constitutes a prefix that can be removed.
- 'Load suffix list' inputs from an external file a list of suffixes to be removed in the course of stemming.

2. In the process box:

- 'Construct sura word lists' makes a sura list where each component of the list contains all the lines in a single sura, and each line is a sequence of words (as opposed to each line being a string of characters, as in the text list which this procedure operates on. This list is then used as the basis for stemming.

3. In the 'Stemming' box:

- 'Stem prefixes' removes the prefixes from the sura word lists created in the preceding step.
- 'Stem suffixes' removes the suffixes from the sura word lists created in the preceding step.

4. In the 'Output' box:

- 'Saved stemmed suras as a single file' and 'Save stemmed suras as multiple files' are self-explanatory.

c) Algorithm

The algorithm consists of two main steps:

Step 1: prefix stemming

Nine prefixes are deleted: *wa, fa, la, li, lil, bi, al, ka, sa*. Starting with the first word in the Qur'an and reading each word in the text to the last, when a word *w* beginning with one of the above prefix letter strings str_w is found:

- Associated with each of the first seven prefixes *wa, fa, la, li, lil, bi, al* is a 'stopword' list contain words for which the str_w is not in fact a prefix but part of the stem; this list is given in section 5.2.1.2.2 above. If *w* is in the stopword list, then str_w is not a prefix and no action is taken, that is, *w* remains unaltered in the Qur'an text. If, however, it is not in the stopword list, then str_w is a prefix: it is deleted from *w* and what remains of *w* after deletion is inserted in place of the original *w* in the text. The stopword list for each of these seven prefixes was compiled manually.

- For the remaining two prefixes *ka, sa* the above procedure is reversed, in that the associated stopword lists contain the words that are stemmed rather than, as above, those that are not; these lists are given in section 5.2.1.2.2 above. The lists are searched and, if *w* is found in them, str_w is removed and what remains of *w* is inserted into the text; if *w* is not found, then no action is taken. The reason for this special case is that words whose stems begin with *ka* and *sa* are much more

frequent than words in which they are prefixes, and so it was easier to compile lists of words to stem rather than lists of words not to stem.*ka*.

Step 2: suffix stemming

This step involves the removal of suffixes, which include pronouns, articles, prepositions, etc. Suffixes ranging from length-1 to length-10 were removed; the list of suffixes is given in section 5.2.1.2.2 above.

The procedure is iterative, removing suffixes in decreasing order of suffix length. For each of the suffix lengths from 10 down to 3, the text of the Qur'an is read from beginning to end, one word at a time. Each time a word w is found containing a suffix str_w in the relevant column of the above table, str_w is deleted from w, and what remains of w replaces the original w in the text. This worked well for suffixes of length-3 to length-10, but length-1 and length-2 suffixes are pervasively ambiguous in the sense that they could be part of word stems rather than suffixes. There was no obvious algorithmic way resolving the ambiguity, and so the stemming for these two suffixes was carried out manually.

d) Program listing

```
unit StemQuran;

{StemQuran is a light stemmer that stems an electronic copy of the Qur'an in the
format described in section 5.2 of the foregoing discussion.
The algorithm consists of two main steps:

Step 1: prefix stemming
Nine prefixes are deleted: wa, fa, la, li, lil, bi, al, ka, sa. Starting with the
first word in the Qur'an and reading each word in the text to the last, when a word
w beginning with one of the above prefix letter strings str(w) is found:
  * Associated with each of the first seven prefixes wa, fa, la, li, lil, bi, al
    is a 'stopword' list contain words for which the str(w) is not in fact a prefix
    but part of the stem; this list is given in section 5.2.1.2.2 above. If w is
    in the stopword list, then str(w) is not a prefix and no action is taken, that
    is, w remains unaltered in the Qur'an text. If, however, it is not in the
```

stopword list, then str(w) is a prefix: it is deleted from w and what remains of w after deletion is inserted in place of the original w in the text.
The stopword list for each of these seven prefixes was compiled manually.
* For the remaining two prefixes ka, sa the above procedure is reversed, in that the associated stopword lists contain the words that are stemmed rather than, as above, those that are not; these lists are given in section 5.2.1.2.2 above. The lists are searched and, if w is found in them, str(w) is removed and what remains of w is inserted into the text; if w is not found, then no action is taken. The reason for this special case is that words whose stems begin with ka and sa are much more frequent than words in which they are prefixes, and so it was easier to compile lists of words to stem rather than lists of words not to stem.

Step 2: suffix stemming
This step involves the removal of suffixes, which include pronouns, articles, prepositions, etc. Suffixes ranging from length-1 to length-10 were removed; the list of suffixes is given in section 5.2.1.2.2 above. The procedure is iterative, removing suffixes in decreasing order of suffix length. For each of the suffix lengths from 10 down to 3, the text of the Qur'an is read from beginning to end, one word at a time. Each time a word w is found containing a suffix str(w) in the relevant column of the above table, str(w) is deleted from w, and what remains of w replaces the original w in the text. This worked well for suffixes of length-3 to length-10, but length-1 and length-2 suffixes are pervasively ambiguous in the sense that they could be part of word stems rather than suffixes. There was no obvious algorithmic way resolving the ambiguity, and so the stemming for these two suffixes was carried out manually.}

```
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;
const
  maxfilenamelength = 30;          {Names of sura files}
  maxlinelength = 600;             {Max nrofcharacters in a single sura line}
  maxnrofwordsinline = 150;        {Max nr of words in a single sura line}
  maxtextlength = 600;             {Max nr of lines in a sura}
  maxnroftexts = 150;              {Max nr of suras in the Qu'ran}
  maxwordlength = 20;              {Max length of any word in the Qur'an}
  maxstopwordpairlistlength = 600; {Max nr of stop words}
  maxwordlistlength = 30000;       {Max nr of words in a sura}
  maxsuffixlength = 10;            {Max length of suffix}
  maxsuffixlistlength = 100;
  maxkawordpairlistlength = 100;   {max nr of ka words}
  maxsawordpairlistlength = 100;   {max nr of sa words}

type
 {File names}
 Tstr = packed array [0..(maxfilenamelength - 1)] of char;
 Tfilenamelist = record
                 list : array [0..(maxnroftexts - 1)] of Tstr;
                 length : longint;
```

```
            end;
{Stop words}
Tstopword = packed array [0..(maxwordlength - 1)] of char;
Tstopwordpair = record
             stopword : Tstopword;
             counter : longint;
            end;
Tstopwordlist = record
             list : array [0..(maxstopwordpairlistlength - 1)] of Tstopwordpair;
             length : longint;
             end;
{ka-words}
Tkaword = packed array [0..(maxwordlength -1)] of char;
Tkawordpair = record
             kaword : Tkaword;
             counter : longint;
            end;
Tkawordlist = record
             list : array [0..(maxkawordpairlistlength - 1)] of Tkawordpair;
             length: longint;
            end;
{sa-words}
Tsaword = packed array [0..(maxwordlength -1)] of char;
Tsawordpair = record
             saword : Tsaword;
             counter : longint;
            end;
Tsawordlist = record
             list : array [0..(maxsawordpairlistlength - 1)] of Tsawordpair;
             length: longint;
            end;
{Suffixes}
Tsuffix = record
          str : packed array [0..(maxsuffixlength - 1)] of char;
          length : longint;
         end;
Tsuffixlist = record
             list : array [0..(maxsuffixlistlength - 1)] of Tsuffix;
             length : longint;
             end;
{Text list = the suras in the Qur'an}
Tline = record
         line : packed array [0..(maxlinelength - 1)] of char;
         length : longint;
        end;
Ttext = record
         text : array [0..(maxtextlength - 1)] of Tline;
         name : Tfilename;
         length : longint;
        end;
```

```
Ttextlist = record
          list : array [0..(maxnroftexts - 1)] of Ttext;
          length : longint;
         end;
{List of suras. This differs from Ttextlist in that, where each item in Ttextlist is a sura
consisting  of a set of lines, ie, a set of character sequences / strings, each item in Tsuralistis
a sura containing  a set of word sequences, ie, each text line has been chopped up into
words}
 Tword = record
        wrd : packed array [0..(maxwordlength - 1)] of char;
        suffixstemmed : boolean;
        length : longint;
       end;
Tsuraline = record
           words : array [0..(maxnrofwordsinline - 1)] of Tword;
           linelength : longint;
          end;
Tsura = record
        suraname : Tfilename;
        linelist : array [0..(maxtextlength - 1)] of Tsuraline;
        suralength : longint;
       end;
Tsuralist = record
           list : array [0..(maxnroftexts - 1)] of Tsura;
           length : longint;
          end;
Tstemform = class(TForm)
  Panel1: TPanel;
  Label1: TLabel;
  RadioButton1: TRadioButton;
  RadioButton2: TRadioButton;
  OpenDialog1: TOpenDialog;
  SaveDialog1: TSaveDialog;
  Memo1: TMemo;
  RadioButton3: TRadioButton;
  Label2: TLabel;
  Panel2: TPanel;
  Label3: TLabel;
  RadioButton4: TRadioButton;
  RadioButton5: TRadioButton;
  Label4: TLabel;
  RadioButton17: TRadioButton;
  RadioButton16: TRadioButton;
  Panel3: TPanel;
  Label5: TLabel;
  RadioButton7: TRadioButton;
  RadioButton15: TRadioButton;
  RadioButton6: TRadioButton;
  RadioButton8: TRadioButton;
  procedure RadioButton1Click(Sender: TObject);
```

```
procedure RadioButton2Click(Sender: TObject);
procedure RadioButton3Click(Sender: TObject);
procedure RadioButton4Click(Sender: TObject);
procedure RadioButton5Click(Sender: TObject);
procedure RadioButton7Click(Sender: TObject);
procedure RadioButton15Click(Sender: TObject);
procedure RadioButton17Click(Sender: TObject);
procedure RadioButton16Click(Sender: TObject);
procedure RadioButton6Click(Sender: TObject);
procedure RadioButton8Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;
var
 filenamelist : Tfilenamelist;
 stopwordlist : Tstopwordlist;
 kawordlist : Tkawordlist;
 sawordlist : Tsawordlist;
 suffixlist : Tsuffixlist;
 textlist : Ttextlist;

 suralist : Tsuralist;
 sura : Tsura;

 stopwordpair : Tstopwordpair;
 kawordpair : Tkawordpair;
 sawordpair : Tsawordpair;
 found : boolean;

 outfile : textfile;
 stemfilenamefile : textfile;

 stemform : Tstemform;

implementation

{$R *.dfm}

                    {INPUT ROUTINES}

procedure Tstemform.RadioButton1Click(Sender: TObject);
{Get list of sura file names}
var
 filenamefile : textfile;
 ch : char;
 i, j : longint;
begin
 {Open file containing list of sura file names}
```

```
stemform.opendialog1.execute;
assignfile (filenamefile, stemform.opendialog1.filename);
reset (filenamefile);
i := 0;
{Read to end of file. Each line of the file contains one name. Read these in succession}
while not eof (filenamefile) do
 begin
  {Make a blank. This means that file names will all be uniform, and can thus be compared
if necessary}
  for j := 0 to (maxfilenamelength - 1) do
 filenamelist.list [i,j] := ' ';
 {Read a single file name}
j := 0;
 while (not eoln (filenamefile)) and (not eof (filenamefile)) do
  begin
   read (filenamefile, ch);
   if ch in ['A'..'Z','a'..'z', '&', '0'..'9', ' ', '/', '-', '*', '^','.', ""] then
   filenamelist.list [i,j] := ch;
   j := j + 1;
  end;
 if not eof (filenamefile) then
  readln (filenamefile);
 i := i + 1;
end;
{Record the number of file names read}
 filenamelist.length := i;
 {Output the list to show it's been read}
 {stemform.memo1.lines.add ('File name list:');
 for i := 0 to (filenamelist.length - 1) do
  stemform.memo1.lines.add (filenamelist.list [i]);
  stemform.memo1.lines.add (' ');}
  stemform.Memo1.Lines.Add('File names loaded');
end;

procedure Tstemform.RadioButton2Click(Sender: TObject);
{Get sura files, based on list of file names input earlier and stored in 'filenamelist'}
var
 infile : textfile;
 ch : char;
 i,j,k : longint;
begin
 {Work through the list of file names}
 for i := 0 to (filenamelist.length - 1) do
  begin
   {Assign each successive file name to a disk file}
   assignfile (infile, filenamelist.list [i]);
   reset (infile);
   {Read the current sura, one line at a time}
   j := 0;
    while not eof (infile) do
```

```
begin
 {Text ID, for later reference}
 textlist.list [i].name := filenamelist.list [i];
 {Make a blank line, for uniformity}
 for k := 0 to (maxlinelength - 1) do
  textlist.list [i].text[j].line [k] := ' ';
 {Now read a line}
 k := 0;
 while (not eof (infile)) and (not eoln (infile)) do
  begin
   read (infile, ch);
   if ch in ['A'..'Z','a'..'z', '&', '0'..'9', ' ', '/', '-', '*', '^','.', ""] then
   begin
    textlist.list [i].text[j].line [k] := ch;
    k := k + 1;
   end;
  end;
  {Record the line length}
 textlist.list [i].text [j].length := k;
 if not eof (infile) then
 readln (infile);
 j := j + 1;
end;
{Record the text length}
textlist.list [i].length := j;
 end;
 {Record text list length}
 textlist.length := filenamelist.length;
 stemform.memo1.Lines.add ('sura files loaded');
end;

procedure Tstemform.RadioButton3Click(Sender: TObject);
{Load list of stop words}
var
 sourcefile : textfile;
 ch : char;
 i, j : longint;
begin
 {Open file}
 stemform.opendialog1.execute;
assignfile (sourcefile, stemform.opendialog1.filename);
reset (sourcefile);
{Read the stop word file. Each line of the file contains one word}
i := 0;
while not eof (sourcefile) do
 begin
  {Make a blank}
  for j := 0 to (maxwordlength - 1) do
   stopwordlist.list [i].stopword[j] := ' ';
  {Read a word}
```

```
j := 0;
while (not eoln (sourcefile)) and (not eof (sourcefile)) do
 begin
  read (sourcefile, ch);
  if ch in ['A'..'Z','a'..'z', '&', ' ', '/', '-', '^', '*'] then
   begin
    stopwordlist.list [i].stopword[j] := ch;
    j := j + 1;
   end;
 end;
 if not eof (sourcefile) then
  readln (sourcefile);
 i := i + 1;
end;
{Record list length}
stopwordlist.length := i;
{for i := 0 to (stopwordlist.length - 1) do
 stemform.memo1.lines.add (stopwordlist.list [i].stopword);}
 stemform.Memo1.lines.Add('Stopwordlist loaded');
end;

procedure Tstemform.RadioButton15Click(Sender: TObject);
{Load list of Ka words}
var
 sourcefile : textfile;
 ch : char;
 i, j : longint;
begin
 {Open file}
 stemform.opendialog1.execute;
 assignfile (sourcefile, stemform.opendialog1.filename);
 reset (sourcefile);
 {Read the ka word file. Each line of the file contains one word}
 i := 0;
 while not eof (sourcefile) do
  begin
   {Make a blank}
   for j := 0 to (maxwordlength - 1) do
    kawordlist.list [i].kaword[j] := ' ';
   {Read a word}
   j := 0;
   while (not eoln (sourcefile)) and (not eof (sourcefile)) do
    begin
     read (sourcefile, ch);
     if ch in ['A'..'Z','a'..'z', '&', ' ', '/', '-', '^', '*'] then
      begin
       kawordlist.list [i].kaword[j] := ch;
       j := j + 1;
      end;
    end;
```

```
 if not eof (sourcefile) then
  readln (sourcefile);
 i := i + 1;
end;
 {Record list length}
 kawordlist.length := i;
 {for i := 0 to (kawordlist.length - 1) do
  stemform.memo1.lines.add (kawordlist.list [i].kaword);}
  stemform.Memo1.Lines.Add('Ka word list loaded');
end;

procedure Tstemform.RadioButton6Click(Sender: TObject);
{Load list of Sa words}
var
 sourcefile : textfile;
 ch : char;
 i, j : longint;
begin
 {Open file}
 stemform.opendialog1.execute;
 assignfile (sourcefile, stemform.opendialog1.filename);
 reset (sourcefile);
 {Read the sa word file. Each line of the file contains one word}
 i := 0;
 while not eof (sourcefile) do
begin
 {Make a blank}
 for j := 0 to (maxwordlength - 1) do
  sawordlist.list [i].saword[j] := ' ';
 {Read a word}
 j := 0;
 while (not eoln (sourcefile)) and (not eof (sourcefile)) do
  begin
   read (sourcefile, ch);
   if ch in ['A'..'Z','a'..'z', '&', ' ', '/', '-', '^', '*'] then
    begin
     sawordlist.list [i].saword[j] := ch;
     j := j + 1;
    end;
  end;
 if not eof (sourcefile) then
  readln (sourcefile);
 i := i + 1;
end;
 {Record list length}
 sawordlist.length := i;
 {for i := 0 to (sawordlist.length - 1) do
  stemform.memo1.lines.add (sawordlist.list [i].saword);}
  stemform.Memo1.Lines.Add('Sa word list loaded');
end;
```

```
procedure Tstemform.RadioButton17Click(Sender: TObject);
{Load list of suffixes}
var
 infile : textfile;
 ch : char;
 i, j : longint;
begin
 {Open file containing list of suffixes}
 stemform.opendialog1.execute;
assignfile (infile, stemform.opendialog1.filename);
reset (infile);
i := 0;
{Read to end of file. Each line of the file contains one suffix. Read these in succession}
while not eof (infile) do
 begin
  {Make a blank. This means that file names will all be uniform, and can thus be compared
if necessary}
  for j := 0 to (maxsuffixlength - 1) do
   suffixlist.list [i].str [j] := ' ';
  {Read a single suffix}
  j := 0;
  ch := '*';
  while (not eoln (infile)) and (not eof (infile)) and (ch <> ' ') do
   begin
    read (infile, ch);
    if ch <> ' ' then
     begin
      suffixlist.list [i].str [j] := ch;
      j := j + 1;
     end;
   end;
  suffixlist.list [i].length := j;
  if not eof (infile) then
   readln (infile);
  i := i + 1;
 end;
{Record the number of file names read}
suffixlist.length := i;
{Output the list to show it's been read}
 {stemform.memo1.lines.add ('Suffix list:');
 for i := 0 to (suffixlist.length - 1) do
  stemform.memo1.lines.add (suffixlist.list [i].str + ' ' + inttostr (suffixlist.list [i].length));
 stemform.memo1.lines.add (' ');}
 stemform.Memo1.lines.Add('Suffix list loaded');
end;

                    {ANALYSIS ROUTINES}

procedure Tstemform.RadioButton4Click(Sender: TObject);
```

```
{Make a sura list, where each component of the list contains all the lines in a single sura,
and each line is a sequence of words (as opposed to each line being a string of characters,
as in the text list which this procedure operates on. This list is then used as the basis for
stemming.}
var
 tempword : Tword;
 i,j,k,l : longint;
 currentline : Tline;
 wordcounter : longint;
begin
{For each of the sura texts}
for i := 0 to (textlist.length - 1) do
 begin
  stemform.memo1.lines.add (textlist.list [i].name);
  {For each line in the current text}
  for j := 0 to (textlist.list [i].length - 1) do
   begin
    {Get the current line; using a separate variable saves a lot of typing in what follows}
    currentline := textlist.list[i].text[j];
    k := 0;
    {Get rid of line number}
    while currentline.line [k] in ['0'..'9', '.', ' '] do
     k := k + 1;
    {Work through the current line looking for words}
    {Count the words in the current line}
     wordcounter := 0;
    while k < currentline.length do
     begin
      {Dump any leading spaces; there will be at least 1 leading space in front of each word
except
       maybe the first in the line, because spaces are what separates words.}
      while currentline.line [k] = ' ' do
       k := k + 1;
      {If not end of current line}
      if k < currentline.length then
       begin
        {Make a temporary word blank}
        for l := 0 to (maxwordlength - 1) do
         tempword.wrd [l] := ' ';
        tempword.length := 0;
        {While the current position (ie, the k index) contains a non-blank character,
        insert it into the temporary word. The index variable l is for the temporary word,
        ie, it's different from the current line index k}
       l := 0;
       while currentline.line [k] <> ' ' do
        begin
         tempword.wrd [l] := currentline.line [k];
         tempword.length := tempword.length + 1;
         {Move forward both in the current line and the temporary word for each character}
         k := k + 1;
```

```
        l := l + 1;
       end;
      tempword.suffixstemmed := false;
      {We now have a word, which needs to be inserted into the current sura}
      {stemform.memo1.lines.add (textlist.list [i].name + ' ' + inttostr (i) + ' Line ' + inttostr
(j) + ' Word ' + tempword.wrd + inttostr (wordcounter)); }
      suralist.list [i].linelist [j].words [wordcounter] := tempword;
      {Increment the word counter}
      wordcounter := wordcounter + 1;
     end;
      suralist.list [i].linelist [j].linelength := wordcounter;
     end;
   end;
   {Insert the identifying file name and the length of the list for the current sura}
   suralist.list [i].suraname := textlist.list[i].name;
   suralist.list [i].suralength := textlist.list [i].length;
  end;
 {= the number of suras}
 suralist.length := textlist.length;
 {Output}
 {for i := 0 to (suralist.length - 1) do
  begin
   stemform.memo1.lines.add (' ');
   stemform.memo1.lines.add (suralist.list [i].suraname);
   for j := 0 to (suralist.list [i].suralength - 1) do
    for k := 0 to (suralist.list [i].linelist [j].linelength - 1) do
     stemform.memo1.Lines.add (suralist.list [i].linelist [j].words [k].wrd);
   end;}
  stemform.Memo1.lines.add ('sura list complete');
 end;

 procedure searchstoplist (var currentword : Tword;
                           var found : boolean);
 var
   i : longint;
   stopword : Tstopword;
 begin
  {Compare the current word supplied as a parameter to each successive word in the stop
 list. If
   the word is found in the stop list, set found true;}
  for i := 0 to ( stopwordlist.length -1 ) do
   begin
     stopword := stopwordlist.list [i].stopword;
     if currentword.wrd = stopword then
     found := true;
   end;
 end;

 procedure searchkalist (var currentword : Tword;
                         var found : boolean);
```

```
var
 i : longint;
 kaword : Tkaword;
begin
 {Compare the current word supplied as a parameter to each successive word in the ka list.
If
 the word is found in the ka list, set found true;}
 for i := 0 to ( kawordlist.length -1 ) do
 begin
  kaword := kawordlist.list [i].kaword;
  if currentword.wrd = kaword then
  found := true;
 end;
end;

procedure searchsalist (var currentword : Tword;
                        var found : boolean);
var
 i : longint;
 saword : Tsaword;
begin
 {Compare the current word supplied as a parameter to each successive word in the sa list.
If
 the word is found in the sa list, set found true;}
 for i := 0 to ( sawordlist.length -1 ) do
 begin
  saword := sawordlist.list [i].saword;
  if currentword.wrd = saword then
  found := true;
 end;
end;

                        {STEMMING ROUTINES}

                          {Prefixes}

procedure stemwa (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'w') and (currentword.wrd [1] = 'a') then
 begin
  for i := 0 to (maxwordlength - 1) do
   tempword.wrd [i] := ' ';
  tempword.length := 0;
  i := 2;
  j := 0;
  while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
```

```
   begin
    tempword.wrd [j] := currentword.wrd [i];
    i := i + 1;
    j := j + 1;
    end;
   tempword.length := currentword.length - 2;
   currentword := tempword;
  end;
end;

procedure stemfa (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'f') and (currentword.wrd [1] = 'a') then
  begin
   for i := 0 to (maxwordlength - 1) do
    tempword.wrd [i] := ' ';
   tempword.length := 0;
   i := 2;
   j := 0;
   while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
    begin
     tempword.wrd [j] := currentword.wrd [i];
     i := i + 1;
     j := j + 1;
    end;
   tempword.length := currentword.length - 2;
   currentword := tempword;
  end;
end;

procedure stemla (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'l') and (currentword.wrd [1] = 'a') then
  begin
for i := 0 to (maxwordlength - 1) do
 tempword.wrd [i] := ' ';
tempword.length := 0;
i := 2;
j := 0;
while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
 begin
  tempword.wrd [j] := currentword.wrd [i];
  i := i + 1;
  j := j + 1;
```

```
   end;
  tempword.length := currentword.length - 2;
  currentword := tempword;
 end;
end;

procedure stemli (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'l') and (currentword.wrd [1] = 'i') then
 begin
  for i := 0 to (maxwordlength - 1) do
   tempword.wrd [i] := ' ';
  tempword.length := 0;
  i := 2;
  j := 0;
  while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
   begin
    tempword.wrd [j] := currentword.wrd [i];
    i := i + 1;
    j := j + 1;
   end;
  tempword.length := currentword.length - 2;
   currentword := tempword;
 end;
end;

procedure stemlil (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'l') and (currentword.wrd [1] = 'i') and (currentword.wrd [2] = 'l')
then
 begin
  for i := 0 to (maxwordlength - 1) do
   tempword.wrd [i] := ' ';
  tempword.length := 0;
  i := 3;
j := 0;
while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
 begin
  tempword.wrd [j] := currentword.wrd [i];
  i := i + 1;
  j := j + 1;
 end;
tempword.length := currentword.length - 3;
currentword := tempword;
```

```
 end;
end;

procedure stembi (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'b') and (currentword.wrd [1] = 'i') then
  begin
 for i := 0 to (maxwordlength - 1) do
  tempword.wrd [i] := ' ';
 tempword.length := 0;
 i := 2;
 j := 0;
 while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
  begin
   tempword.wrd [j] := currentword.wrd [i];
   i := i + 1;
   j := j + 1;
  end;
 tempword.length := currentword.length - 2;
 currentword := tempword;
end;
end;

procedure stemka (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'k') and (currentword.wrd [1] = 'a') then
  begin
   for i := 0 to (maxwordlength - 1) do
    tempword.wrd [i] := ' ';
   tempword.length := 0;
   i := 2;
   j := 0;
   while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
   begin
    tempword.wrd [j] := currentword.wrd [i];
    i := i + 1;
    j := j + 1;
   end;
  tempword.length := currentword.length - 2;
  currentword := tempword;
 end;
end;

procedure stemsa (var currentword : Tword);
```

```
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 's') and (currentword.wrd [1] = 'a') then
  begin
   for i := 0 to (maxwordlength - 1) do
    tempword.wrd [i] := ' ';
   tempword.length := 0;
   i := 2;
   j := 0;
   while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
    begin
     tempword.wrd [j] := currentword.wrd [i];
     i := i + 1;
     j := j + 1;
    end;
   tempword.length := currentword.length - 2;
   currentword := tempword;
  end;
end;

procedure stemal (var currentword : Tword);
var
 tempword : Tword;
 i,j : longint;
begin
 if (currentword.wrd [0] = 'a') and (currentword.wrd [1] = 'l') then
  begin
   for i := 0 to (maxwordlength - 1) do
    tempword.wrd [i] := ' ';
   tempword.length := 0;
   i := 2;
   j := 0;
   while (currentword.wrd [i] <> ' ') and (i <= maxwordlength) do
    begin
     tempword.wrd [j] := currentword.wrd [i];
     i := i + 1;
     j := j + 1;
    end;
   tempword.length := currentword.length - 2;
   currentword := tempword;
  end;
end;

procedure Tstemform.RadioButton5Click(Sender: TObject);
var
 currentword : Tword;
 i,j,k : longint;
begin
```

```
for i := 0 to (suralist.length - 1) do
 begin
  sura := suralist.list [i];
  {stemform.Memo1.lines.add (wordlist.name);}
  for j := 0 to (sura.suralength - 1) do
   for k := 0 to (sura.linelist [j].linelength - 1) do
    begin
     currentword := sura.linelist [j].words [k];
     found := false;
     searchstoplist (currentword, found);
     if not found then
      begin
       stemwa (currentword);
       sura.linelist [j].words [k] := currentword;
      end;
    end;
  suralist.list [i] := sura;
 end;

for i := 0 to (suralist.length - 1) do
 begin
  sura := suralist.list [i];
  {stemform.Memo1.lines.add (wordlist.name);}
  for j := 0 to (sura.suralength - 1) do
   for k := 0 to (sura.linelist [j].linelength - 1) do
    begin
     currentword := sura.linelist [j].words [k];
     found := false;
     searchstoplist (currentword, found);
     if not found then
      begin
       stemfa (currentword);
       sura.linelist [j].words [k] := currentword;
      end;
    end;
  suralist.list [i] := sura;
 end;

for i := 0 to (suralist.length - 1) do
 begin
  sura := suralist.list [i];
  {stemform.Memo1.lines.add (wordlist.name);}
  for j := 0 to (sura.suralength - 1) do
   for k := 0 to (sura.linelist [j].linelength - 1) do
    begin
     currentword := sura.linelist [j].words [k];
     found := false;
     searchstoplist (currentword, found);
     if not found then
      begin
```

```
      stemla (currentword);
      sura.linelist [j].words [k] := currentword;
     end;
   end;
 suralist.list [i] := sura;
end;

for i := 0 to (suralist.length - 1) do
 begin
 sura := suralist.list [i];
{stemform.Memo1.lines.add (wordlist.name);}
for j := 0 to (sura.suralength - 1) do
 for k := 0 to (sura.linelist [j].linelength - 1) do
  begin
   currentword := sura.linelist [j].words [k];
   found := false;
   searchstoplist (currentword, found);
   if not found then
    begin
     stemlil (currentword);
     sura.linelist [j].words [k] := currentword;
    end;
  end;
 suralist.list [i] := sura;
end;

for i := 0 to (suralist.length - 1) do
 begin
 sura := suralist.list [i];
 {stemform.Memo1.lines.add (wordlist.name);}
 for j := 0 to (sura.suralength - 1) do
  for k := 0 to (sura.linelist [j].linelength - 1) do
   begin
    currentword := sura.linelist [j].words [k];
    found := false;
    searchstoplist (currentword, found);
    if not found then
     begin
      stemli (currentword);
      sura.linelist [j].words [k] := currentword;
     end;
   end;
 suralist.list [i] := sura;
end;

for i := 0 to (suralist.length - 1) do
 begin
 sura := suralist.list [i];
 {stemform.Memo1.lines.add (wordlist.name);}
 for j := 0 to (sura.suralength - 1) do
```

```
for k := 0 to (sura.linelist [j].linelength - 1) do
 begin
  currentword := sura.linelist [j].words [k];
  found := false;
  searchstoplist (currentword, found);
  if not found then
   begin
    stembi (currentword);
    sura.linelist [j].words [k] := currentword;
   end;
end;
 suralist.list [i] := sura;
end;

for i := 0 to (suralist.length - 1) do
 begin
  sura := suralist.list [i];
  {stemform.Memo1.lines.add (wordlist.name);}
  for j := 0 to (sura.suralength - 1) do
   for k := 0 to (sura.linelist [j].linelength - 1) do
    begin
     currentword := sura.linelist [j].words [k];
     found := false;
     searchstoplist (currentword, found);
     if not found then
      begin
       stemka (currentword);
       sura.linelist [j].words [k] := currentword;
      end;
    end;
  suralist.list [i] := sura;
 end;

for i := 0 to (suralist.length - 1) do
 begin
  sura := suralist.list [i];
  {stemform.Memo1.lines.add (wordlist.name);}
  for j := 0 to (sura.suralength - 1) do
   for k := 0 to (sura.linelist [j].linelength - 1) do
    begin
     currentword := sura.linelist [j].words [k];
     found := false;
     searchstoplist (currentword, found);
     if not found then
      begin
       stemsa (currentword);
       sura.linelist [j].words [k] := currentword;
      end;
    end;
  suralist.list [i] := sura;
```

```
 end;

for i := 0 to (suralist.length - 1) do
 begin
  sura := suralist.list [i];
  {stemform.Memo1.lines.add (wordlist.name);}
  for j := 0 to (sura.suralength - 1) do
   for k := 0 to (sura.linelist [j].linelength - 1) do
    begin
     currentword := sura.linelist [j].words [k];
     found := false;
     searchstoplist (currentword, found);
     if not found then
      begin
       stemal (currentword);
       sura.linelist [j].words [k] := currentword;
      end;
    end;
   suralist.list [i] := sura;
  end;

 stemform.memo1.Lines.add ('Prefix stemming complete');
end;
```

```
                    {Suffixes}

function containssuffix (word : Tword;
                         suffix : Tsuffix) : boolean;
{See is the word supplied as a parameter contains the supplied suffix}
var
 found : boolean;
 wordindex : longint;
 i : longint;
begin
 found := true;
 {If the word is shorter than the suffix, there's no point even looking}
 if word.length < suffix.length then
   found := false
 else
 begin
  {Match up suffix and word from the right, character by character. If there's any
   failure to match, the word doesn't contain this suffix}
  found := true;
  wordindex := word.length - 1;
  for i := (suffix.length - 1) downto 0 do
   begin
    {stemform.Memo1.lines.add (word.wrd [wordindex] + ' ' + suffix.str [i]);}
    if suffix.str [i] <> word.wrd [wordindex] then
     found := false;
```

```
    wordindex := wordindex - 1;
   end;
  end;
 containssuffix := found;
end;

procedure removesuffix (var word : Tword;
                        suffix : Tsuffix);
var
 wordindex : longint;
 i : longint;
begin
 {Replace the characters in the word corresponding to the suffix with blanks; work from the
right, like Arabs}
 wordindex := word.length - 1;
 for i := (suffix.length - 1) downto 0 do
  begin
   word.wrd [wordindex] := ' ';
   word.length := word.length - 1;
   wordindex := wordindex - 1;
  end;
end;

procedure Tstemform.RadioButton16Click(Sender: TObject);
{This procedure does all the suffix stemming. It goes through the suffix list and,
 for each suffix, considers each word in the sura list for removal of that
 suffix --if the suffix is present in a given word, it is removed, and the stemmed
 word is returned to the sura list. IMPORTANT: ths procedure assumes that
 the list of suffixes which is imported is in ASCENDING ORDER OF LENGTH}
var
 currentsuffix : Tsuffix;
 currentword : Tword;
 i,j,k,l : longint;
begin
 {For each suffix in the imported suffix list}
 for i := (suffixlist.length - 1) downto 0 do
  begin
   {Get the current suffix}
   currentsuffix := suffixlist.list [i];
   {For each word in the list of word lists}
  for j := 0 to (suralist.length - 1) do
   begin
    for k := 0 to (suralist.list [j].suralength - 1) do
     for l := 0 to (suralist.list [j].linelist [k].linelength - 1) do
      begin
       {Get the current word}
       currentword := suralist.list [j].linelist [k].words [l];
       {If the current word contains the current suffix}
       if containssuffix (currentword, currentsuffix) then
        begin
```

```
        removesuffix (currentword, currentsuffix);
        currentword.suffixstemmed := true;
        suralist.list [j].linelist [k].words [l] := currentword;
       end;
      end;
    end;
  end;
 stemform.memo1.Lines.add ('Suffix stemming complete');
end;
```

{OUTPUT}

```
procedure writeword (word : Tword);
var
 i : longint;
begin
 i := 0;
 while word.wrd [i] <> ' ' do
  begin
   write (outfile, word.wrd [i]);
   i := i + 1;
  end;
 write (outfile, ' ');
end;
```

```
procedure Tstemform.RadioButton7Click(Sender: TObject);
{Save the list of word lists to an output file}
var
 i,j,k : longint;
begin
 stemform.savedialog1.execute;
 assignfile (outfile, stemform.savedialog1.filename);
 rewrite (outfile);
  for i := 0 to (suralist.length - 1) do
   begin
    j := 0;
    while filenamelist.list [i,j] <> '.' do
     begin
      write (outfile, filenamelist.list [i,j]);
     j := j + 1;
     end;
    writeln (outfile);
    for j := 0 to (suralist.list[i].suralength - 1) do
     begin
      for k := 0 to (suralist.list [i].linelist [j].linelength - 1) do
       writeword (suralist.list [i].linelist [j].words [k]);
      writeln (outfile);
     end;
   end;
closefile (outfile);
```

```
 stemform.memo1.lines.add ('List saved');
end;

procedure Tstemform.RadioButton8Click(Sender: TObject);
var
 str1 : Tstr;
 str2 : Tstr;
 str3 : string;
 i,j,k : longint;
begin
 assignfile (stemfilenamefile, 'stemfilenames.txt');
rewrite (stemfilenamefile);
for i := 0 to (suralist.length - 1) do
 begin
  str1 := 'stem';
  str2 := filenamelist.list [i];
  str3 := str1 + str2;
  stemform.memo1.lines.add (str3);
  writeln (stemfilenamefile, str3);
  assignfile (outfile, str3);
  rewrite (outfile);
  j := 0;
  while filenamelist.list [i,j] <> '.' do
   begin
    write (outfile, filenamelist.list [i,j]);
    j := j + 1;
   end;
  writeln (outfile);
  for j := 0 to (suralist.list[i].suralength - 1) do
   begin
    for k := 0 to (suralist.list [i].linelist [j].linelength - 1) do
     writeword (suralist.list [i].linelist [j].words [k]);
    writeln (outfile);
   end;
  closefile (outfile);
 end;
closefile (stemfilenamefile);
stemform.Memo1.lines.add ('Files saved');
end;

end.
```

1.2 **CreateFreqMatrix**

a) Purpose

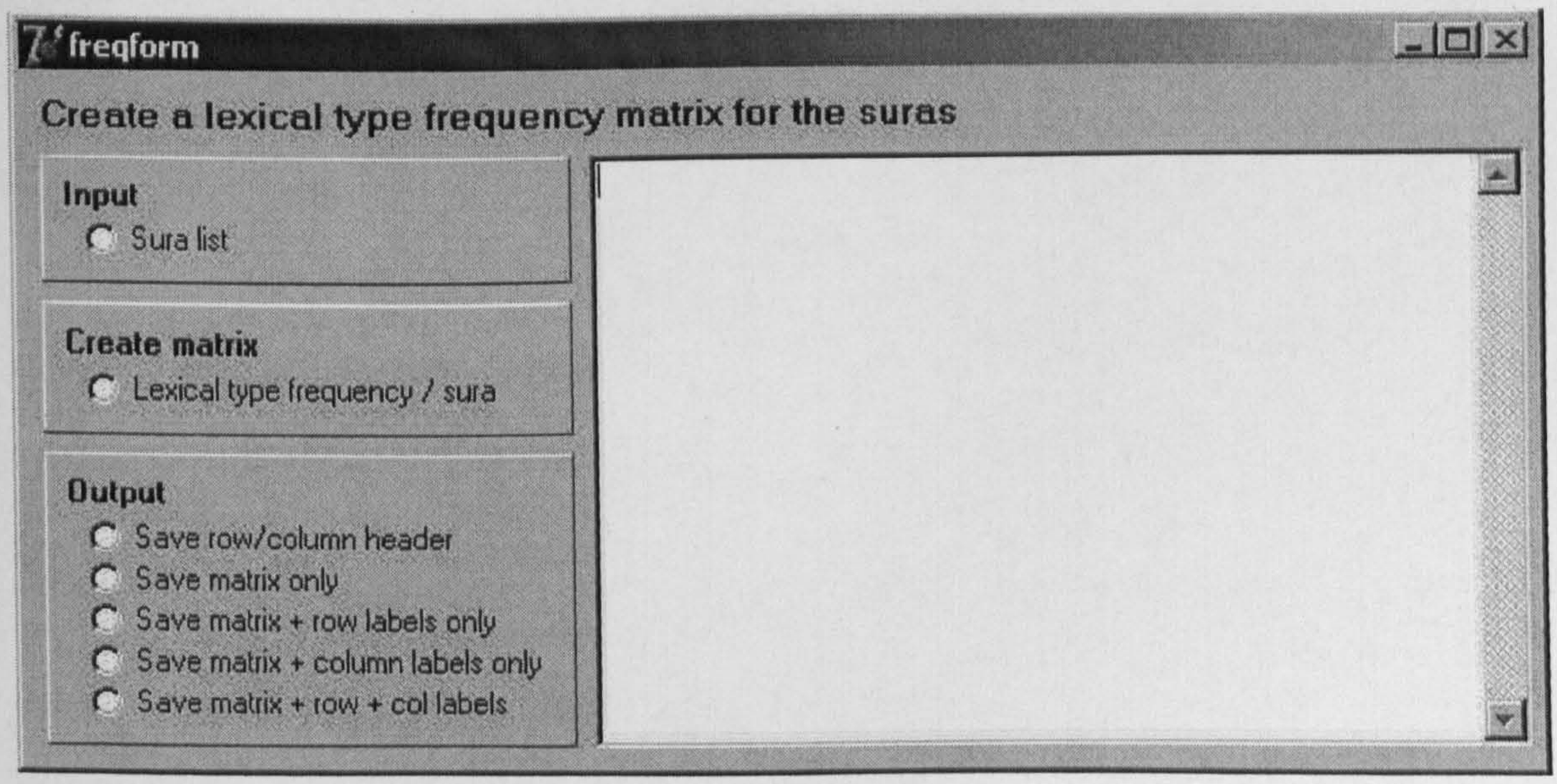
Given a list of sura names and a corresponding set of suras such that, for i = 1..114, $name_i$ denotes $sura_i$, construct a vector for each $sura_i$ such that:

- Each element of the $sura_i$ vector represents a lexical type, with the result that the vector has as many elements n as there are lexical types in the Qur'an.
- The value in each element of $vector_j$, for $j = 1..n$, is the number of times the associated lexical type occurs in $sura_i$.

For convenience, the set of 114 sura vectors is represented as a matrix $M_{114,n}$ in which the rows represent suras and the columns represent word types.

b) Use

The user interface looks like this:



Radio buttons are intended to be used in strict sequence from top to bottom:

1. In the 'Input' box, 'Sura list' prompts for the name of a file containing a list of 114 sura filenames.

2. In the 'Create matrix' box, 'Lexical type frequency / sura' constructs the matrix M

3. In the 'Output' box, the sequence of radio buttons allows M to be formatted in various ways:

- 'Save row/column header' saves the number of rows and columns in the matrix in the first two lines of the output file.
- 'Save matrix only' saves the matrix without row or column labels.
- The remaing buttons save row and/or column labels with the matrix in ways indicated by the button captions.

4. The text area on the right of the interface shows messages to the user during program execution and also allows display of various kinds of interim output during program debugging.

c) Algorithm

1. Define the following constants:

- *nrofsuras*: the number of suras in the Qur'an, that is, 114.
- *maxnroflexicaltypes*: user-predefined maximum number of lexical types across all 114 suras in the Qur'an

2. Define the following variables:

- *filenamelist*: a list of *nrofsuras* file names such that each name denotes a different text file containing one sura.

- *lextypelist*: a list of all the lexical types in the Qur'an.
- *lextypelistlength*: the length of *lextypelist*.
- *frequencymatrix*: a matrix with *nrofsuras* rows and *maxnroflexicaltypes* columns such that, for any $i = 1..nrofsuras$, $j = 1..lextypelistlength <= maxnroflexicaltypes$, the value at $frequencymatrix_{ij}$ is the frequency of lexical type $lextypelist_j$ in $sura_i$.

3. Read *filenamelist* from a user-specified file.

4. Initialize *lextypelistlength* to 0.

5. Initialize all values of $frequencymatrix_{ij}$ to 0, for $i = 1..nrofsuras$, $j = 1..maxnroflexicaltypes$.

6. For $i = 1$ to *nrofsuras*

begin

i. Input the text of $sura_i$ corresponding to $filenamelist_i$.

ii. For $j = 1$ to the number of lexical tokens in $sura_i$

begin

- Read $token_j$ from the text of $sura_i$.
- For $k = 1$ to *lextypelistlength*

If $token_j = lextypelist_j$, then increment $frequencymatrix_{ik}$ by 1.

- If $token_j$ was not found in the immediately preceding step, then increment *lextypelistlength* by 1, insert $token_j$ into

lextypelist$_{lextypelistlength}$, and increment *frequencymatrix*$_{i,lextypelistlength}$ by 1.

end

end

7. Output *frequencymatrix* to a user-specified file.

d) Program listing

```
unit CreateFreqMatrix;

{Given a list of sura names and a corresponding set of suras formatted by program
'formattext' and stemmed by program
 'stem' such that, for i = 1..114, name[i] denotes sura[i], construct a vector for each sura[i]
such that:
 * Each element of the sura[i] vector represents a lexical type, with the result that the vector
has as many elements n as there
  are lexical types in the Qur'an.
 * The value in each element of vector[j], for j = 1..n, is the number of times the associated
lexical type occurs in sura[i].
For convenience, the set of 114 sura vectors is represented as a matrix M[114,n] in which
the rows represent suras and
the columns represent word types.}

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

const
 maxfilenamelength = 30;          {max length of file names in file name list}
 maxfilenamelistlength = 114;     {max nr of input files}
 maxlexicaltypelength = 30;       {max length of lexical types}
 maxnroflexicaltypes = 10000;     {max nr of lexical types in Qur'an}
 nrofsuras = 114;                 {must be same as maxfilenamelistlength, ie, one vector
per file}

type
 {Input file name list}
 Tfilename = packed array [0..(maxfilenamelength - 1)] of char;
 Tfilenamelist = record
                 list : array [0..(maxfilenamelistlength - 1)] of Tfilename;
```

```
        length : longint;
      end;

{Selected token list}
Tlexicaltype = packed array [0..(maxlexicaltypelength - 1)] of char;
Tlexicaltypelist = record
            list : array [0..(maxnroflexicaltypes - 1)] of Tlexicaltype;
            length : longint;
           end;

{Frequency matrix}
Tfreqmatrix = record
            m : array [0..(nrofsuras - 1), 0..(maxnroflexicaltypes - 1)] of longint;
            nrofrows : longint;
            nrofcols : longint;
           end;

Tfreqform = class(TForm)
   Label1: TLabel;
   Panel2: TPanel;
   Panel3: TPanel;
   Panel4: TPanel;
   Memo1: TMemo;
   OpenDialog1: TOpenDialog;
   SaveDialog1: TSaveDialog;
   Label2: TLabel;
   RadioButton2: TRadioButton;
   Label3: TLabel;
   RadioButton4: TRadioButton;
   Label4: TLabel;
   RadioButton5: TRadioButton;
   RadioButton1: TRadioButton;
   RadioButton6: TRadioButton;
   RadioButton7: TRadioButton;
   RadioButton8: TRadioButton;
   procedure FormCreate(Sender: TObject);
   procedure RadioButton2Click(Sender: TObject);
   procedure RadioButton4Click(Sender: TObject);
   procedure RadioButton5Click(Sender: TObject);
   procedure RadioButton6Click(Sender: TObject);
    procedure RadioButton8Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure RadioButton7Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
```

```
freqform: Tfreqform;
filenamelist : Tfilenamelist;
lexicaltypelist : Tlexicaltypelist;
freqmatrix : Tfreqmatrix;
lexicaltokencounter : longint;
saveheader : boolean;
infile : textfile;
inbuffer: array[1..4096] of char;
outfile : textfile;
outbuffer: array[1..4096] of char;

implementation
{$R *.dfm}

procedure Tfreqform.FormCreate(Sender: TObject);
begin
 saveheader := false;
end;

procedure Tfreqform.RadioButton2Click(Sender: TObject);
{Read file name list from external file}
var
 ch : char;
 i, j : longint;
begin
freqform.opendialog1.execute;
assignfile (infile, freqform.opendialog1.filename);
reset (infile);
system.settextbuf (infile, inbuffer);
i := 0;
while not eof (infile) do
 begin
  j := 0;
  while (not eoln (infile)) and (not eof (infile)) do
   begin
    read (infile, ch);
    {convert to lower case if necessary}
    if ch in ['A'..'Z'] then
     ch:= chr(ord(ch) + 32);
    filenamelist.list [i,j] := ch;
    j := j + 1;
   end;
  if not eof (infile) then
   readln (infile);
  i := i + 1;
 end;
filenamelist.length := i;
freqform.memo1.lines.add ('File name list read');
end;
```

```
procedure Tfreqform.RadioButton4Click(Sender: TObject);
{Construct frequency matrix}
var
 candidatetype : Tlexicaltype;
 ch : char;
 found : boolean;
 foundindex : longint;
 i,j : longint;
begin
 lexicaltokencounter := 0;
freqmatrix.nrofrows := nrofsuras;
freqmatrix.nrofcols := 0;
lexicaltypelist.length := 0;
for i := 0 to (nrofsuras - 1) do
 for j := 0 to (maxnroflexicaltypes - 1) do
  freqmatrix.m [i,j] := 0;
{Process each input file in succession}
for i := 0 to (filenamelist.length - 1) do
 begin
  freqform.memo1.lines.add (filenamelist.list [i]);
  assignfile (infile, filenamelist.list [i]);
  reset (infile);
  system.settextbuf (infile, inbuffer);
  {Skip the sura title}
  readln (infile);
  {Read each token in the current file}
  ch := '*';
  while not eof (infile) do
   begin
    {Initialize the template for the token about to be read}
    for j := 0 to (maxlexicaltypelength - 1) do
     candidatetype [j] := ' ';
    {Eliminate any leading junk}
    ch := '*';
    while (not (ch in ['A'..'Z', 'a'..'z', '/', '-', '&'])) and (not eoln (infile)) and (not eof (infile))
do
     read (infile, ch);
    {If not eof at this stage}
    if not eof (infile) then
     begin
{If at end of line, go to next line}
if eoln (infile) then
 readln (infile)
else
{Read the token}
 begin
  {The first character has already been read above; insert it into the candidate type}
  candidatetype [0] := ch;
  {Read the rest of the characters}
  j := 1;
```

```
        while (ch <> ' ') and (not eoln (infile)) and (not eof (infile)) do
         begin
          read (infile, ch);
          candidatetype [j] := ch;
          j := j + 1;
         end;
        lexicaltokencounter := lexicaltokencounter + 1;
        {See if the token just read is in the list of selected tokens, and note the token list
index}
        if lexicaltypelist.length = 0 then
         begin
          lexicaltypelist.list [0] := candidatetype;
          lexicaltypelist.length := 1;
          freqmatrix.m [i,0] := freqmatrix.m [i,0] + 1;
          freqmatrix.nrofcols := freqmatrix.nrofcols + 1;
         end
        else
         begin
          found := false;
          for j := 0 to (lexicaltypelist.length - 1) do
           begin
            if candidatetype = lexicaltypelist.list [j] then
             begin
              found := true;
              foundindex := j;
             end;
           end;
          if found then
           freqmatrix.m [i,foundindex] := freqmatrix.m [i,foundindex] + 1
          else
           begin
            lexicaltypelist.list [lexicaltypelist.length] := candidatetype;
            lexicaltypelist.length := lexicaltypelist.length + 1;
            freqmatrix.m [i,lexicaltypelist.length] := freqmatrix.m i,lexicaltypelist.length] + 1;
            freqmatrix.nrofcols := freqmatrix.nrofcols + 1;
           end;
         end;
       end;
      end;
    end;
   closefile (infile);
   end;
  freqform.memo1.lines.add ('Total lexical tokens found in stemmed suras: ' + inttostr
(lexicaltokencounter));
  freqform.memo1.lines.add ('Total lexical types found in stemmed suras: ' + inttostr
(lexicaltypelist.length));
  freqform.memo1.lines.add ('Processing complete');
end;

procedure Tfreqform.RadioButton5Click(Sender: TObject);
```

```
{Save matrix}
var
 i,j : longint;
begin
 freqform.savedialog1.execute;
 assignfile (outfile, freqform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 if saveheader then
  begin
  writeln (outfile, freqmatrix.nrofrows);
  writeln (outfile, freqmatrix.nrofcols);
 end;
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
 j := 0;
 while lexicaltypelist.list [i,j] <> ' ' do
  begin
   write (outfile, lexicaltypelist.list [i,j]);
   j := j + 1;
  end;
 write (outfile, ' ');
 end;
writeln (outfile);
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
 j := 1;
 while filenamelist.list [i,j] <> ' ' do
  begin
   write (outfile, filenamelist.list [i,j]);
   j := j + 1;
  end;
 write (outfile, ' ');
 for j := 0 to (freqmatrix.nrofcols - 1) do
  write (outfile, freqmatrix.m [i,j], ' ');
 writeln (outfile);
end;
closefile (outfile);
freqform.memo1.lines.add ('Done');
end;

procedure Tfreqform.RadioButton6Click(Sender: TObject);
{Save matrix}
var
 i,j : longint;
begin
 freqform.savedialog1.execute;
 assignfile (outfile, freqform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
```

```
if saveheader then
 begin
  writeln (outfile, freqmatrix.nrofrows);
  writeln (outfile, freqmatrix.nrofcols);
 end;
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j], ' ');
  writeln (outfile);
 end;
 closefile (outfile);
 freqform.memo1.lines.add ('Done');
end;

procedure TfreqformRadioButton7Click(Sender: TObject);
{Save matrix}
var
 i,j : longint;
begin
 freqform.savedialog1.execute;
 assignfile (outfile, freqform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 if saveheader then
begin
 writeln (outfile, freqmatrix.nrofrows);
 writeln (outfile, freqmatrix.nrofcols);
end;
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
 j := 1;
 while filenamelist.list [i,j] <> ' ' do
  begin
   write (outfile, filenamelist.list [i,j]);
   j := j + 1;
  end;
 write (outfile, ' ');
 for j := 0 to (freqmatrix.nrofcols - 1) do
  write (outfile, freqmatrix.m [i,j], ' ');
  writeln (outfile);
 end;
 closefile (outfile);
 freqform.memo1.lines.add ('Done');
end;

procedure Tfreqform.RadioButton8Click(Sender: TObject);
{Save matrix}
var
 i,j : longint;
```

```
begin
 freqform.savedialog1.execute;
 assignfile (outfile, freqform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 if saveheader then
  begin
   writeln (outfile, freqmatrix.nrofrows);
   writeln (outfile, freqmatrix.nrofcols);
  end;
for i := 0 to (lexicaltypelist.length - 1) do
 begin
  j := 0;
  while lexicaltypelist.list [i,j] <> ' ' do
   begin
    write (outfile, lexicaltypelist.list [i,j]);
    j := j + 1;
   end;
  write (outfile, ' ');
 end;
writeln (outfile);
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j], ' ');
   writeln (outfile);
 end;
 closefile (outfile);
 freqform.memo1.lines.add ('Done');
end;

procedure Tfreqform.RadioButton1Click(Sender: TObject);
{On-start initialization}
begin
 saveheader := true;
end;

procedure Tfreqform.RadioButton7Click(Sender: TObject);
{Save matrix}
var
 i,j : longint;
begin
 freqform.savedialog1.execute;
 assignfile (outfile, freqform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 if saveheader then
  begin
   writeln (outfile, freqmatrix.nrofrows);
   writeln (outfile, freqmatrix.nrofcols);
```

```
 end;
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
 j := 1;
 while filenamelist.list [i,j] <> ' ' do
  begin
   write (outfile, filenamelist.list [i,j]);
   j := j + 1;
  end;
  write (outfile, ' ');
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j], ' ');
  writeln (outfile);
 end;
 closefile (outfile);
 freqform.memo1.lines.add ('Done');
end;

end.
```

1.3 **EditMatrix**

a) Purpose

As its name indicates, program *EditMatrix* provides a range of facilities for editing a frequency matrix M created by program *CreateFreqMatrix* in ways that are relevant to the present discussion. Given M as input, *EditMatrix* provides:

i. Normalization for variation in sura length

ii. Several types of dimensionality reduction, including

- Removal of frequency-1 columns and columns associated with function words.
- Retention of explicitly-specified columns.
- Removal / retention of columns on the basis of column standard deviation.
- Removal / retention of columns on the basis of column term frequency / inverse document frequency.
- Removal / retention of columns on the basis of column Poisson distribution.
- Removal / retention of columns on the basis of column entropy.
- Creation of a reduced-dimensionality matrix by linear principal components analysis.

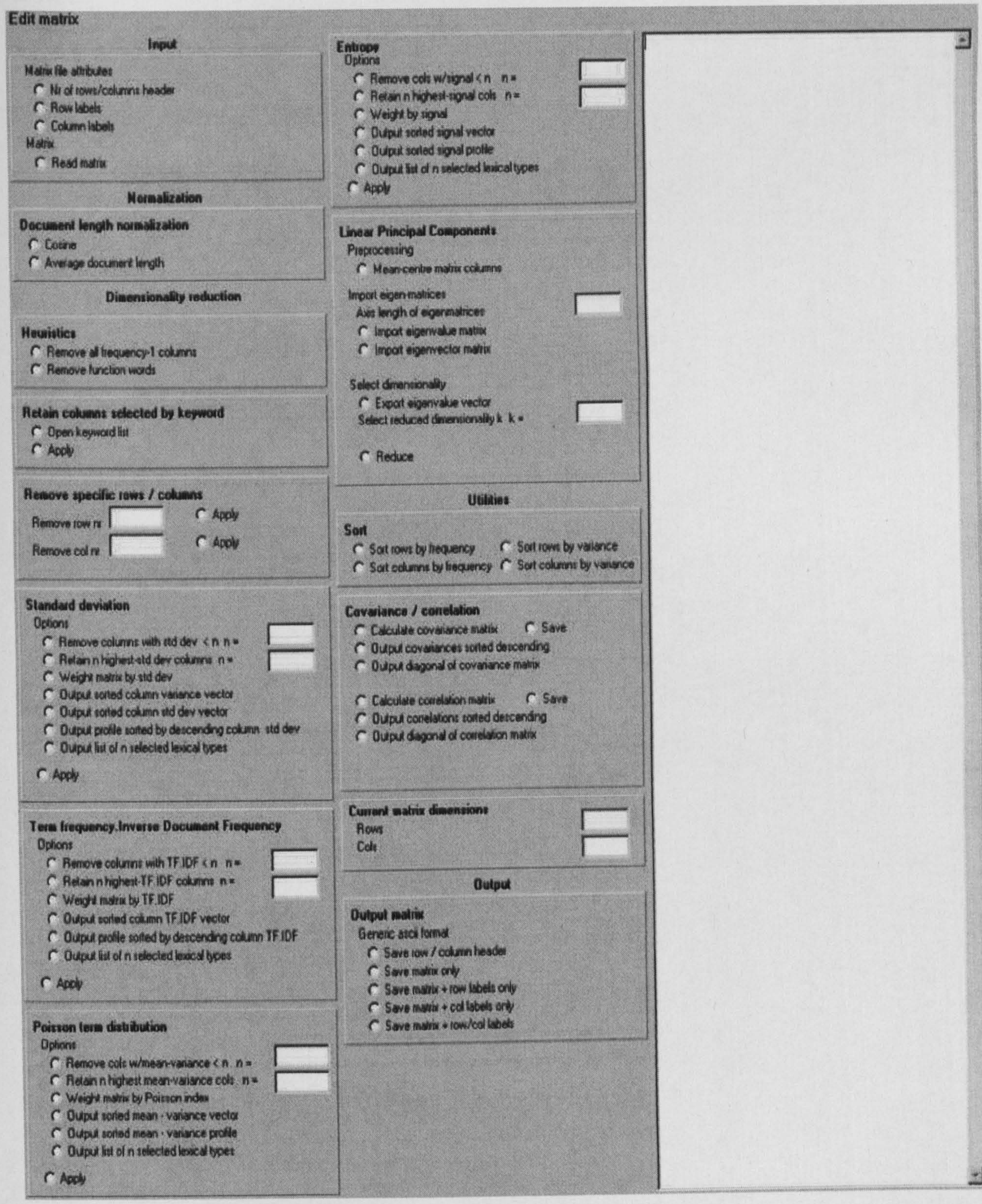
iii. Various utilities:

- Sorting of rows and columns by frequency and covariance.
- Calculation, sorting, and output of covariance and correlation matrices.

iv. Output of the edited matrix in a variety of formats.

b) Use

The interface for *EditMatrix* looks like this:



i. *Input*

Input to this program is a numerical-valued matrix. 'Matrix file attributes' allows specification of the input matrix file: whether there is a header giving the number of rows and columns, whether there are row labels, and whether there are column labels. 'Read matrix' loads the matrix from the user-specified file.

ii. *Normalization*

Normalizes the input matrix to compensate for variation in sura length either by cosine normalization or average document length.

iii. *Dimensionality reduction*

Offers a range of dimensionality reduction methods. Details of these various methods are given in chapter 5 of the foregoing discussion.

- 'Heuristics': Removes of frequency-1 columns and columns associated with function words. In the latter case, a file listing the function words must be selected.
- 'Retain columns selected by keyword': Retains columns associated with user-specified keywords. A file listing the keywords must be selected.
- 'Standard deviation': Removes / retains columns on the basis of column standard deviation, and weights column values by standard deviation. There are also facilities for generating (i) sorted variance and standard deviation vectors for all columns in the matrix, (ii) a profile listing columns labels, frequencies, variances, and standard deviations simultaneously for all columns, and (iii) a list of column labels sorted by standard deviation. Required options are selected, and 'Apply' carries them out.

- 'Term frequency.Inverse Document Frequency': Removes / retains columns on the basis of TF.IDF, and weights column values by TF.IDF. There are also facilities for generating (i) a sorted TF.IDF vector for all matrix columns, (ii) a TF.IDF-sorted profile listing column labels and corresponding TF.IDF values, and (iii) a list of column labels sorted by TF.IDF. Required options are selected, and 'Apply' carries them out
- 'Poisson term distribution': Removes / retains columns on the basis of column Poisson distribution, and weights column values by Poisson value. There are also facilities for generating (i) a sorted Poisson distribution vector for all matrix columns, (ii) a Poisson distribution-sorted profile listing column labels and corresponding Poisson values, and (iii) a list of column labels sorted by Poisson value. Required options are selected, and 'Apply' carries them out
- 'Entropy': Removes / retains columns on the basis of column Poisson entropy, and weights column values by entropy value. There are also facilities for generating (i) a sorted entropy vector for all matrix columns, (ii) an entropy-sorted profile listing column labels and corresponding entropy values, and (iii) a list of column labels sorted by entropy value. Required options are selected, and 'Apply' carries them out
- 'Linear principal components': Creates a reduced-dimensionality matrix by linear principal components analysis. The operations provided in this section do not include calculation of the eigenvalue and eigenvector matrices. Instead, they interact with external calculation of the covariance and eigenvalue / eigenvector matrices:

- 'Mean centre matrix columns' mean-centres the original data matrix columns in preparation for calculation of the covariance matrix. The resulting mean-centred

matrix is then saved using 'Save matrix only' in the Output section of this program, described below.

- 'Import eigen-matrices' imports the externally-calculated eigenvalue and eigenvector matrices.
- 'Select dimensionality' selects the number of principal components to retain
- 'Project' projects the original data matrix into the reduced-dimensionality eigen-space.

The sequencing of these operations is described in greater detail in the preceding discussion of dimensionality of the Qur'an data matrix by variable redefinition.

iv. *Utilities*

This section of the interface provides a small range of utilities that were found useful in the course of data matrix manipulation:

1. 'Sort' sorts the rows and columns of the matrix by frequency and variance.
2. 'Covariance / correlation' calculates and outputs covariance and correlation matrices for the data matrix.
3. 'Current matrix dimensions' shows the effect of the various dimensionality reduction operations on the data matrix.

v. *Output*

Allows the transformed data matrix to be saved in a variety of self-explanatory formats; 'Save row / column header' writes the number of rows and columns in the transformed matrix to the first two lines of the output file.

Finally, the text box on the right of the interface shows messages to the user during program execution and also allows display of various kinds of interim output during program debugging.

c) Algorithms

i. *Input*

If the number of rows and columns is included in the file containing the matrix, use a deterministic loop to read values into the variable *freqmatrix* in row order, otherwise use a nondeterministic loop. Any column and row labels are read as part of the process.

ii. *Normalization*

Implements the functions described in section 5.1.6.1 of the dissertation, 'Document length variation' .

iii. *Dimensionality reduction*

Implements the functions described in section 5.1.6.2 of the dissertation, 'Sparsity minimization'.

iv. *Utilities*

- Sort: Creates a temporary matrix and populates it with rows or columns from the original frequency matrix in descending order of magnitude of row / column frequency or magnitude, as appropriate. The original matrix is then overwritten by the sorted temporary matrix.

- Covariance / correlation: Calculates and outputs covariance and correlation matrices from the original frequency matrix. Also outputs covariances / correlations in descending order as well as the diagonals of the covariance / correlation matrix.

d) Program listing

```
unit EditMatrix;

{Provides a range of facilities for editing a frequency matrix M created by program
CreateFrequencyMatrix. Given M as input, EditMatrix provides:
1. Normalization for variation in sura length
2. Several types of dimensionality reduction, including
 * Removal of frequency-1 columns and columns associated with function words.
 * Retention of explicitly-specified columns.
 * Removal / retention of columns on the basis of column standard deviation.
 * Removal / retention of columns on the basis of column term frequency / inverse
document frequency.
 * Removal / retention of columns on the basis of column Poisson distribution.
 * Removal / retention of columns on the basis of column entropy.
 * Creation of a reduced-dimensionality matrix by linear principal components analysis.
3. Various utilities:
 * Sorting of rows and columns by frequency and covariance.
 * Calculation, sorting, and output of covariance and correlation matrices.
4. Output of the edited matrix in a variety of formats.}

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, fmath, fspec;

const
 maxstrlength = 18;   {for row and column labels}
 maxnrofrows = 1000;
 maxnrofcols = 8000;

type
  {Main frequency matrix}
  Tstr = packed array [0..(maxstrlength - 1)] of char;
  Trowlabellist = array [0..(maxnrofrows - 1)] of Tstr;
  Tcollabellist = array [0..(maxnrofcols - 1)] of Tstr;
  Twordlist = record
              list : array [0..(maxnrofcols - 1)] of Tstr;
              length : longint;
             end;
  Tmatrix = record
            m : array [0..(maxnrofrows - 1), 0..(maxnrofcols - 1)] of single;
```

```
        nrofrows : longint;
        nrofcols : longint;
        rowlabellist : Trowlabellist;
        collabellist : Tcollabellist;
       end;

{Eigen and covariance matrices}
Teigenmatrix = record
            m : array [0..(maxnrofcols - 1), 0..(maxnrofcols - 1)] of single;
            nrofrows : longint;
            nrofcols : longint;
           end;
Tcovariancematrix = record
              m : array [0..(maxnrofcols - 1), 0..(maxnrofcols - 1)] of single;
              axislength : longint;
             end;
{Profile}
Tcontentnode = record
            variablename : Tstr;
            frequency : single;
            variance : single;
            stddev : single;
            idf : single;
            tfidf : single;
            poisson : single;
            entropy : single;
            signal : single;
           end;
Tprofile = record
          list : array [0..(maxnrofcols - 1)] of Tcontentnode;
          selected : array [0..(maxnrofcols - 1)] of boolean;
          length : longint;
         end;
Trealvector = record {general-purpose vector}
            v : array [0..(maxnrofcols - 1)] of single;
            selected : array [0..(maxnrofcols - 1)] of boolean;
            length : longint;
           end;
Tintegervector = record  {general purpose vector}
              v : array [0..(maxnrofcols - 1)] of longint;
              selected : array [0..(maxnrofcols - 1)] of boolean;
              length : longint;
             end;

{Tree}
 Tpointer = ^Tnode;
 Tnode = record
        value : single;
        left : Tpointer;
        right : Tpointer;
```

```
    end;

Tmatrixform = class(TForm)
    Label1: TLabel;
    Panel1: TPanel;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    Label3: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Panel4: TPanel;
    Label15: TLabel;
    RadioButton16: TRadioButton;
    RadioButton17: TRadioButton;
    RadioButton18: TRadioButton;
    RadioButton19: TRadioButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    Memo1: TMemo;
    RadioButton14: TRadioButton;
    Panel6: TPanel;
    RadioButton24: TRadioButton;
    RadioButton25: TRadioButton;
    Label24: TLabel;
    RadioButton26: TRadioButton;
    RadioButton27: TRadioButton;
    Panel8: TPanel;
    Label25: TLabel;
    RadioButton32: TRadioButton;
    Label37: TLabel;
    Label38: TLabel;
    Label39: TLabel;
    Panel3: TPanel;
    Label40: TLabel;
    RadioButton28: TRadioButton;
    RadioButton9: TRadioButton;
    Panel10: TPanel;
    Label41: TLabel;
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    Edit3: TEdit;
    RadioButton13: TRadioButton;
    RadioButton11: TRadioButton;
    Panel11: TPanel;
    Label7: TLabel;
    RadioButton10: TRadioButton;
    RadioButton21: TRadioButton;
    Edit2: TEdit;
```

```
Edit7: TEdit;
RadioButton23: TRadioButton;
RadioButton30: TRadioButton;
Panel12: TPanel;
Label22: TLabel;
Label10: TLabel;
Edit13: TEdit;
Label11: TLabel;
Edit14: TEdit;
Panel13: TPanel;
Label4: TLabel;
Panel14: TPanel;
Label19: TLabel;
RadioButton8: TRadioButton;
RadioButton37: TRadioButton;
RadioButton39: TRadioButton;
RadioButton45: TRadioButton;
RadioButton46: TRadioButton;
Edit16: TEdit;
Edit17: TEdit;
RadioButton47: TRadioButton;
RadioButton48: TRadioButton;
Edit18: TEdit;
Edit19: TEdit;
Panel16: TPanel;
Label27: TLabel;
RadioButton49: TRadioButton;
RadioButton50: TRadioButton;
Panel2: TPanel;
Label44: TLabel;
Label9: TLabel;
RadioButton58: TRadioButton;
RadioButton59: TRadioButton;
Edit27: TEdit;
RadioButton61: TRadioButton;
RadioButton63: TRadioButton;
Edit21: TEdit;
RadioButton56: TRadioButton;
RadioButton53: TRadioButton;
Label47: TLabel;
RadioButton57: TRadioButton;
Label12: TLabel;
RadioButton65: TRadioButton;
RadioButton66: TRadioButton;
Edit1: TEdit;
RadioButton68: TRadioButton;
Label8: TLabel;
RadioButton69: TRadioButton;
Label13: TLabel;
RadioButton70: TRadioButton;
```

```
RadioButton71: TRadioButton;
RadioButton72: TRadioButton;
RadioButton73: TRadioButton;
RadioButton74: TRadioButton;
RadioButton76: TRadioButton;
RadioButton77: TRadioButton;
RadioButton78: TRadioButton;
StaticText2: TStaticText;
StaticText3: TStaticText;
RadioButton64: TRadioButton;
Panel5: TPanel;
Label14: TLabel;
Label16: TLabel;
Edit8: TEdit;
Label17: TLabel;
Edit9: TEdit;
RadioButton15: TRadioButton;
RadioButton20: TRadioButton;
StaticText4: TStaticText;
StaticText5: TStaticText;
Panel7: TPanel;
Label18: TLabel;
RadioButton5: TRadioButton;
RadioButton12: TRadioButton;
RadioButton22: TRadioButton;
Label20: TLabel;
Label2: TLabel;
RadioButton29: TRadioButton;
RadioButton31: TRadioButton;
RadioButton33: TRadioButton;
RadioButton34: TRadioButton;
RadioButton35: TRadioButton;
RadioButton36: TRadioButton;
Label21: TLabel;
procedure FormCreate(Sender: TObject);
procedure RadioButton1Click(Sender: TObject);
procedure RadioButton2Click(Sender: TObject);
procedure RadioButton3Click(Sender: TObject);
procedure RadioButton4Click(Sender: TObject);
procedure RadioButton11Click(Sender: TObject);
procedure RadioButton13Click(Sender: TObject);
procedure RadioButton16Click(Sender: TObject);
procedure RadioButton14Click(Sender: TObject);
procedure RadioButton17Click(Sender: TObject);
procedure RadioButton18Click(Sender: TObject);
procedure RadioButton19Click(Sender: TObject);
procedure RadioButton26Click(Sender: TObject);
procedure RadioButton27Click(Sender: TObject);
procedure RadioButton24Click(Sender: TObject);
procedure RadioButton25Click(Sender: TObject);
```

```
procedure RadioButton32Click(Sender: TObject);
procedure RadioButton28Click(Sender: TObject);
procedure RadioButton6Click(Sender: TObject);
procedure RadioButton10Click(Sender: TObject);
procedure RadioButton9Click(Sender: TObject);
procedure RadioButton21Click(Sender: TObject);
procedure RadioButton23Click(Sender: TObject);
procedure RadioButton30Click(Sender: TObject);
procedure RadioButton38Click(Sender: TObject);
procedure RadioButton39Click(Sender: TObject);
procedure RadioButton37Click(Sender: TObject);
procedure RadioButton45Click(Sender: TObject);
procedure RadioButton46Click(Sender: TObject);
procedure RadioButton47Click(Sender: TObject);
procedure RadioButton48Click(Sender: TObject);
procedure RadioButton49Click(Sender: TObject);
procedure RadioButton50Click(Sender: TObject);
procedure RadioButton7Click(Sender: TObject);
procedure RadioButton63Click(Sender: TObject);
procedure RadioButton58Click(Sender: TObject);
procedure RadioButton59Click(Sender: TObject);
procedure RadioButton61Click(Sender: TObject);
procedure RadioButton56Click(Sender: TObject);
procedure RadioButton53Click(Sender: TObject);
procedure RadioButton57Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure RadioButton66Click(Sender: TObject);
procedure RadioButton65Click(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit7Change(Sender: TObject);
procedure RadioButton69Click(Sender: TObject);
procedure RadioButton68Click(Sender: TObject);
procedure Edit16Change(Sender: TObject);
procedure Edit17Change(Sender: TObject);
procedure RadioButton70Click(Sender: TObject);
procedure RadioButton72Click(Sender: TObject);
procedure Edit18Change(Sender: TObject);
procedure Edit19Change(Sender: TObject);
procedure RadioButton71Click(Sender: TObject);
procedure RadioButton73Click(Sender: TObject);
procedure RadioButton74Click(Sender: TObject);
procedure RadioButton76Click(Sender: TObject);
procedure RadioButton77Click(Sender: TObject);
procedure RadioButton78Click(Sender: TObject);
procedure RadioButton64Click(Sender: TObject);
procedure RadioButton15Click(Sender: TObject);
procedure RadioButton20Click(Sender: TObject);
procedure RadioButton5Click(Sender: TObject);
procedure RadioButton29Click(Sender: TObject);
```

```
  procedure RadioButton12Click(Sender: TObject);
  procedure RadioButton31Click(Sender: TObject);
  procedure RadioButton33Click(Sender: TObject);
  procedure RadioButton34Click(Sender: TObject);
  procedure RadioButton35Click(Sender: TObject);
  procedure RadioButton36Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
 end;

var
 matrixform: Tmatrixform;

{Matrices}
 freqmatrix : Tmatrix;
 tempfreqmatrix : Tmatrix;
 eigenvaluematrix : Teigenmatrix;
 eigenvectormatrix : Teigenmatrix;
 tempeigenmatrix : Teigenmatrix;
 pcareducedmatrix : Tmatrix;
 covariancematrix : Tcovariancematrix;

{Parameters}
 keywordlist : Twordlist;
 selectedcolumn : longint;
 columnvector : Tintegervector;
 poissondistributionvector : Trealvector;
 columnfrequency : longint;
 lambda : single;
 svdk : longint;
 pcak : longint;
 nreal : single;
 ninteger : longint;

{Profiles}
 profile : Tprofile;
 tempprofile : Tprofile;
 sortedprofile : Tprofile;

{Booleans for user selections}
 readheader : boolean;
 readrowlabels : boolean;
 readcollabels : boolean;
 saveheader : boolean;
 normalized : boolean;
 outputcolvariancevector : boolean;
 outputcolstddevvector : boolean;
 outputcolvarianceprofile : boolean;
```

```
outputtfidfvector : boolean;
outputtfidfprofile : boolean;
outputtermdiscvector : boolean;
outputtermdiscprofile : boolean;
outputpoissonvector : boolean;
outputpoissonprofile : boolean;
outputsignalvector : boolean;
outputsignalprofile : boolean;
outputselectedlexicaltypelist : boolean;
weightbystddev : boolean;
weightbytfidf : boolean;
weightbytermdisc : boolean;
weightbypoisson : boolean;
weightbysignal : boolean;
removecolsbyvariance1 : boolean;
removecolsbyvariance2 : boolean;
removecolsbytfidf1 : boolean;
removecolsbytfidf2 : boolean;
removecolsbytermdisc1 : boolean;
removecolsbytermdisc2 : boolean;
removecolsbypoisson1 : boolean;
removecolsbypoisson2 : boolean;
removecolsbysignal1 : boolean;
removecolsbysignal2 : boolean;

{Binary tree}
 rootnode : Tpointer;
 newnode : Tpointer;
 parentnode : Tpointer;

{File variables}
 infile : textfile;
 inbuffer: array[1..4096] of char;
 outfile : textfile;
 outbuffer: array[1..4096] of char;

implementation
{$R *.dfm}

{*** INITIALIZATION ***}

procedure Tmatrixform.FormCreate(Sender: TObject);
{Initialization on program start}
begin
 readheader := false;
 readrowlabels := false;
 readcollabels := false;
 saveheader := false;
 normalized := false;
 outputcolvariancevector := false;
```

```
outputcolstddevvector := false;
outputcolvarianceprofile := false;
outputtfidfvector := false;
outputtfidfprofile := false;
outputpoissonvector := false;
outputpoissonprofile := false;
outputsignalvector := false;
outputsignalprofile := false;
outputselectedlexicaltypelist := false;
weightbystddev := false;
weightbytfidf := false;
weightbypoisson := false;
weightbysignal := false;
removecolsbyvariance1 := false;
removecolsbyvariance2 := false;
removecolsbytfidf1 := false;
removecolsbytfidf2 := false;
removecolsbypoisson1 := false;
removecolsbypoisson2 := false;
removecolsbysignal1 := false;
removecolsbysignal2 := false;
end;
                        {*** UTILITIES ***}
{Create a unit vector}
function unitvector (vect : Trealvector;
                     length : longint) : Trealvector;
var
 sumofsquares : single;
 absolutevalue : single;
 tempvect : Trealvector;
 i : longint;
begin
 sumofsquares := 0;
 for i := 0 to (length - 1) do
  sumofsquares := sumofsquares + (vect.v[i] * vect.v[i]);
 absolutevalue := sqrt (sumofsquares);
 for i := 0 to (length - 1) do
  if absolutevalue <> 0 then
   tempvect.v [i] := (1/absolutevalue) * vect.v [i]
  else
    tempvect.v [i] := 0;
 unitvector := tempvect;
end;
function sortprofile : Tprofile;
{Sort a profile}
var
 largest : single;
 index : longint;
 i,j,k : longint;
begin
```

```
if outputcolvariancevector or outputcolvarianceprofile then
 begin
  k := 0;
  for i := 0 to (profile.length - 1) do
   begin
    largest := 0;
    for j := 0 to (profile.length - 1) do
     begin
      if (profile.list [j].stddev > largest) and (not profile.selected [j]) then
       begin
        largest := profile.list [j].stddev;
        index := j;
       end;
     end;
    tempprofile.list [k] := profile.list [index];
    k := k + 1;
    tempprofile.selected [index] := false;
    profile.selected [index] := true;
   end;
 end;
if outputtfidfvector or outputtfidfprofile then
 begin
  k := 0;
  for i := 0 to (profile.length - 1) do
   begin
    largest := 0;
    for j := 0 to (profile.length - 1) do
     begin
      if (profile.list [j].tfidf > largest) and (not profile.selected [j]) then
       begin
        largest := profile.list [j].tfidf;
        index := j;
       end;
     end;
    tempprofile.list [k] := profile.list [index];
    k := k + 1;
    tempprofile.selected [index] := false;
    profile.selected [index] := true;
   end;
 end;
if outputpoissonvector or outputpoissonprofile then
 begin
  k := 0;
  for i := 0 to (profile.length - 1) do
   begin
    largest := 0;
    for j := 0 to (profile.length - 1) do
     begin
      if (profile.list [j].poisson > largest) and (not profile.selected [j]) then
       begin
```

```
    largest := profile.list [j].poisson;
    index := j;
   end;
 end;
tempprofile.list [k] := profile.list [index];
k := k + 1;
tempprofile.selected [index] := false;
profile.selected [index] := true;
end;
end;
if outputsignalvector or outputsignalprofile then
begin
 k := 0;
 for i := 0 to (profile.length - 1) do
  begin
   largest := 0;
   for j := 0 to (profile.length - 1) do
    begin
     if (profile.list [j].signal > largest) and (not profile.selected [j]) then
      begin
       largest := abs (profile.list [j].signal);
       index := j;
      end;
    end;
   tempprofile.list [k] := profile.list [index];
    k := k + 1;
    tempprofile.selected [index] := false;
    profile.selected [index] := true;
   end;
 end;
for i := 0 to (profile.length - 1) do
 profile.selected [i] := false;
tempprofile.length := profile.length;
sortprofile := tempprofile;
end;

                        {*** BINARY TREE ***}

procedure makenode (var root : Tpointer;
                    value : single);
{Create tree node}
begin
 new (root);
 root^.value := value;
 root^.left := nil;
 root^.right := nil;
end;

procedure insertnode (var parentnode : Tpointer;
                      newnode : Tpointer);
```

```
{Insert node in tree}
begin
 if parentnode = nil then
  parentnode := newnode
 else
  begin
   if newnode^.value >= parentnode.value then
    insertnode (parentnode^.left, newnode)
   else
    insertnode (parentnode^.right, newnode);
  end;
end;
procedure inorder (root : Tpointer);
{Inorder traversal of tree}
var
 i,j : longint;
begin
 if root <> nil then
  begin
   inorder (root^.left);
   writeln (outfile, root^.value:12:6);
   inorder (root^.right);
  end;
end;
```

```
{*** INPUT ***}

procedure Tmatrixform.RadioButton1Click(Sender: TObject);
{Boolean for existence of header in matrix input file}
begin
 readheader := true;
end;

procedure Tmatrixform.RadioButton3Click(Sender: TObject);
{Boolean for existence of row labels in matrix input file}
begin
 readrowlabels := true;
 matrixform.radiobutton17.Enabled := true;
end;
```

```
procedure Tmatrixform.RadioButton2Click(Sender: TObject);
{Boolean for existence of column labels in matrix input file}
begin
 readcollabels := true;
 matrixform.radiobutton18.Enabled := true;
end;

procedure Tmatrixform.RadioButton4Click(Sender: TObject);
{Input the matrix to be edited}
var
```

```
templabel : Tstr;
ch : char;
i,j : longint;
begin
if readrowlabels and readcollabels then
 matrixform.radiobutton19.Enabled := true;
matrixform.OpenDialog1.execute;
assignfile (infile,matrixform.OpenDialog1.FileName);
reset (infile);
system.settextbuf (infile, inbuffer);
{If a row/column header exists in the input matrix, use a deterministic loop}
if readheader then
 begin
 readln (infile, freqmatrix.nrofrows);
 readln (infile, freqmatrix.nrofcols);
 {If there are column labels, read then first}
 if readcollabels then
  begin
  i := 0;
  while (not seekeoln (infile)) do
   begin
    for j := 0 to (maxstrlength - 1) do
     templabel [j] := ' ';
    ch := '$';
    j := 0;
   while ch in [chr(33)..chr(126)] do
    begin
     read (infile, ch);
     templabel [j] := ch;
     j := j + 1;
    end;
   freqmatrix.collabellist [i] := templabel;
   i := i + 1;
  end;
 readln (infile);
end;
{Read the matrix; if there are row labels, read each label at the start of each row}
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
 if readrowlabels then
begin
 for j := 0 to (maxstrlength - 1) do
  templabel [j] := ' ';
ch := '$';
j := 0;
while (ch <> ' ') and (ch <> chr(9)) do
 begin
  read (infile, ch);
  templabel [j] := ch;
  j := j + 1;
```

```
      end;
     freqmatrix.rowlabellist [i] := templabel;
    end;
   for j := 0 to (freqmatrix.nrofcols - 1) do
    read (infile, freqmatrix.m [i,j]);
   readln (infile);
  end;
 end
{If a row/column header does not exist in the input matrix, use a nonterministic loop}
else
begin
 {If there are column labels, read then first}
 if readcollabels then
  begin
   i := 0;
   while (not seekeoln (infile)) do
    begin
     for j := 0 to (maxstrlength - 1) do
      templabel [j] := ' ';
     ch := '$';
     j := 0;
     while ch in [chr(33)..chr(126)] do
      begin
       read (infile, ch);
       templabel [j] := ch;
       j := j + 1;
      end;
     freqmatrix.collabellist [i] := templabel;
     i := i + 1;
    end;
   readln (infile);
  end;
{Read the matrix; if there are row labels, read each label at the start of each row}
i := 0;
while not eof (infile) do
 begin
  if readrowlabels then
   begin
    for j := 0 to (maxstrlength - 1) do
     templabel [j] := ' ';
    ch := chr (32);
    while not (ch in [chr(33)..chr(126)]) do
     read (infile, ch);
    templabel [0] := ch;
    j := 1;
    while (ch <> ' ') and (ch <> chr(9)) do
     begin
      read (infile, ch);
      templabel [j] := ch;
      j := j + 1;
```

```
  end;
  freqmatrix.rowlabellist [i] := templabel;
 end;
j := 0;
while (not seekeoln (infile)) and (not seekeof (infile)) do
 begin
  read (infile, freqmatrix.m [i,j]);
  j := j + 1;
 end;
if eoln (infile) then
    begin
     readln (infile);
     i := i + 1;
    end;
   end;
 end;
freqmatrix.nrofrows := i;
freqmatrix.nrofcols := j;
matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
closefile (infile);
matrixform.memo1.lines.Add('Matrix loaded');
end;

{*** NORMALIZATION ***}
procedure Tmatrixform.RadioButton56Click(Sender: TObject);
{Cosine normalization}
var
 sumofsquares : single;
 magnitude : single;
 i, j : longint;
begin
 for i := 0 to (freqmatrix.nrofrows - 1) do
  begin
   sumofsquares := 0;
   for j := 0 to (freqmatrix.nrofcols - 1) do
    sumofsquares := sumofsquares + (freqmatrix.m [i,j] * freqmatrix.m [i,j]);
   magnitude := sqrt (sumofsquares);
   for j := 0 to (freqmatrix.nrofcols - 1) do
    if magnitude > 0 then
     freqmatrix.m [i,j] := freqmatrix.m [i,j] / magnitude;
  end;
 matrixform.memo1.lines.add ('Matrix normalized by cosine');
end;

procedure Tmatrixform.RadioButton32Click(Sender: TObject);
{Normalize by average document length}
var
 globalsum : single;
```

```
 globalmean : single;
 rowsum : single;
 i,j : longint;
begin
 globalsum := 0;
 for i := 0 to (freqmatrix.nrofrows - 1) do
  for j := 0 to (freqmatrix.nrofcols - 1) do
   globalsum := globalsum + freqmatrix.m [i,j];
 globalmean := globalsum / freqmatrix.nrofrows;
 for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  rowsum := 0;
  for j := 0 to (freqmatrix.nrofcols - 1) do
   rowsum := rowsum + freqmatrix.m [i,j];
  for j := 0 to (freqmatrix.nrofcols - 1) do
   if rowsum <> 0 then
    freqmatrix.m [i,j] := freqmatrix.m [i,j] * (globalmean / rowsum);
 end;
 normalized := true;
 matrixform.memo1.lines.add('Matrix normalized by average document length');
end;

{*** DIMENSIONALITY REDUCTION: HEURISTICS ***}

procedure Tmatrixform.RadioButton6Click(Sender: TObject);
{Remove frequency-1 columns}
var
 total : single;
 i,j : longint;
begin
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
   total := 0;
   for j := 0 to (freqmatrix.nrofrows - 1) do
    total := total + freqmatrix.m [j,i];
   if total <= 1 then
    freqmatrix.collabellist [i] := 'xxx';
  end;
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofcols := 0;
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
   if freqmatrix.collabellist [i] <> 'xxx' then
    begin
     tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [i];
     for j := 0 to (freqmatrix.nrofrows - 1) do
      tempfreqmatrix.m [j, tempfreqmatrix.nrofcols] := freqmatrix.m [j,i];
     tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
    end;
```

```
 end;
 freqmatrix := tempfreqmatrix;
 matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
 matrixform.Memo1.Lines.add ('Frequency-1 columns removed');
end;

procedure Tmatrixform.RadioButton7Click(Sender: TObject);
{Remove function-word columns from matrix}
var
 ch : char;
 functionword : boolean;
 i,j,k : longint;
begin
 {Load list of function words from file}
 matrixform.opendialog1.execute;
 assignfile (infile, matrixform.opendialog1.filename);
 reset (infile);
 system.settextbuf (infile, inbuffer);
 i := 0;
 while not eof (infile) do
  begin
   for j := 0 to (maxstrlength - 1) do
    keywordlist.list [i,j] := ' ';
   j := 0;
   while (not eoln (infile)) and (not eof (infile)) do
    begin
     read (infile, ch);
     keywordlist.list [i,j] := ch;
     j := j + 1;
    end;
   matrixform.Memo1.lines.add (keywordlist.list [i]);
   if not eof (infile) then
    readln (infile);
   i := i + 1;
  end;
keywordlist.length := i;
{Initialize temporary matrix}
tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
tempfreqmatrix.nrofcols := 0;
tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  functionword := false;
  for j := 0 to (keywordlist.length - 1) do
   begin
    if freqmatrix.collabellist [i] = keywordlist.list [j] then
     functionword := true;
   end;
  if not functionword then
```

```
  begin
   for k := 0 to (freqmatrix.nrofrows - 1) do
    tempfreqmatrix.m [k, tempfreqmatrix.nrofcols] := freqmatrix.m [k, i];
   tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [i];
   tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
  end;
 end;
{Finish up}
freqmatrix := tempfreqmatrix;
matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
 matrixform.memo1.lines.add ('Function words removed');
end;

{*** DIMENSIONALITY REDUCTION: RETAIN COLUMNS SELECTED BY
KEYWORD ***}

procedure Tmatrixform.RadioButton49Click(Sender: TObject);
{Read list of keywords from external file. Any duplicates are merged}
var
 ch : char;
 tempstr : Tstr;
 found : boolean;
 i : longint;
begin
matrixform.opendialog1.execute;
assignfile (infile, matrixform.opendialog1.filename);
reset (infile);
system.settextbuf (infile, inbuffer);
keywordlist.length := -1;
while not eof (infile) do
 begin
  for i := 0 to (maxstrlength - 1) do
   tempstr [i] := ' ';
  i := 0;
  while (not eoln (infile)) and (not eof (infile)) do
   begin
    read (infile, ch);
    tempstr [i] := ch;
    i := i + 1;
   end;
{Look for tempstr in existing list, and if it's not already there add it}
found := false;
for i := 0 to (keywordlist.length - 1) do
 if keywordlist.list [i] = tempstr then
  found := true;
if not found then
 begin
  keywordlist.length := keywordlist.length + 1;
  keywordlist.list [keywordlist.length] := tempstr;
```

```
   end;
  if not eof (infile) then
   readln (infile);
 end;
keywordlist.length := keywordlist.length + 1;
matrixform.memo1.lines.add ('Keyword list read. Nr of keywords = ' + inttostr
(keywordlist.length));
end;

procedure Tmatrixform.RadioButton50Click(Sender: TObject);
{Select keyword columns in frequency matrix for retention}
var
i,j,k : longint;
begin
{Initialize temporary matrix}
tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
tempfreqmatrix.nrofcols := 0;
tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
for i := 0 to (keywordlist.length - 1) do
 begin
  for j := 0 to (freqmatrix.nrofcols - 1) do
   begin
    if freqmatrix.collabellist [j] = keywordlist.list [i] then
     begin
      for k := 0 to (freqmatrix.nrofrows - 1) do
       tempfreqmatrix.m [k, i] := freqmatrix.m [k, j];
      tempfreqmatrix.collabellist [i] := freqmatrix.collabellist [j];
      tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
     end;
   end;
 end;
{Finish up}
freqmatrix := tempfreqmatrix;
matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
matrixform.memo1.lines.add ('Keyword columns retained');
end;

{*** DIMENSIONALITY REDUCTION: REMOVE SPECIFIC ROWS / COLUMNS
***}

procedure Tmatrixform.RadioButton15Click(Sender: TObject);
{Remove a specific row from the matrix}
var
 removeindex : longint;
 rowindex : longint;
 i,j : longint;
begin
 {get index of row to remove}
 removeindex := strtoint (matrixform.edit8.Text);
```

```
tempfreqmatrix.nrofcols := freqmatrix.nrofcols;
tempfreqmatrix.collabellist := freqmatrix.collabellist;
tempfreqmatrix.nrofrows := 0;
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  if i <> removeindex then
   begin
    for j := 0 to (freqmatrix.nrofcols - 1) do
     tempfreqmatrix.m [tempfreqmatrix.nrofrows, j] := freqmatrix.m [i,j];
    tempfreqmatrix.rowlabellist [tempfreqmatrix.nrofrows] := freqmatrix.rowlabellist [i];
     tempfreqmatrix.nrofrows := tempfreqmatrix.nrofrows + 1;
    end;
  end;
 freqmatrix := tempfreqmatrix;
 matrixform.edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton20Click(Sender: TObject);
{Remove selected column from matrix}
var
 removeindex : longint;
 rowindex : longint;
 i,j : longint;
begin
 {Get index of column to be removed}
 removeindex := strtoint (matrixform.edit9.Text);
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofcols := 0;
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
   if i <> removeindex then
    begin
     for j := 0 to (freqmatrix.nrofrows - 1) do
      tempfreqmatrix.m [j, tempfreqmatrix.nrofcols] := freqmatrix.m [j,i];
     tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [i];
     tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
    end;
  end;
 freqmatrix := tempfreqmatrix;
 matrixform.edit14.Text := inttostr (freqmatrix.nrofcols);
 matrixform.memo1.lines.add ('Done');
end;

{*** DIMENSIONALITY REDUCTION: STANDARD DEVIATION ***}

procedure Tmatrixform.RadioButton13Click(Sender: TObject);
{Boolean for removal of columns with std dev < n}
begin
```

```
 removecolsbyvariance1 := true;
 matrixform.RadioButton11.enabled := false;
 matrixform.edit3.enabled := false;
end;

procedure Tmatrixform.Edit1Change(Sender: TObject);
{Get user input}
begin
 nreal := strtofloat (matrixform.Edit1.Text);
end;

procedure Tmatrixform.RadioButton11Click(Sender: TObject);
{Booleans for retention of n highest std dev columns}
begin
 removecolsbyvariance2 := true;
 matrixform.RadioButton13.enabled := false;
 matrixform.edit1.enabled := false;
end;

procedure Tmatrixform.Edit7Change(Sender: TObject);
{Get user input}
begin
 ninteger := strtoint (matrixform.Edit7.Text);
end;

procedure Tmatrixform.Edit3Change(Sender: TObject);
{Get user input}
begin
 ninteger := strtoint (matrixform.Edit3.text);
end;

procedure Tmatrixform.RadioButton53Click(Sender: TObject);
{Boolean for matrix weighting by std dev}
begin
 weightbystddev := true;
end;

procedure Tmatrixform.RadioButton78Click(Sender: TObject);
{Boolean to output sorted column variance vector}
begin
 outputcolvariancevector := true;
end;

procedure Tmatrixform.RadioButton28Click(Sender: TObject);
{Boolean to output sorted column std dev vector}
begin
 outputcolstddevvector := true;
end;

procedure Tmatrixform.RadioButton9Click(Sender: TObject);
```

```
{Boolean to output profile sorted by descending column std dev}
begin
 outputcolvarianceprofile := true;
end;

procedure Tmatrixform.RadioButton73Click(Sender: TObject);
{Boolean to output list of n seleted lexical types}
begin
 outputselectedlexicaltypelist := true;
end;

procedure Tmatrixform.RadioButton57Click(Sender: TObject);
{Apply dimensionality reduction by variance / standard deviation}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 frequency : single;
 mean : single;
 sumofsquareddeviations : single;
 variance : single;
 stddev : single;
 largest : single;
 index : longint;
 i,j,k : longint;
begin
{Make a profile of variances and standard deviations, one for each column}
{Initialize the profile}
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  profile.list [i].variablename := freqmatrix.collabellist [i];
  profile.list [i].frequency := 0;
  for j := 0 to (freqmatrix.nrofrows - 1) do
   profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
  profile.selected [i] := false;
 end;
profile.length := freqmatrix.nrofcols;
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  mean := profile.list [i].frequency / freqmatrix.nrofrows;
  {Sum squared deviations from mean}
  sumofsquareddeviations := 0;
  for j := 0 to (freqmatrix.nrofrows - 1) do
   sumofsquareddeviations := sumofsquareddeviations +
                        ((freqmatrix.m [j,i] - mean) *
                         (freqmatrix.m [j,i] - mean));
  {Variance}
  profile.list [i].variance := sumofsquareddeviations / freqmatrix.nrofrows;
  profile.list [i].stddev := sqrt (profile.list [i].variance);
 end;
if removecolsbyvariance1 then
```

```
{If remove columns with stddev < n}
begin
 {Create a temporary matrix and then populate from the input matrix}
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.nrofcols := 0;
 for i := 0 to (profile.length - 1) do
  begin
   largest := -1;
   for j := 0 to (profile.length - 1) do
    if (profile.list [j].stddev > largest) and (not profile.selected [j]) then
     begin
      largest := profile.list [j].stddev;
      index := j;
     end;
    if largest > nreal then
     begin
      for j := 0 to (freqmatrix.nrofrows - 1) do
       tempfreqmatrix.m [j, tempfreqmatrix.nrofcols] := freqmatrix.m [j, index];
      tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist
[index];
      tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
     end;
    profile.selected [index] := true;
   end;
  freqmatrix := tempfreqmatrix;
 end;
if removecolsbyvariance2 then
 {If retain n highest std dev columns}
 begin
  {Create a temporary matrix and then populate from the input matrix}
  tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
  tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
  tempfreqmatrix.nrofcols := 0;
  for i := 0 to (ninteger-1) do
   begin
    largest := -1;
    for j := 0 to (profile.length - 1) do
     begin
      if (profile.list [j].stddev > largest) and (not profile.selected [j]) then
       begin
        largest := profile.list [j].stddev;
        index := j;
       end;
     end;
 end;
for k := 0 to (freqmatrix.nrofrows - 1) do
 tempfreqmatrix.m [k, tempfreqmatrix.nrofcols] := freqmatrix.m [k, index];
tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [index];
tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
profile.selected [index] := true;
```

```
  end;
  freqmatrix := tempfreqmatrix;
 end;
 {If the matrix has changed by the above options AND the following outputs are required,
then
  the profile needs to be recalculated}
 if (removecolsbyvariance1 or removecolsbyvariance2) and (outputcolvariancevector or
outputcolstddevvector or outputcolvarianceprofile) then
  begin
   for i := 0 to (freqmatrix.nrofcols - 1) do
   begin
    profile.list [i].variablename := freqmatrix.collabellist [i];
    profile.list [i].frequency := 0;
    for j := 0 to (freqmatrix.nrofrows - 1) do
     profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
    profile.selected [i] := false;
   end;
  profile.length := freqmatrix.nrofcols;
  for i := 0 to (freqmatrix.nrofcols - 1) do
   begin
    mean := profile.list [i].frequency / freqmatrix.nrofrows;
    sumofsquareddeviations := 0;
    for j := 0 to (freqmatrix.nrofrows - 1) do
     sumofsquareddeviations := sumofsquareddeviations +
                               ((freqmatrix.m [j,i] - mean) *
                                (freqmatrix.m [j,i] - mean));
     profile.list [i].variance := sumofsquareddeviations / freqmatrix.nrofrows;
     profile.list [i].stddev := sqrt (profile.list [i].variance);
    end;
  end;
 if outputcolvariancevector then
  begin
   assignfile (outfile,'variancevector.txt');
   rewrite (outfile);
   system.settextbuf (outfile, buffer);
   sortedprofile := sortprofile;
   for i := 0 to (sortedprofile.length - 1) do
    writeln (outfile, sortedprofile.list [i].variance:12:10);
   closefile (outfile);
  end;
 if outputcolstddevvector then
  begin
   assignfile (outfile,'stddevvector.txt');
   rewrite (outfile);
   system.settextbuf (outfile, buffer);
   sortedprofile := sortprofile;
   for i := 0 to (sortedprofile.length - 1) do
    writeln (outfile, sortedprofile.list [i].stddev:12:10);
   closefile (outfile);
  end;
```

```
if outputcolvarianceprofile then
 begin
  assignfile (outfile,'varianceprofile.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  sortedprofile := sortprofile;
  for i := 0 to (sortedprofile.length - 1) do
   writeln (outfile, sortedprofile.list [i].variablename, ' ',
                  sortedprofile.list [i].frequency:12:1, ' ',
                  sortedprofile.list [i].variance:12:3, ' ',
                  sortedprofile.list [i].stddev:12:3);
  closefile (outfile);
 end;
if weightbystddev then
 begin
  for i := 0 to (freqmatrix.nrofcols - 1) do
   for j := 0 to (freqmatrix.nrofrows - 1) do
    freqmatrix.m [j,i] := freqmatrix.m [j,i] * profile.list [i].stddev;
 end;
if outputselectedlexicaltypelist then
 begin
  assignfile (outfile,'lexicaltypesbyvariance.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  for i := 0 to (freqmatrix.nrofcols -1) do
   writeln (outfile, freqmatrix.collabellist [i]);
  closefile (outfile);
 end;.
 matrixform.edit13.text := inttostr (freqmatrix.nrofrows);
 matrixform.edit14.text := inttostr (freqmatrix.nrofcols);
 matrixform.Memo1.lines.add ('Dimensionality reduction by standard deviation complete');
end;
                {*** DIMENSIONALITY REDUCTION: TF.IDF ***}

procedure Tmatrixform.RadioButton23Click(Sender: TObject);
{Boolean for removal of columns with tf.idf < n}
begin
 removecolsbytfidf1 := true;
 matrixform.RadioButton30.enabled := false;
 matrixform.edit7.enabled := false;
end;

procedure Tmatrixform.Edit2Change(Sender: TObject);
{Get user input}
begin
 nreal := strtofloat (matrixform.Edit2.Text);
end;

procedure Tmatrixform.RadioButton30Click(Sender: TObject);
{Boolean to retain n highest tf.idf columns}
```

```
begin
 removecolsbytfidf2 := true;
 matrixform.RadioButton23.enabled := false;
 matrixform.edit2.enabled := false;
end;

procedure Tmatrixform.RadioButton65Click(Sender: TObject);
{Boolean to weight matrix by tf.idf}
begin
 weightbytfidf := true;
end;

procedure Tmatrixform.RadioButton10Click(Sender: TObject);
{Boolean to output sorted column tf.idf vector}
begin
 outputtfidfvector := true;
end;

procedure Tmatrixform.RadioButton21Click(Sender: TObject);
{Boolean to output profile sorted by descending column tf.idf}
begin
 outputtfidfprofile := true;
end;

procedure Tmatrixform.RadioButton74Click(Sender: TObject);
{Boolean to output list of n seleted lexical types}
begin
 outputselectedlexicaltypelist := true;
end;

procedure Tmatrixform.RadioButton66Click(Sender: TObject);
{Apply dimensionality reduction by tf.idf}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 typefrequency : single;
 tokenfrequency : single;
 idf : single;
 tfidf : single;
 largest : single;
 index : longint;
 i,j,k : longint;
begin
 {Make a tf.idf profile, one for each column}
 {Initialize the profile}
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
   profile.list [i].variablename := freqmatrix.collabellist [i];
   profile.list [i].frequency := 0;
   for j := 0 to (freqmatrix.nrofrows - 1) do
```

```
  profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
 profile.selected [i] := false;
end;
profile.length := freqmatrix.nrofcols;
{Add tfidf to profile, one for each column}
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  typefrequency := 0;
  tokenfrequency := 0;
  for j := 0 to (freqmatrix.nrofrows - 1) do
  begin
   if freqmatrix.m [j,i] > 0 then
    typefrequency := typefrequency + 1;
   tokenfrequency := tokenfrequency + freqmatrix.m [j,i];
  end;
 if typefrequency = 0 then
  begin
   profile.list [i].idf := 0;
   profile.list [i].tfidf := 0;
  end
 else
  begin
   idf := log2 (freqmatrix.nrofrows / typefrequency);
   matrixform.Memo1.lines.add (floattostrf (idf, fffixed, 12, 3));
   profile.list [i].idf := idf;
    tfidf := idf * tokenfrequency;
    profile.list [i].tfidf := tfidf;
   end;
 end;
if removecolsbytfidf1 then
 {If remove columns with tf.idf < n}
 begin
  tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
  tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
  tempfreqmatrix.nrofcols := 0;
  for i := 0 to (profile.length - 1) do
   begin
    largest := -1;
    for j := 0 to (profile.length - 1) do
     if (profile.list [j].tfidf > largest) and (not profile.selected [j]) then
      begin
       largest := profile.list [j].tfidf;
       index := j;
      end;
     if largest > nreal then
      begin
       for j := 0 to (freqmatrix.nrofrows - 1) do
        tempfreqmatrix.m [tempfreqmatrix.nrofcols, j] := freqmatrix.m [index, j];
       tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist
[index];
```

```
    tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
   end;
  profile.selected [index] := true;
 end;
freqmatrix := tempfreqmatrix;
end;
if removecolsbytfidf2 then
{If retain n highest tf.idf columns}
begin
tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
tempfreqmatrix.nrofcols := 0;
for i := 0 to (ninteger-1) do
begin
largest := -1;
for j := 0 to (profile.length - 1) do
 begin
  if (profile.list [j].tfidf > largest) and (not profile.selected [j]) then
   begin
    largest := profile.list [j].tfidf;
    index := j;
   end;
 end;
for k := 0 to (freqmatrix.nrofrows - 1) do
    tempfreqmatrix.m [k,tempfreqmatrix.nrofcols] := freqmatrix.m [k,index];
   tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [index];
   tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
   profile.selected [index] := true;
  end;
 freqmatrix := tempfreqmatrix;
end;
{If the matrix has changed by the above options AND the following outputs are required,
then
 the profile needs to be recalculated}
if (removecolsbytfidf1 or removecolsbytfidf2) and (outputtfidfvector or outputtfidfprofile)
then
 begin
  for i := 0 to (freqmatrix.nrofcols - 1) do
   begin
    profile.list [i].variablename := freqmatrix.collabellist [i];
    profile.list [i].frequency := 0;
    for j := 0 to (freqmatrix.nrofrows - 1) do
     profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
    profile.selected [i] := false;
   end;
 profile.length := freqmatrix.nrofcols;
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
   typefrequency := 0;
   tokenfrequency := 0;
```

```
for j := 0 to (freqmatrix.nrofrows - 1) do
 begin
  if freqmatrix.m [j,i] > 0 then
   typefrequency := typefrequency + 1;
  tokenfrequency := tokenfrequency + freqmatrix.m [j,i];
 end;
idf := log2 (freqmatrix.nrofrows / typefrequency);
profile.list [i].idf := idf;
tfidf := idf * tokenfrequency;
profile.list [i].tfidf := tfidf;
  end;
 end;
if outputtfidfvector then
 begin
  assignfile (outfile,'tfidfvector.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  sortedprofile := sortprofile;
  for i := 0 to (sortedprofile.length - 1) do
   writeln (outfile, sortedprofile.list [i].tfidf:12:3);
  closefile (outfile);
 end;
if outputtfidfprofile then
 begin
  assignfile (outfile,'tfidfprofile.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  sortedprofile := sortprofile;
  for i := 0 to (sortedprofile.length - 1) do
   writeln (outfile, sortedprofile.list [i].variablename, ' ',
                  sortedprofile.list [i].frequency:12:1, ' ',
                  sortedprofile.list [i].tfidf:12:3);
  closefile (outfile);
 end;
if weightbytfidf then
 begin
  for i := 0 to (freqmatrix.nrofcols - 1) do
   for j := 0 to (freqmatrix.nrofrows - 1) do
    freqmatrix.m [j,i] := freqmatrix.m [j,i] * profile.list [i].tfidf;
 end;
if outputselectedlexicaltypelist then
 begin
  assignfile (outfile,'lexicaltypesbytfidf.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  for i := 0 to (freqmatrix.nrofcols -1) do
   writeln (outfile, freqmatrix.collabellist [i]);
  closefile (outfile);
 end;
matrixform.edit13.Text := inttostr (freqmatrix.nrofrows);
```

```
 matrixform.edit14.text := inttostr (freqmatrix.nrofcols);
 matrixform.Memo1.lines.add ('Dimensionality reduction by tf.idf complete');
end;
                {*** POISSON TERM DISTRIBUTION ***}

procedure Tmatrixform.RadioButton45Click(Sender: TObject);
{Boolean to remove columns with mean-variance criterion < n}
begin
 removecolsbypoisson1 := true;
 matrixform.RadioButton46.enabled := false;
 matrixform.edit17.enabled := false;
end;

procedure Tmatrixform.Edit16Change(Sender: TObject);
{Get user input}
begin
 nreal := strtofloat (matrixform.Edit16.Text);
end;

procedure Tmatrixform.RadioButton46Click(Sender: TObject);
{Boolean to retain n columns with highest mean-variance criterion}
begin
 removecolsbypoisson2 := true;
 matrixform.RadioButton45.enabled := false;
 matrixform.edit16.enabled := false;
end;

procedure Tmatrixform.Edit17Change(Sender: TObject);
{Get user input}
begin
 ninteger := strtoint (matrixform.Edit17.Text);
end;

procedure Tmatrixform.RadioButton68Click(Sender: TObject);
{Boolean to weight matrix by Poisson index}
begin
 weightbypoisson := true;
end;

procedure Tmatrixform.RadioButton37Click(Sender: TObject);
{Boolean to output sorted mean-variance criterion vector}
begin
 outputpoissonvector := true;
end;

procedure Tmatrixform.RadioButton39Click(Sender: TObject);
{Boolean to output sorted mean-variance criterion profile}
begin
 outputpoissonprofile := true;
end;
```

```
procedure Tmatrixform.RadioButton76Click(Sender: TObject);
{Boolean to output list of n seleted lexical types}
begin
 outputselectedlexicaltypelist := true;
end;

procedure Tmatrixform.RadioButton69Click(Sender: TObject);
{Dimensionality reduction by Poisson term distribution}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 columnvector : Trealvector;
 largest : single;
 index : longint;
 sum : single;
 mean : single;
 sumofsquares : single;
 variance : single;
 difference : single;
 i,j,k : longint;
begin
 {{Make a mean-variance criterion profile, one for each column}
 {Initialize the profile}
 for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  profile.list [i].variablename := freqmatrix.collabellist [i];
  profile.list [i].frequency := 0;
  for j := 0 to (freqmatrix.nrofrows - 1) do
   profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
  profile.selected [i] := false;
 end;
profile.length := freqmatrix.nrofcols;
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  {Get current column vector}
  for j := 0 to (freqmatrix.nrofrows - 1) do
   columnvector.v [j] := freqmatrix.m [j,i];
  columnvector.length := freqmatrix.nrofrows;
  {Calculate mean of current vector}
   sum := 0;
   for j := 0 to (columnvector.length - 1) do
    sum := sum + columnvector.v [j];
   mean := sum / columnvector.length;
   {Calculate variance of currentvector}
   sumofsquares := 0;
   for j := 0 to (columnvector.length - 1) do
    sumofsquares := sumofsquares + ((columnvector.v [j] - mean) * (columnvector.v [j] -
mean));
   variance := sumofsquares / columnvector.length;
```

```
 {Calculate mean / variance difference}
 difference := abs (mean - variance);
 profile.list [i].poisson := difference;
end;
if removecolsbypoisson1 then
{If remove columns by mean-variance criterion < n}
 begin
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.nrofcols := 0;
  for i := 0 to (profile.length - 1) do
   begin
    largest := -1;
    for j := 0 to (profile.length - 1) do
     if (profile.list [j].poisson > largest) and (not profile.selected [j]) then
      begin
       largest := profile.list [j].poisson;
       index := j;
      end;
     if largest > nreal then
      begin
       for j := 0 to (freqmatrix.nrofrows - 1) do
        tempfreqmatrix.m [j, tempfreqmatrix.nrofcols] := freqmatrix.m [j, index];
       tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist
[index];
        tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
      end;
     profile.selected [index] := true;
    end;
 freqmatrix := tempfreqmatrix;
end;
if removecolsbypoisson2 then
 {If retain n highest mean-variance citerion columns}
 begin
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.nrofcols := 0;
 for i := 0 to (ninteger-1) do
  begin
largest := -1;
for j := 0 to (profile.length - 1) do
 begin
  if (profile.list [j].poisson > largest) and (not profile.selected [j]) then
   begin
    largest := profile.list [j].poisson;
    index := j;
   end;
 end;
for k := 0 to (freqmatrix.nrofrows - 1) do
 tempfreqmatrix.m [k, tempfreqmatrix.nrofcols] := freqmatrix.m [k, index];
```

```
    tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [index];
    tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
    profile.selected [index] := true;
   end;
  freqmatrix := tempfreqmatrix;
 end;
 {If the matrix has changed by the above options AND the following outputs are required,
then
 the profile needs to be recalculated}
 if (removecolsbypoisson1 or removecolsbypoisson2) and (outputpoissonvector or
outputpoissonprofile) then
  begin
   for i := 0 to (freqmatrix.nrofcols - 1) do
    begin
     profile.list [i].variablename := freqmatrix.collabellist [i];
     profile.list [i].frequency := 0;
     for j := 0 to (freqmatrix.nrofrows - 1) do
      profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
     profile.selected [i] := false;
    end;
   profile.length := freqmatrix.nrofcols;
   for i := 0 to (freqmatrix.nrofcols - 1) do
    begin
     {Get current column vector}
     for j := 0 to (freqmatrix.nrofrows - 1) do
      columnvector.v [j] := freqmatrix.m [j,i];
     columnvector.length := freqmatrix.nrofrows;
     {Calculate mean of current vector}
     sum := 0;
     for j := 0 to (columnvector.length - 1) do
      sum := sum + columnvector.v [j];
     mean := sum / columnvector.length;
     {Calculate variance of currentvector}
     sumofsquares := 0;
     for j := 0 to (columnvector.length - 1) do
      sumofsquares := sumofsquares + ((columnvector.v [j] - mean) * (columnvector.v [j] -
mean));
     variance := sumofsquares / columnvector.length;
     {Calculate mean / variance difference}
     difference := abs (mean - variance);
     profile.list [i].poisson := difference;
    end;
 end;
if outputpoissonvector then
 begin
  assignfile (outfile,'poissonvector.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  sortedprofile := sortprofile;
  for i := 0 to (sortedprofile.length - 1) do
```

```
  writeln (outfile, sortedprofile.list [i].poisson:12:3);
 closefile (outfile);
end;
if outputpoissonprofile then
 begin
 assignfile (outfile,'poissonprofile.txt');
 rewrite (outfile);
 system.settextbuf (outfile, buffer);
 sortedprofile := sortprofile;
 for i := 0 to (sortedprofile.length - 1) do
  writeln (outfile, sortedprofile.list [i].variablename, ' ',
                    sortedprofile.list [i].frequency:12:1, ' ',
                    sortedprofile.list [i].poisson:12:3);
 closefile (outfile);
end;
if weightbypoisson then
 begin
 for i := 0 to (freqmatrix.nrofcols - 1) do
  for j := 0 to (freqmatrix.nrofrows - 1) do
   freqmatrix.m [j,i] := freqmatrix.m [j,i] * profile.list [i].poisson;
end;
if outputselectedlexicaltypelist then
 begin
 assignfile (outfile,'lexicaltypesbypoisson.txt');
 rewrite (outfile);
  system.settextbuf (outfile, buffer);
  for i := 0 to (freqmatrix.nrofcols -1) do
   writeln (outfile, freqmatrix.collabellist [i]);
  closefile (outfile);
 end;
 matrixform.edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.edit14.text := inttostr (freqmatrix.nrofcols);
 matrixform.Memo1.lines.add ('Dimensionality reduction by Poisson term distribution
complete');
end;
                        {*** ENTROPY ***}

procedure Tmatrixform.RadioButton47Click(Sender: TObject);
{Boolean to remove columns with entropy < n}
begin
 removecolsbysignal1 := true;
 matrixform.RadioButton48.enabled := false;
 matrixform.edit19.enabled := false;
end;

procedure Tmatrixform.Edit18Change(Sender: TObject);
{Get user input}
begin
 nreal := strtofloat (matrixform.Edit18.Text);
end;
```

```
procedure Tmatrixform.RadioButton48Click(Sender: TObject);
{Boolean to retain n highest-entropy columns}
begin
 removecolsbysignal2 := true;
 matrixform.RadioButton47.enabled := false;
 matrixform.edit18.enabled := false;
end;

procedure Tmatrixform.Edit19Change(Sender: TObject);
{Get user input}
begin
 ninteger := strtoint (matrixform.Edit19.Text);
end;

procedure Tmatrixform.RadioButton72Click(Sender: TObject);
{Boolean to weight by entropy}
begin
 weightbysignal := true;
end;

procedure Tmatrixform.RadioButton38Click(Sender: TObject);
{Boolean to output sorted entropy vector}
begin
 outputsignalvector := true;
end;

procedure Tmatrixform.RadioButton71Click(Sender: TObject);
{Boolean to output sorted entropy profile}
begin
 outputsignalprofile := true;
end;

procedure Tmatrixform.RadioButton77Click(Sender: TObject);
{Boolean to output list of n seleted lexical types}
begin
 outputselectedlexicaltypelist := true;
end;

procedure Tmatrixform.RadioButton70Click(Sender: TObject);
 {Dimensionality reduction by entropy}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 currentfrequency : single;
 totalfrequency : single;
 probability : single;
 sum : single;
 largest : single;
 index : longint;
```

```
 i,j,k : longint;
begin
 {Make a profile of entropies and signals, one for each column}
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
   profile.list [i].variablename := freqmatrix.collabellist [i];
   profile.list [i].frequency := 0;
   for j := 0 to (freqmatrix.nrofrows - 1) do
    profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
   totalfrequency := profile.list [i].frequency;
 if totalfrequency = 0 then
  begin
   profile.list [i].entropy := 0;
   profile.list [i].signal := 0;
  end
 else
  begin
   profile.list [i].entropy := 0;
   for j := 0 to (freqmatrix.nrofrows - 1) do
    begin
     currentfrequency := freqmatrix.m [j,i];
     probability := currentfrequency / totalfrequency;
     if probability <> 0 then
      profile.list [i].entropy := profile.list [i].entropy + (probability * log2(1/probability));
    end;
    profile.list [i].signal := log2(totalfrequency) - profile.list [i].entropy;
   end;
  profile.selected [i] := false;
 end;
profile.length := freqmatrix.nrofcols;
if removecolsbysignal1 then
 {If remove columns with signal < n}
 begin
  tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
  tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
  tempfreqmatrix.nrofcols := 0;
  for i := 0 to (profile.length - 1) do
   begin
    largest := -1;
    for j := 0 to (profile.length - 1) do
 if (profile.list [j].signal > largest) and (not profile.selected [j]) then
  begin
   largest := profile.list [j].signal;
   index := j;
  end;
if largest > nreal then
 begin
  for j := 0 to (freqmatrix.nrofrows - 1) do
   tempfreqmatrix.m [j, tempfreqmatrix.nrofcols] := freqmatrix.m [j, index];
```

```
      tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist
[index];
      tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
     end;
    profile.selected [index] := true;
   end;
  freqmatrix := tempfreqmatrix;
 end;
if removecolsbysignal2 then
 {If retain n highest-signal columns}
begin
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.nrofcols := 0;
 for i := 0 to (ninteger-1) do
  begin
   largest := -1;
   for j := 0 to (profile.length - 1) do
    begin
     if (profile.list [j].signal > largest) and (not profile.selected [j]) then
      begin
       largest := profile.list [j].signal;
       index := j;
      end;
    end;
     for k := 0 to (freqmatrix.nrofrows - 1) do
      tempfreqmatrix.m [k, tempfreqmatrix.nrofcols] := freqmatrix.m [k, index];
     tempfreqmatrix.collabellist [tempfreqmatrix.nrofcols] := freqmatrix.collabellist [index];
     tempfreqmatrix.nrofcols := tempfreqmatrix.nrofcols + 1;
     profile.selected [index] := true;
    end;
   freqmatrix := tempfreqmatrix;
  end;
 {If the matrix has changed by the above options AND the following outputs are required,
then
  the profile needs to be recalculated}
 if (removecolsbysignal1 or removecolsbysignal2) and (outputsignalvector or
outputsignalprofile) then
  begin
   for i := 0 to (freqmatrix.nrofcols - 1) do
begin
 profile.list [i].variablename := freqmatrix.collabellist [i];
 profile.list [i].frequency := 0;
 for j := 0 to (freqmatrix.nrofrows - 1) do
  profile.list [i].frequency := profile.list [i].frequency + freqmatrix.m [j,i];
 totalfrequency := profile.list [i].frequency;
 if totalfrequency = 0 then
  begin
   profile.list [i].entropy := 0;
   profile.list [i].signal := 0;
```

```
 end
else
 begin
  profile.list [i].entropy := 0;
  for j := 0 to (freqmatrix.nrofrows - 1) do
   begin
    currentfrequency := freqmatrix.m [j,i];
    probability := currentfrequency / totalfrequency;
    if probability <> 0 then
     profile.list [i].entropy := profile.list [i].entropy + (probability * log2(1/probability));
     end;
     profile.list [i].signal := log2(totalfrequency) - profile.list [i].entropy;
    end;
   profile.selected [i] := false;
  end;
 profile.length := freqmatrix.nrofcols;
end;
if weightbysignal then
 begin
  for i := 0 to (freqmatrix.nrofcols - 1) do
   for j := 0 to (freqmatrix.nrofrows - 1) do
    freqmatrix.m [j,i] := freqmatrix.m [j,i] * profile.list [i].signal;
 end;
if outputsignalvector then
 begin
  assignfile (outfile,'signalvector.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  sortedprofile := sortprofile;
  for i := 0 to (sortedprofile.length - 1) do
    writeln (outfile, sortedprofile.list [i].signal:12:3);
  closefile (outfile);
 end;
if outputsignalprofile then
 begin
  assignfile (outfile,'signalprofile.txt');
  rewrite (outfile);
  system.settextbuf (outfile, buffer);
  sortedprofile := sortprofile;
  for i := 0 to (sortedprofile.length - 1) do
    writeln (outfile, sortedprofile.list [i].variablename, ' ',
                      sortedprofile.list [i].frequency:12:1, ' ',
                      sortedprofile.list [i].entropy:12:3, ' ',
                      sortedprofile.list [i].signal:12:3);
  closefile (outfile);
 end;
if outputselectedlexicaltypelist then
 begin
  assignfile (outfile,'lexicaltypesbysignal.txt');
  rewrite (outfile);
```

```
  system.settextbuf (outfile, buffer);
  for i := 0 to (freqmatrix.nrofcols -1) do
   writeln (outfile, freqmatrix.collabellist [i]);
  closefile (outfile);
 end;
 matrixform.edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.edit14.text := inttostr (freqmatrix.nrofcols);
 matrixform.Memo1.lines.add ('Dimensionality reduction by signal complete');
end;
        {*** DIMENSIONALITY REDUCTION: PRINCIPAL COMPONENTS ***}

procedure Tmatrixform.RadioButton63Click(Sender: TObject);
{Mean-centre matrix columns}
var
 sum : single;
 mean : single;
 i,j : longint;
begin
 for j := 0 to (freqmatrix.nrofcols - 1) do
  begin
   sum := 0;
   for i := 0 to (freqmatrix.nrofrows - 1) do
    sum := sum + freqmatrix.m [i,j];
   mean := sum / freqmatrix.nrofrows;
   for i := 0 to (freqmatrix.nrofrows - 1) do
    freqmatrix.m [i,j] := freqmatrix.m [i,j] - mean;
  end;
 matrixform.memo1.lines.add ('Matrix mean-centred');
end;

procedure Tmatrixform.RadioButton58Click(Sender: TObject);
{Import eigenvalue matrix}
var
 infile : system.textfile;
 inbuffer: array[1..4096] of char;
 nrofrows : longint;
 nrofcols : longint;
 i,j : longint;
begin
 eigenvaluematrix.nrofrows := strtoint (matrixform.edit21.text);
eigenvaluematrix.nrofcols := eigenvaluematrix.nrofrows;
matrixform.OpenDialog1.execute;
assignfile (infile,matrixform.OpenDialog1.FileName);
reset (infile);
system.settextbuf (infile, inbuffer);
for i := 0 to (eigenvaluematrix.nrofrows - 1) do
 for j := 0 to (eigenvaluematrix.nrofcols - 1) do
  read (infile,eigenvaluematrix.m [i,j]);
closefile (infile);
```

```
 {Reverse column and row order because Matlab sorts eigen matrices in ascending order,
and I want them in descending}
 for i := 0 to (eigenvaluematrix.nrofcols - 1) do
  begin
    for j := 0 to (eigenvaluematrix.nrofrows - 1) do
     tempeigenmatrix.m [j,i] := eigenvaluematrix.m [j,(eigenvaluematrix.nrofcols - 1) - i];
  end;
 tempeigenmatrix.nrofrows := eigenvaluematrix.nrofrows;
 tempeigenmatrix.nrofcols := eigenvaluematrix.nrofcols;
 eigenvaluematrix := tempeigenmatrix;
  for i := 0 to (eigenvaluematrix.nrofrows - 1) do
  begin
   for j := 0 to (eigenvaluematrix.nrofcols - 1) do
    tempeigenmatrix.m [i,j] := eigenvaluematrix.m [(eigenvaluematrix.nrofrows - 1) - i, j];
  end;
 tempeigenmatrix.nrofrows := eigenvaluematrix.nrofrows;
 tempeigenmatrix.nrofcols := eigenvaluematrix.nrofcols;
 eigenvaluematrix := tempeigenmatrix;
 matrixform.memo1.lines.add ('Eigenvalue matrix loaded');
end;

procedure Tmatrixform.RadioButton64Click(Sender: TObject);
{Import eigenvector matrix}
var
 infile : system.textfile;
 inbuffer: array[1..4096] of char;
 nrofrows : longint;
 nrofcols : longint;
 i,j : longint;
begin
 eigenvectormatrix.nrofrows := strtoint (matrixform.edit21.text);
 eigenvectormatrix.nrofcols := eigenvaluematrix.nrofrows;
 matrixform.OpenDialog1.execute;
 assignfile (infile,matrixform.OpenDialog1.FileName);
 reset (infile);
 system.settextbuf (infile, inbuffer);
 for i := 0 to (eigenvectormatrix.nrofrows - 1) do
  for j := 0 to (eigenvectormatrix.nrofcols - 1) do
   read (infile,eigenvectormatrix.m [i,j]);
 {Reverse column order because Matlab sorts eigen matrices in ascending order, and I want
them in descending}
 for i := 0 to (eigenvectormatrix.nrofcols - 1) do
  begin
    for j := 0 to (eigenvectormatrix.nrofrows - 1) do
     tempeigenmatrix.m [j,i] := eigenvectormatrix.m [j,(eigenvectormatrix.nrofcols - 1) - i];
  end;
 tempeigenmatrix.nrofrows := eigenvectormatrix.nrofrows;
 tempeigenmatrix.nrofcols := eigenvectormatrix.nrofcols;
 eigenvectormatrix := tempeigenmatrix;
 closefile (infile);
```

```
 matrixform.memo1.lines.add ('Eigenvector matrix loaded');
end;

procedure Tmatrixform.RadioButton59Click(Sender: TObject);
{Export eigenvalue vector}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, buffer);
 for i := 0 to (eigenvaluematrix.nrofrows - 1) do
  for j := 0 to (eigenvaluematrix.nrofcols - 1) do
   if i = j then
    write (outfile, eigenvaluematrix.m [i,j]:18:6);
 closefile (outfile);
end;

procedure Tmatrixform.RadioButton61Click(Sender: TObject);
{Project frequency matrix into reduced-dikensionality space}
var
 infile : system.textfile;
 buffer: array[1..4096] of char;
 sum : single;
 i,j,k : longint;
begin
 pcareducedmatrix.nrofrows := freqmatrix.nrofrows;
 pcareducedmatrix.nrofcols := strtoint (matrixform.Edit27.Text);
 for i := 0 to (freqmatrix.nrofrows - 1) do
  begin
   for j := 0 to (pcareducedmatrix.nrofcols - 1) do
    begin
     sum := 0;
     for k := 0 to (freqmatrix.nrofcols - 1) do
      begin
       sum := sum + (eigenvectormatrix.m [k,j] * freqmatrix.m [i,k]);
      end;
     pcareducedmatrix.m [i,j] := sum;
    end;
  end;
 pcareducedmatrix.rowlabellist := freqmatrix.rowlabellist;
 freqmatrix := pcareducedmatrix;
 matrixform.edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.edit14.text := inttostr (freqmatrix.nrofcols);
 matrixform.memo1.lines.add ('Done');
end;
                    {*** UTILITIES: SORT ***}
```

```
procedure Tmatrixform.RadioButton26Click(Sender: TObject);
{Sort rows by descending frequency}
var
 highestfrequency : single;
 sum : single;
 index : longint;
 i,j,k : longint;
begin
 tempfreqmatrix.collabellist := freqmatrix.collabellist;
tempfreqmatrix.nrofcols := freqmatrix.nrofcols;
tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
highestfrequency := 0;
for i := 0 to (tempfreqmatrix.nrofrows - 1) do
 begin
  for j := 0 to (freqmatrix.nrofrows - 1) do
   begin
    sum := 0;
    for k := 0 to (freqmatrix.nrofcols - 1) do
     sum := sum + freqmatrix.m [j,k];
    if (sum >= highestfrequency) and (freqmatrix.rowlabellist [j] <> 'xxx') then
     begin
      highestfrequency := sum;
      index := j;
     end;
   end;
  for j := 0 to (tempfreqmatrix.nrofcols - 1) do
   tempfreqmatrix.m [i,j] := freqmatrix.m [index,j];
  tempfreqmatrix.rowlabellist [i] := freqmatrix.rowlabellist [index];
  freqmatrix.rowlabellist [index] := 'xxx';
  highestfrequency := 0;
 end;
freqmatrix := tempfreqmatrix;
matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
matrixform.memo1.lines.add ('Matrix rows sorted by row frequency');
end;

procedure Tmatrixform.RadioButton24Click(Sender: TObject);
{Sort columns by descending frequency}
var
 highestfrequency : single;
 sum : single;
 index : longint;
 i,j,k : longint;
begin
 tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
 tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
 tempfreqmatrix.nrofcols := freqmatrix.nrofcols;
 for i := 0 to (tempfreqmatrix.nrofcols - 1) do
```

```
begin
 highestfrequency := 0;
 for j := 0 to (freqmatrix.nrofcols - 1) do
  begin
   sum := 0;
   for k := 0 to (freqmatrix.nrofrows - 1) do
    sum := sum + freqmatrix.m [k,j];
   if (sum >= highestfrequency) and (freqmatrix.collabellist [j] <> 'xxx') then
    begin
     highestfrequency := sum;
      index := j;
    end;
   end;
  for j := 0 to (tempfreqmatrix.nrofrows - 1) do
   tempfreqmatrix.m [j,i] := freqmatrix.m [j,index];
  tempfreqmatrix.collabellist [i] := freqmatrix.collabellist [index];
  freqmatrix.collabellist [index] := 'xxx';
  highestfrequency := 0;
 end;
freqmatrix := tempfreqmatrix;
matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
matrixform.memo1.lines.add ('Matrix columns sorted by frequency');
end;

procedure Tmatrixform.RadioButton27Click(Sender: TObject);
{Sort matrix rows in order of descending variance}
var
 currentlargestvalue : single;
 currentlargestindex : longint;
 tempvariancelist : Trealvector;
 sum : single;
 mean : single;
 sumofsquareddeviations : single;
 variance : single;
 i,j,k : longint;
begin
 {Make a list of variances, one for each column}
 for i := 0 to (freqmatrix.nrofrows - 1) do
  begin
 {sum a column}
 sum := 0;
 for j := 0 to (freqmatrix.nrofcols - 1) do
  sum := sum + freqmatrix.m [i,j];
 {Mean}
 mean := sum / freqmatrix.nrofcols;
 {Sum squared deviations from mean}
 sumofsquareddeviations := 0;
 for j := 0 to (freqmatrix.nrofcols - 1) do
  sumofsquareddeviations := sumofsquareddeviations +
```

```
                    ((freqmatrix.m [i,j] - mean) *
                     (freqmatrix.m [i,j] - mean));
 {Variance}
 tempvariancelist.v [i] := sumofsquareddeviations / freqmatrix.nrofcols;
end;
for i := 0 to (freqmatrix.nrofrows - 1) do
 tempvariancelist.selected [i] := false;
{Initialize temporary matrix}
tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
tempfreqmatrix.nrofcols := freqmatrix.nrofcols;
tempfreqmatrix.collabellist := freqmatrix.collabellist;
for i := 0 to (tempfreqmatrix.nrofrows - 1) do
 begin
  currentlargestvalue := -1;
  for j := 0 to (freqmatrix.nrofrows - 1) do
   begin
    if (tempvariancelist.v [j] > currentlargestvalue) and (not tempvariancelist.selected [j])
then
     begin
      currentlargestvalue := tempvariancelist.v [j];
      currentlargestindex := j;
     end;
   end;
  tempvariancelist.selected [currentlargestindex] := true;
  for k := 0 to (freqmatrix.nrofcols - 1) do
   tempfreqmatrix.m [i,k] :=  freqmatrix.m [currentlargestindex,k];
  tempfreqmatrix.rowlabellist [i] := freqmatrix.rowlabellist [currentlargestindex];
 end;
 {Finish up}
 freqmatrix := tempfreqmatrix;
 matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
 matrixform.memo1.lines.add ('Matrix rows sorted by variance');
end;

procedure Tmatrixform.RadioButton25Click(Sender: TObject);
{Sort matrix by descending variance}
var
 currentlargestvalue : single;
 currentlargestindex : longint;
 tempvariancelist : Trealvector;
 sum : single;
 mean : single;
 sumofsquareddeviations : single;
 variance : single;
 i,j,k : longint;
begin
 {Make a list of variances, one for each column}
 for i := 0 to (freqmatrix.nrofcols - 1) do
  begin
```

```
{sum a column}
sum := 0;
for j := 0 to (freqmatrix.nrofrows - 1) do
 sum := sum + freqmatrix.m [j,i];
{Mean}
mean := sum / freqmatrix.nrofrows;
{Sum squared deviations from mean}
sumofsquareddeviations := 0;
for j := 0 to (freqmatrix.nrofrows - 1) do
 sumofsquareddeviations := sumofsquareddeviations +
                           ((freqmatrix.m [j,i] - mean) *
                            (freqmatrix.m [j,i] - mean));
 {Variance}
 tempvariancelist.v [i] := sumofsquareddeviations / freqmatrix.nrofrows;
end;
for i := 0 to (freqmatrix.nrofcols - 1) do
 tempvariancelist.selected [i] := false;
{Initialize temporary matrix}
tempfreqmatrix.nrofrows := freqmatrix.nrofrows;
tempfreqmatrix.nrofcols := freqmatrix.nrofcols;
tempfreqmatrix.rowlabellist := freqmatrix.rowlabellist;
for i := 0 to (tempfreqmatrix.nrofcols - 1) do
 begin
  currentlargestvalue := -1;
  for j := 0 to (freqmatrix.nrofcols - 1) do
   begin
    if (tempvariancelist.v[j] > currentlargestvalue) and (not tempvariancelist.selected [j])
then
     begin
      currentlargestvalue := tempvariancelist.v [j];
      currentlargestindex := j;
     end;
   end;
  tempvariancelist.selected [currentlargestindex] := true;
  for k := 0 to (freqmatrix.nrofrows - 1) do
   tempfreqmatrix.m [k,i] :=  freqmatrix.m [k,currentlargestindex];
  tempfreqmatrix.collabellist [i] := freqmatrix.collabellist [currentlargestindex];
 end;
{Finish up}
freqmatrix := tempfreqmatrix;
 matrixform.Edit13.Text := inttostr (freqmatrix.nrofrows);
 matrixform.Edit14.Text := inttostr (freqmatrix.nrofcols);
 matrixform.memo1.lines.add ('Matrix columns sorted by variance');
end;
          {*** UTILITIES: COVARIANCE / CORRELATION ***}

procedure Tmatrixform.RadioButton5Click(Sender: TObject);
{Calculate covariance matrix}
var
 sum : single;
```

```
 meani, meanj : single;
 i,j,k : longint;
begin
 for i := 0 to (freqmatrix.nrofcols - 1)do
  begin
   sum := 0;
   for k := 0 to (freqmatrix.nrofrows - 1) do
    sum := sum + freqmatrix.m [k,i];
   meani := sum / freqmatrix.nrofrows;
   for j := 0 to (freqmatrix.nrofcols - 1) do
   begin
    sum := 0;
    for k := 0 to (freqmatrix.nrofrows - 1) do
     sum := sum + freqmatrix.m [k,j];
    meanj := sum / freqmatrix.nrofrows;
    sum := 0;
    for k := 0 to (freqmatrix.nrofrows - 1) do
     sum := sum + ((meani - freqmatrix.m [k,i])) * (meanj - freqmatrix.m [k,j]);
    covariancematrix.m [i,j] := sum;
   end;
  end;
 covariancematrix.axislength := freqmatrix.nrofcols;
 matrixform.memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton29Click(Sender: TObject);
{Save covariance matrix}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, buffer);
 for i := 0 to (covariancematrix.axislength - 1) do
  begin
   for j := 0 to (covariancematrix.axislength - 1) do
    write (outfile, covariancematrix.m [i,j]:12:6);
   writeln (outfile);
  end;
 closefile (outfile);
 matrixform.memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton33Click(Sender: TObject);
{Output covariances sorted descending}
var
 i,j : longint;
```

```
begin
 {Construct sort tree}
 rootnode := nil;
 for i := 0 to (freqmatrix.nrofrows - 1) do
  for j := 0 to i do
   if (i <> j) and (abs (freqmatrix.m [i,j]) > 0.29) then
    begin
     makenode (newnode, freqmatrix.m [i,j]);
     insertnode (rootnode, newnode);
    end;
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 {Output sort tree}
 inorder (rootnode);
 closefile (outfile);
 matrixform.Memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton36Click(Sender: TObject);
{Output diagonal of covariance matrix}
var
 i, j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 for i := 0 to (covariancematrix.axislength - 1) do
  for j := 0 to (covariancematrix.axislength - 1) do
   if i = j then
    writeln (outfile, covariancematrix.m [i,j]:12:6);
 closefile (outfile);
 matrixform.Memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton12Click(Sender: TObject);
{Calculate correlation matrix}
var
 sum : single;
 meani, meanj : single;
 variancei : single;
 variancej : single;
 stddevi : single;
 stddevj : single;
 correlation : single;
 covariance : single;
 i,j,k : longint;
begin
```

```
for i := 0 to (freqmatrix.nrofcols - 1)do
 begin
  sum := 0;
  for k := 0 to (freqmatrix.nrofrows - 1) do
   sum := sum + freqmatrix.m [k,i];
  meani := sum / freqmatrix.nrofrows;
  variancei := 0;
  for k := 0 to (freqmatrix.nrofrows - 1) do
   variancei := variancei + ((meani - freqmatrix.m [k,i]) * (meani - freqmatrix.m [k,i]));
  stddevi := sqrt (variancei);
  for j := 0 to (freqmatrix.nrofcols - 1) do
 begin
  sum := 0;
  for k := 0 to (freqmatrix.nrofrows - 1) do
   sum := sum + freqmatrix.m [k,j];
  meanj := sum / freqmatrix.nrofrows;
  variancej := 0;
  for k := 0 to (freqmatrix.nrofrows - 1) do
   variancej := variancej + ((meanj - freqmatrix.m [k,j]) * (meanj - freqmatrix.m [k,j]));
  stddevj := sqrt (variancej);
  sum := 0;
  for k := 0 to (freqmatrix.nrofrows - 1) do
   sum := sum + ((meani - freqmatrix.m [k,i])) * (meanj - freqmatrix.m [k,j]);
  covariance := sum;
  correlation := covariance / (stddevi * stddevj);
    covariancematrix.m [i,j] := correlation;
   end;
  end;
 covariancematrix.axislength := freqmatrix.nrofcols;
 matrixform.memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton31Click(Sender: TObject);
{Output correlation matrix}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
matrixform.savedialog1.execute;
assignfile (outfile,matrixform.savedialog1.filename);
rewrite (outfile);
system.settextbuf (outfile, buffer);
for i := 0 to (covariancematrix.axislength - 1) do
 begin
  for j := 0 to (covariancematrix.axislength - 1) do
   write (outfile, covariancematrix.m [i,j]:12:6);
  writeln (outfile);
 end;
closefile (outfile);
```

```
 matrixform.memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton34Click(Sender: TObject);
{Output correlations sorted descending}
var
 i,j : longint;
begin
 {Construct sort tree}
 rootnode := nil;
for i := 0 to (freqmatrix.nrofrows - 1) do
 for j := 0 to i do
  if (i <> j) and (abs (freqmatrix.m [i,j]) > 0.29) then
   begin
    makenode (newnode, freqmatrix.m [i,j]);
    insertnode (rootnode, newnode);
   end;
matrixform.savedialog1.execute;
assignfile (outfile,matrixform.savedialog1.filename);
rewrite (outfile);
system.settextbuf (outfile, outbuffer);
{Output tree}
inorder (rootnode);
closefile (outfile);
matrixform.Memo1.lines.add ('Done');
end;

procedure Tmatrixform.RadioButton35Click(Sender: TObject);
{Output diagonal of correlation matrix}
var
 i, j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, outbuffer);
 for i := 0 to (covariancematrix.axislength - 1) do
  for j := 0 to (covariancematrix.axislength - 1) do
   if i = j then
    writeln (outfile, covariancematrix.m [i,j]:12:6);
 closefile (outfile);
 matrixform.Memo1.lines.add ('Done');
end;

{*** OUTPUT MATRIX ***}

procedure Tmatrixform.RadioButton16Click(Sender: TObject);
{Boolean to output row/column header}
begin
 saveheader := true;
```

```
end;

procedure Tmatrixform.RadioButton14Click(Sender: TObject);
{Output matrix only}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
 matrixform.savedialog1.execute;
assignfile (outfile,matrixform.savedialog1.filename);
rewrite (outfile);
system.settextbuf (outfile, buffer);
if saveheader then
 begin
  writeln (outfile, freqmatrix.nrofrows);
  writeln (outfile, freqmatrix.nrofcols);
 end;
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j]:16:8, ' ');
  writeln (outfile);
 end;
closefile (outfile);
 matrixform.memo1.lines.Add('Matrix only saved');
end;

procedure Tmatrixform.RadioButton17Click(Sender: TObject);
{Output matrix with row labels only}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, buffer);
 if saveheader then
 begin
  writeln (outfile, freqmatrix.nrofrows);
  writeln (outfile, freqmatrix.nrofcols);
 end;
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
 j := 0;
 while (freqmatrix.rowlabellist [i,j] <> '.') and (freqmatrix.rowlabellist [i,j] <> ' ') do
  begin
   write (outfile, freqmatrix.rowlabellist [i,j]);
```

```
    j := j + 1;
   end;
  write (outfile, ' ');
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j]:16:8, ' ');
  writeln (outfile);
 end;
closefile (outfile);
matrixform.memo1.lines.Add('Matrix + row labels only saved');
end;

procedure Tmatrixform.RadioButton18Click(Sender: TObject);
{Output matrix with column labels only}
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, buffer);
 if saveheader then
  begin
   writeln (outfile, freqmatrix.nrofrows);
   writeln (outfile, freqmatrix.nrofcols);
  end;
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
 j := 0;
 while freqmatrix.collabellist [i,j] <> ' ' do
  begin
   write (outfile, freqmatrix.collabellist [i,j]);
   j := j + 1;
  end;
 write (outfile, ' ');
 end;
writeln (outfile);
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j]:16:8, ' ');
  writeln (outfile);
 end;
closefile (outfile);
matrixform.memo1.lines.Add('Matrix + cols only saved');
end;

procedure Tmatrixform.RadioButton19Click(Sender: TObject);
{Output matrix with both row and column labels}
```

```
var
 outfile : system.textfile;
 buffer: array[1..4096] of char;
 i,j : longint;
begin
 matrixform.savedialog1.execute;
 assignfile (outfile,matrixform.savedialog1.filename);
 rewrite (outfile);
 system.settextbuf (outfile, buffer);
 if saveheader then
 begin
  writeln (outfile, freqmatrix.nrofrows);
  writeln (outfile, freqmatrix.nrofcols);
 end;
for i := 0 to (freqmatrix.nrofcols - 1) do
 begin
  j := 0;
  while freqmatrix.collabellist [i,j] <> ' ' do
   begin
    write (outfile, freqmatrix.collabellist [i,j]);
    j := j + 1;
   end;
  write (outfile, ' ');
 end;
writeln (outfile);
for i := 0 to (freqmatrix.nrofrows - 1) do
 begin
  j := 0;
  while (freqmatrix.rowlabellist [i,j] <> '.') and (freqmatrix.rowlabellist [i,j] <> ' ') do
   begin
    write (outfile, freqmatrix.rowlabellist [i,j]);
    j := j + 1;
   end;
  write (outfile, ' ');
  for j := 0 to (freqmatrix.nrofcols - 1) do
   write (outfile, freqmatrix.m [i,j]:16:8, ' ');
  writeln (outfile);
 end;
closefile (outfile);
matrixform.memo1.lines.Add ('Matrix + row and col labels saved');
end;

end.
```

Appendix 2: List of function words

As noted in the foregoing discussion, function words such as preopositions and determiners were removed from the text of the Qur'an before extraction of the frequency data. Because of the complexity of Arabic morphology, it was easier to compile a list of function words manually than to attempt to define an algorithm for the task. Some were taken from a concordance of the Qur'anic lexicon compiled by Abd Al-Baqi (1987), and the rest were specified by the present author on the basis of her knowledge of Arabic. There is a certain amount of subjectivity about what should count as a function word; the list used for present purposes is given below.

aaaindahu	aaala	aaalayha	aaalayhi	aaalayhim
aaalayhima	aaalayhimu	aaalayhinna	aaalayhu	aaalayka
aaalayki	aaalaykum	aaalaykuma	aaalaykumu	aaalayna
aaallakum	aaan	aaanha	aaanhu	aaanhum
aaanhuma	aaanhumu	aaani	aaanka	aaankum
aaankumu	aaannee	aaayn-seen-qaf	aaayunuhum	aainda
aaindaha	aaindahu	aaindahum	aaindahumu	aaindaka
aaindakum	aaindana	aaindee	aaindi	aaindihi
aaindika	aaindina	aanta	aantum	afaanta
afaghayra	afahum	afahumu	afa-in	afakullama
afala	afalam	afama	afaman	afamin
aghayra	ahaola-i	ahatha	ahum	a-in
a-innaka	a-innakum	a-itha	ala	alastu
alaysa	alif-lam-meem	alif-lam-meem-ra	alif-lam-meem-sad	alla
allatee	allathee	allatheena	am	ama
amamahu	ami	amma	amman	an
ana	ani	anna	annaba	annaha
annahu	annahum	annahuma	annahumu	annahunna
annakum	annama	annee	anta	antum
antumu	athalika	aúna	aw	awaman
ayllu	ayna	ayya	ayyakumu	ayyi
ayyin	ayyuha	ayyuhum	ayyukum	baaada
baaadaha	baaadahu	baaadahum	baaadahunna	baaadakum
baaadan	baaadi	baaadiha	baaadihi	baaadihim
baaadika	baaadikum	baaadu	baaaduha	baaaduhum
baaadukum	baaaduna	baaaeedun	bal	bayna
baynaha	baynahu	baynahum	baynahuma	baynahumu
baynahunna	baynaka	baynakum	baynakumu	baynana
bayni	baynihim	baynihima	baynika	baynikum

baynina	bayyannahu	bee	bi	bi-an
bi-ann	bi-annahum	bi-ayyi	biha	bihi
bihim	bihima	bihimu	bika	bikum
bikumu	bima	biman	bina	bithati
biya	don	doona	doonee	dooniha
doonihim	doonihima	doonihimu	doonikum	doonina
faama	faamma	faanna	faayna	fabima
fahu	fahum	fahuwa	fa-imma	fa-in
fa-ini	fa-inna	fa-innahum	fa-innama	fa-itha
falahu	falahum	falakum	falam	falamma
falan	falawla	falima	fama	faman
famani	famin	famina	faqad	faqadi
fathamma	fathanika	fawqa	fawqaha	fawqahum
fawqahumu	fawqakum	fawqakumu	fawqi	fawqihi
fawqihim	fawqihinna	fawqikum	fee	feeha
feehi	feehim	feehima	feehinna	feekum
feema	feena	ghayra	ghayraha	ghayrakum
ghayree	ghayri	ghayrihi	ghayrikum	ghayru
ha-meem	haola-i	hatha	hathani	hathihi
hatta	hawlakum	hawlihim	haythu	him
himu	hinna	hiya	hiyah	hum
huma	humu	hunalika	hunna	ila
ilayha	ilayhi	ilayhim	ilayhimu	ilayhinna
ilayka	ilayki	ilaykum	ilaykuma	ilaykumu
ilayna	ilayya	illa	imam	in
inahu	inee	ini	inna	inna
innaha	innahu	innahum	innahuma	innahumu
innahunna	innaka	innaki	innakum	innama
innana	innanee	inne	innee	ith
itha	ithan	ithi	iyyahu	iyyahum
iyyaka	iyyakum	iyyaya	kaallathee	kaanna
kaannaha	kaannahum	kaannama	kaayyin	kaf-ha-ya-aaayn-sad
kalla	kathalika	kathaliki	kay	kayla
khalfihi	khilalakum	khilalihi	kilahuma	kilta
kulla	kullaha	kullahu	kullama	kullan
kulli	kulliha	kullihi	kullin	kullu
kulluhu	kulluhum	kulluhunna	kullun	kuntum
laaaala	laaaalla	laaaallahu	laaaallahum	laaaallaka
laaaallakum	laaaallee	ladayhi	ladayhim	ladayna
ladayya	ladun	ladunhu	ladunka	ladunna
lafee	laha	lahu	lahum	lahuma
lahumu	lahunna	la-in	la-ini	laka
laki	lakin	lakini	lakinna	lakinnahu
lakinnahum	lakinnakum	lakinnee	lakum	lakuma
lakumu	lakunna	lam	lama	laman
lamin	lamina	lamma	lan	lana
laqad	laqadi	lasta	lastu	lastum

lastunna	law	lawi	lawla	laysa
laytanee	lee	li-alla	likay	lima
liman	limani	lithalika	liya	lnnaha
ma	maaaa	maaaaha	maaaahu	maaaahum
maaaahumu	maaaaka	maaaakum	maaaakuma	maaaana
maaaiya	mahma	man	mani	mata
matha	miman	mimma	mimmani	min
mina	minha	minhu	minhum	minhuma
minhumu	minhunna	minka	minkum	minkunna
minna	minnee	nahnu	nahum	noon
ola-i	ola-ika	ola-ikum	qabla	qablahu
qablahum	qablee	qabli	qabliha	qablihim
qablihimu	qablika	qablikum	qablina	qablu
qad	qadi	qaf	sad	sawa-i
sawa-in	sawaon	sawfa	ta-seen	ta-seem-meem
tahta	tahti	tahtihim	tha	thalika
thaliki	thalikum	thalikumu	thalikunna	thamma
thati	thatu	thee	thoo	thumma
tilka	tilkuma	tilkumu	wa	waamma
waana	waani	waanna	waannahu	wabi
wabima	wafee	wahuwa	wa-ila	wa-imma
wa-in	wa-inna	wa-innama	wa-ith	wa-itha
wakam	wala	walahu	walahum	wala-in
walakin	walamma	walan	walaw	walawi
wama	waman	wamani	wamimma	wamin
wamina	waminhum	waminna	waqad	waraahu
waraakum	wara-i	wara-ihi	wara-ihim	wara-ikum
waykaanna	waykaannahu	yashao		