# Named Entity Extraction for Speech

James Horlock

Thesis submitted for the degree of Doctor of Philosophy

University of Edinburgh

September 2003

## Dedication

To Alex,

Friend, best friend, girlfriend, fiancée and wife, you've been a support to me from the start of my thesis to its completion. Thank you for your encouragement, love and patience.

## Acknowledgements

Thank you to everyone at CSTR for all the help throughout the years, for your technical support as well as all the educational pointers throughout my stay. Thank you for making me welcome.

In particular, a huge thank you goes to my primary supervisor Simon King, who after being a great colleague for my first year of study proved to be an even better supervisor. He has been an encouragement from the word go, and has maintained confidence in my work - when I would have given up. I would highly recommend Simon as a supervisor to anyone foolish enough to undertake a PhD.

Thanks also go to Marc Moens for his support as a secondary supervisor, and to Paul Taylor my primary supervisor for my first year of study for their help and advice over the years.

I would like to thank everyone in CSTR, as I'm confident that I have disturbed each of you at some point in order to get help and advice; none more so than Robert Clark. I really don't know that I could have achieved this thesis without what seemed to be constant technical support from the computer guru Rob. Thank you, Rob, for not losing patience - or for hiding it so well if you did! Thanks also to Korin, Joe, Jithendra and Yoshinori who I've troubled on too numerous occasions to count.

Thank you to Claire Grover, who has supplied an otherwise penniless student with funds sufficient to complete this thesis and make his mortgage payments. Thank you also to the EPSRC for their funding for the first three years of the thesis.

Thank you to Phil Woodland of Cambridge University Engineering Department for the use of the HTK word lattices used throughout this thesis. Thanks also go the computing support team at Buccleuch Place.

Thank you to my parents for their continual support over the years, and probably the wisest advice and the hugest favour I got during my PhD - 'Buy a flat in Edinburgh, and we'll act as guarantor'. Thank you in particular for proof reading this thesis.

Finally, thank you to my wonderful wife Alex. Thank you for being my best friend, for marrying me during these years - and for always being there, even when I got really unbearable! Happy anniversary!

## Declaration

I declare that, apart from where properly indicated, the work contained in this thesis is entirely the product of my own work.

## Abstract

Named entity extraction is a field that has generated much interest over recent years with the explosion of the World Wide Web and the necessity for accurate information retrieval. Named entity extraction, the task of finding specific entities within documents, has proven of great benefit for numerous information extraction and information retrieval tasks.

As well as multiple language evaluations, named entity extraction has been investigated on a variety of media forms with varying success. In general, these media forms have all been based upon standard text and assumed that any variation from standard text constitutes noise.

We investigate how it is possible to find named entities in speech data. Where others have focussed on applying named entity extraction techniques to transcriptions of speech, we investigate a method for finding the named entities direct from the word lattices associated with the speech signal. The results show that it is possible to improve named entity recognition at the expense of word error rate (WER) in contrast to the general view that F-score is directly proportional to WER.

We use a Hidden Markov Model (HMM) style approach to the task of named entity extraction and show how it is possible to utilise a HMM to find named entities within speech lattices. We further investigate how it is possible to improve results by considering an alternative derivation of the joint probability of words and entities than is traditionally used. This new derivation is particularly appropriate to speech lattices as no presumptions are made about the sequence of words.

The HMM style approach that we use requires using a number of language models in parallel. We have developed a system for discriminately retraining these language models based upon the results of the output, and we show how it is possible to improve named entity recognition by iterations over both training data and development data.

We also consider how part-of-speech (POS) can be used within word lattices. We devise a method of labelling a word lattice with POS tags and adapt the model to make use of these POS tags when producing the best path through the lattice. The resulting path provides the most likely sequence of words, entities and POS tags and we show how this new path is better than the previous path which ignored the POS tags.

# Contents

# List of Figures

# List of Tables

## 0.1 Notation

Throughout this thesis the notation adopted is that set out in table 1.

| | |
|---|---|
| $L$ | The length of the word sequence |
| $w$ | A word |
| $w_i$ | The specific word at position $i$ in the word sequence |
| $W_i^j$ | The word sequence from $w_i$ through $w_j$ |
| $W$ | The word sequence $W_1^L$ |
| $e$ | An entity |
| $e_i$ | The entity corresponding to the word $w_i$ |
| $E_i^j$ | The entity sequence from $e_i$ through $e_j$ |
| $E$ | The entity sequence $E_1^L$ |
| $t$ | A part of speech tag |
| $t_i$ | The part of speech tag corresponding to the word $w_i$ |
| $T_i^j$ | The part of speech tag sequence from $t_i$ through $t_j$ |
| $T$ | The part of speech tag sequence $T_1^L$ |

Table 1: *The notation used throughout this thesis.*

The main focus of this thesis is named entity extraction from American broadcast news speech word lattices. For this reason when referring to words within the lattices, they will be referred to as they are within the lattice. Therefore words which would normally start with a capital letter in written text will not, for example "january", "peter" and "london"; similarly, words will be spelled in American English, for example "labor", "savor", and "savior".

Throughout the thesis we refer to seven types of named entities, an example of each is given in table 2. In some places in the thesis these will be abbreviated to those in table 3, to aid clarity. When eight types of named entities are referred to, this is to include a not-an-entity named entity which carries no markup.

| Words | Markup |
|---|---|
| James Horlock | <ENAMEX TYPE="PERSON">James Horlock</ENAMEX> |
| Edinburgh | <ENAMEX TYPE="LOCATION">Edinburgh</ENAMEX> |
| Edinburgh University | <ENAMEX TYPE="ORGANIZATION">Edinburgh University</ENAMEX> |
| Three thirty | <TIMEX TYPE="TIME">Three thirty</TIMEX> |
| Thirtieth June | <TIMEX TYPE="DATE">Thirtieth June</TIMEX> |
| Five percent | <NUMEX TYPE="PERCENTAGE">Five percent</NUMEX> |
| Five million dollars | <NUMEX TYPE="MONEY">Five million dollars</NUMEX> |

Table 2: *Examples of named entity markup.*

| Words | Abbreviated markup |
|---|---|
| James Horlock | <PERSON>James Horlock</PERSON> |
| Edinburgh | <PLACE>Edinburgh</PLACE> |
| Edinburgh University | <ORG>Edinburgh University</ORG> |
| Three thirty | <TIME>Three thirty</TIME> |
| Thirtieth June | <DATE>Thirtieth June</DATE> |
| Five percent | <PERCENT>Five percent</PERCENT> |
| Five million dollars | <MONEY>Five million dollars</MONEY> |

Table 3: *Abbreviated examples of named entity markup.*

# Chapter 1

# Introduction

## 1.1 Preamble

Over the past decade information extraction has been a particularly hot issue. Since the formation of the World Wide Web in 1989, its growth has been phenomenal. Terabytes of information have become available; the problem which remains is how to index and search this rapidly extending information source to obtain required information. Initially, the World Wide Web primarily contained text information, but with its growth it has evolved to be a multimedia source, containing not just text but pictures, sound, video and more. The resulting web is vast and uncharted and extracting desired information is non-trivial.

The World Wide Web is not the only source of vast amounts of data. According to (*National Association of Broadcasters Information Resource Center* 2000) there were over 44,000 radio stations broadcasting in the year 2000 and that number is on the increase. The information contained in these broadcasts, for the most part, is lost immediately after broadcast.

The task of information extraction is to extract information from a source that is relevant to some specific need. It is based on the assumption that the source contains both information and noise, and that the noise may be regarded as redundant.

Information extraction is often perceived as a means rather than an end, and is used in other tasks such as information retrieval. The distinction between these tasks is often unclear and hazy. Theoretically, information extraction is about locating the information, whilst information retrieval is about returning the information that has been found. Information extraction is about finding and labelling information (thus effectively adding to the size of the source), whilst information retrieval is about returning a subset of the source (and is therefore smaller than the source). Named entity extraction is the information extraction task of finding and labelling specific entities (such as monetary expressions) within a source.

## 1.2 Formulation of the problem

Although much work has gone into the task of information extraction, and indeed into named entity extraction, this work has primarily been focused on electronic text data. Unsurprisingly, named entity extraction from hypertext markup language (HTML) documents has received the greatest attention because HTML constitutes the primary format of documents on the World Wide Web. Some work, however, has been directed to other formats of information - such as optical character recognition (OCR) (Cheung, Pang, Lyu, Ng & King 2000), speech (Kim & Woodland 2000) and video (Wactlar 1999).

The main concern of this thesis is how to extract named entities from speech data. For the purpose of this thesis we have investigated how to find standard named entities, although the principles apply to non-standard named entities. The standard named entities are people's names, place names, organizations, times, dates, monetary expressions and percentages. These fit into three broad categories: ENAMEX - referring to names (people's names, place names, organizations), NUMEX - referring to numerical expressions (monetary expressions, percentages) and TIMEX - referring to temporal expressions (times, dates). In this thesis we focus on finding these within speech data, and marking up the speech data to reflect the discoveries.

Since speech data is essentially a digital representation of speech waves, it is difficult to find and mark up named entities within this data source. Previously, all named entity extraction from speech has been conducted on transcripts, either manual or automatically recognised. Standard generalised markup language (SGML), or, more latterly, extensible markup language (XML) markup, has been added to the transcript to indicate the locations of named entities.

From the point of view of text, transcriptions of speech are generally noisy. This noise is partly due to the original source and partly due to the transcription of the source. These two distinct causes of noise are illustrated in the following examples: "Peter Piper picked a peck of pickled pepper" and "Let us hear the prayer he taught us". The first is a difficult thing to say (thus noise occurs in the production of the speech); the second is potentially a difficult thing to hear (the noise occurs during the transcription of the speech) - where children are reported to have thought the vicar said "Let us hear the prairie tortoise" (and could even have mistaken the phrase for "Lettuce here - the prairie tortoise"). There may also be other problems such as background noise in the recording, badly positioned microphones, or non-recorded visual stimulii that may add to the confusion. Many of these phenomena - disfluencies, coarticulations, etc have been investigated separately but from the point of view of trying to generate standard text, they are all essentially noise.

A typical transcription of speech is shown in figure 1.1, whereas the substance of the conversation is shown in figure 1.2. It is possible that person 1 in the conversation did not articulate all of the words shown in figure 1.2, making the automatic transcription more difficult.

SHOE ME I YOU HAVE UH TIME SORRY
TIME YEAH IS FOUR THIRTY SEVEN

Figure 1.1: *Corrupt transcript of a conversation concerning the current time.*

Person 1: Excuse me, do you have the time?
Person 2: Sorry?
Person 1: The time?
Person 2: Yeah. it's 4:37

Figure 1.2: *Explanation of the conversation concerning the current time.*

The transcription, figure 1.1, is almost incomprehensible without the realisation that it is a transcription of speech. By reading the corrupt transcription aloud, however, the mind is possibly able to convert back to the original conversation.

A number of things are apparent from the transcription which are important to note about general transcriptions of speech.

- The text is case insensitive
- There is a noticable lack of punctuation
- There are no indications of speaker changes
- People often use a different vocabulary when speaking than when writing e.g. use of "Yeah" (McCarthy 1999)
- Formatting changes; e.g. 4:37 becomes four thirty seven, just as $1,500,000 would become one point five million dollars
- Gramatical structures are not as carefully adhered to. Sentences may contain repeated words or part words, and may include corrections; e.g. "Pete Peter Pan", "Yeah I mean no".

Ji-Hwan Kim, in his thesis (Kim 2001), addresses some of the issues raised in the above list, and shows innovative methods for finding and adding some of these missing features back into the transcription. He also shows that by adding these features to the transcriptions it is possible to improve named entity recognition.

In this thesis the primary focus is on the use of speech word lattices, as described in chapter 3. We will use speech transcripts to provide comparisons with our results based on word lattices. The fundamental hypothesis of this thesis is that it is possible to obtain better named entity extraction from word lattices than from transcriptions.

## 1.3 Scope of this thesis

The objective of this thesis is show that it is possible to perform named entity extraction on word lattices as well as on transcriptions and to present a comparison of the corresponding results. In the thesis we will introduce the standard way that statistical systems evaluate named entity sequences on text data. Using this methodology as a base we will devolop a system that can extract a named entity sequence together with a word sequence from a word lattice. We will show that it is possible to get better results from the word lattice than are achieved by finding the 1-best transcription of the word lattice and using named entity extraction on that transcription.

Having established a method for named entity extraction from word lattices we will investigate methods that can be applied to the lattices to improve F-score results. The first method we investigate is to reconsider the definition of joint probability that is traditionally used in these problems and show that there is a better method - "better" in that it requires fewer independence assumptions. Building on this, we produce a similar model, which makes fewer independence assumptions, and show how this new model yields better F-scores. We also investigate how POS can be used to improve the F-score in lattices that do not contain POS information. We provide a method of POS tagging the lattices and show how these new lattices can be used to provide better results than the originals. We describe a method for discriminately training language models in such a way as to discriminate against words which occur in different language models, in an attempt to show that, although the presence of a word in training data generally suggests that it is more likely to recur than a word which does not occur in the training data, this is not always the case. If common words are found once or twice in a particular data, we regard them as less likely to be correct than less common words. In all cases we use the same underlying model architecture and show how we are able to gain better F-scores with these alterations to the model.

## 1.4 Organization of this thesis

The thesis consists of eight chapters: chapter 2 introduces previous work in the subject area. Chapter 3 introduces the data that we use throughout the thesis for our evaluations. Chapter 4 offers a detailed description of a standard statistical model that has been used for the task of named entity extraction from speech transcripts, together with differences between the systems that have used this standard model. In chapter 5 we describe our extensions to the standard model, such as explaining the adaptation to use speech word lattices instead of text and an alternative to the traditional mathematical breakdown of the problem. In chapter 6 we introduce the concept of using discriminative training of the language models used by the system to improve named entity extraction results. In chapter 7 we show how it is possible to label word lattices with part of speech tags, and use these to improve the named entity F-scores. Finally, we conclude the thesis, with suggesting further possible work which could improve upon our findings, in chapter 8.

# Chapter 2

# Literature Review

We begin with a definition of a named entity. A named entity is any word or phrase that corresponds to an item in exactly one predetermined category (named entity class). The categories can comprise any collection of items; examples include people's names, computer processors, job titles, proteins, and so on.

Although any new category can become a named entity class, over time standards have been formed to aid comparison of results in research. Although the standards are evolving and more standards are emerging, the fundamental standard that has generally been adopted is to use ENAMEX, NUMEX and TIMEX; covering respectively names, numbers and temporal expressions.

We now provide a brief history of named entity extraction together with some of the uses of named entity extraction and, in particular, references to named entities within speech. We also briefly comment on part-of -speech (POS) tagging because in chapter 7 we show how POS information can be used to enhance named entity extraction from speech data.

## 2.1 A brief history of named entity extraction

The task of named entity recognition was first given a formal introduction in the Message Understanding Conferences (MUC) (Chinchor 1997). Since that time it has grown to become a well formulated and understood task.

### 2.1.1 Three approaches to extracting named entities

There are three basic approaches to named entity extraction: Rule-based, Stochastic and Hybrid systems. Here we detail examples of all three types of named entity extraction from text systems. In section 2.3 we return to the differing types of systems and focus on named entity extraction from speech data. We give details of Rule-based systems for named entity extraction from speech in section 2.3.1 and stochastic systems for speech in section 2.3.2. There have not to date been any Hybrid systems used for information extraction from speech.

#### Rule-based

An example of a Rule-based system that was used for named entity extraction is described in (Farmakiotou, Karkaletsis, Koutsias, Sigletos, Spyropoulos & Stamatopoulos 2000). This system is designed to find named entities within Greek financial texts. The system is broadly divided into three main components: Linguistic Preprocessing, Named Entity Identification, and Named Entity Classification. As these names suggest, this system classifies the named entities after finding them, rather than simultaneously as most statistical systems do.

Linguistic Preprocessing essentially involves tokenisation of the input text stream, sentence splitting, part of speech (POS) tagging followed by stemming, and gazetteer lookup. The idea behind stemming after POS tagging is that the word is essentially the combination of the POS and the stem. If a word is uncommon then its information has effectively been split between the POS and the stem so it may be compared with other words which have either the same POS or the same stem. The gazetteer lookup allows well-known named entities to be established at this early stage in the process.

Named Entity Identification involves detection of the boundaries, and is subdivided into three further elements: initial delimitation, separation and exclusion. The intial delimitation uses very general patterns to find likely named entities. This delimitation process may well result in some distinct named entities being grouped into single entities and even the classification of non-entities as entities. The separation stage deals with breaking the incorrectly grouped entities. Greek, like English, suffers from the difficulty of dealing with 'and' within entities (for example: "Marks and Spencer" is one entity whilst "Microsoft and AOL" is two). The exclusion stage excludes known pitfalls by making use of a "killer" list, effectively a gazetteer of non-entity words.

Named Entity Classification again is subdivided into three elements; namely, application of rules, gazetteer-based classification, and partial matching. The rules take into account both internal and external evidence, such as a company designator - e.g. Ltd - or a preceeding title - e.g. "Mr" respectively. The gazetteer-based classification is carried out in preferencial order; that is, organisations are classified first as they can include people or place names. Finally the partial matching works by looking for entities which are similar to those already classified.

### Stochastic

An example of a stochastic system is the maximum entropy (Max Ent) named entity system, appropriately named MENE, developed at New York University (Borthwick 1999). This system, which was built from a combination of knowledge sources, initially contained no handcrafted rules; however in later experiments such rules were used to improve the results. The system visualises the named entity task as associating a tag with each word. For the standard named entities this corresponds to associating one of twenty nine possible tags (in general $4.\#entities + 1$ tags); entity_start, entity_continue, entity_end, entity_unique or not-an-entity. These twenty nine tags form the space of "futures" for the Max Ent formulation.

MENE incorporates solely binary-valued features from a number of sources. These sources can be classified as binary features, lexical features, section features, dictionary features and external systems features. Whereas all of the features have binary output, the "binary" features are features which are binary without modification; for example, "the previous word starts with a capital letter". The lexical features are where most of the power of this method comes from. By considering the current word $w_0$, and looking at the context $w_{-2}...w_2$, lexical features can be formed; for example, "if preceeding word is 'Mr' and the tag is person_start". Section features make predictions based on the particular section of the article; for example, "is this word within the preamble to the text". The dictionary features correspond to one of five tags for each of the eight dictionaries used; namely start, continue, end, unique or not-in-dictionary. Finally, the external systems features made use of the output of other named entity systems. In particular they made use of systems from the University of Manitoba (Lin 1998) - a finite state pattern matcher which gives special attention to collocations (words in context), and IsoQuest (Krupka & Hausman 1998) - a commercial Rule-based system capable of an F-score of 81.96 on single case MUC 7 data. An example of one such feature would be "if IsoQuest tagged this word as person_start and the tag is person_start".

The features which were selected were chosen by simply creating a pool of features and then selecting all features which fired at least three times on the training data. Having selected the features, it was simply a matter of training the weights of a Max Ent model. Having trained, decoding involved running the Max Ent model on the new text.

### Hybrid

An example of a hybrid system is that used by Edinburgh University's Language Technology Group to win the competition at the 7th Message Understanding Conference (Mikheev, Moens & Grover 1998). This system consisted of a pipeline of tools from the freely available Text Tokenisation Toolkit (TTT) (Grover, Matheson, Mikheev & Moens 2000). Each tool from the toolkit either removes markup or adds new markup without damaging previous markup. The toolkit works on XML text.

Although TIMEX and NUMEX were regarded as trivial, were found by creating a grammar, and only required a single tool to identify them, a pipeline of processes was needed to mark up the ENAMEX expressions. The pipeline consisted of five steps: Sure-fire rules, Partial Match 1, Relaxed rules, Partial Match 2, and Title Assignment.

Sure-fire rules are used to mark up expressions which are not only likely on the basis of lexicons and gazetteers but which also have context that justifies the markup. Thus, although in general "Washington" will not be marked up as a place at this stage, when found in the context "Washington area" it will be marked up as such. Similarly "Gates" would not be marked up as a person but within the context "Founder Bill Gates" it would be so marked up.

Partial Match 1: using the information already found in the Sure-fire rules, partial matches of those expressions already found are marked as possible entities. If "Lockheed Martin Production" has been labelled as an organization, and later in the text the expression "Lockheed Martin" occurs, it will be marked as a possible organisation. The second step in this partial match stage uses a Max Ent model to decide whether the context is supportive of the possible markup. If the Max Ent model decides that it is supportive, the markup is added.

Relaxed rules are used to mark up expressions that have not been classified. At this stage 'Washington' would be classified as a location unless it had already been classified as a person in step 1.

Partial Match 2 uses exactly the same principles as Partial Match 1 to find, for example, references to 'White' after identifying that 'John White' is indeed a person.

Title Assignment deals with case insensitive titles of newswire text, by attempting to find partial matches within the full body text. The Max Ent model is used again to make the final decision.

### 2.1.2 Non-standard and Non-English named entities

Although much of the research that has gone into named entity extraction has been focused on standard named entities in English, a number of groups have focused attention on the detection of non-standard named entities and on the detection of named entities in other langauges. Much of this work has been based on the most successful methods found when investigating the standard named entities.

The GENIA project (Collier, Park, Ogata, Takeishi, Nobata & Ohta 1999) has led a number of groups to focus on named entity extraction of DNA structures, finding genes, proteins, DNA sequences etc within biology/medical documents (Sun, J, Zhang, Zhou & Huang 2000), (Henderson, Salzberg & Fasman 1997).

Non-English named entity extraction has also been considered: Japanese (Sekine & Eriguchi 2000), Mandarin Chinese (Sun, J, Zhang, Zhou & Huang 2002), Arabic, French, German, Finnish, Malagasy, Persian, Polish, Russian, Spanish and Swedish (Poibeau, Acoulon, Avaux, Beroff-Bnat, Cadeau, Calberg, Delale, De Temmerman, Guenet, Huis, Jamalpour, Krul, Marcus, Picoli & Plancq 2003).

Cross-Lingual Multi-Agent Retail Comparison (CROSSMARC) (Karkaletsis, Spyropoulos, Souflis, Hachey, Pazienza, Vindigni, Cartier & Coch 2003), a European project with partners in France, Greece, Italy and the United Kingdom, has been focussing on general information extraction in each of the respective languages. The project initially considered a laptop domain - finding laptops for sale on the internet. The named entities included laptop manufacturers, processor speeds, and hard disk capacities, and the project sought to show the ease of portability to a completely different domain - job vacancies in the IT/Telecom industry - with named entities including job title, programming languages, area. The transition was possible, although not as trivial as may have been expected (Farmakiotou, Karkaletsis, Samaritakis, Petasis & Spyropoulos 2002).

### 2.1.3 Competitions

A number of competitions, generally sponsored by DARPA, have been run in the field of named entity extraction. The most well known of these conference competitions is the Text Retrieval Conference (TREC) (Harman 1993). The first of these conferences were the Message Understanding Conferences (MUC) (Chinchor 1997), with other notable conferences being Automatic Content Extraction (ACE) (Chinchor, Brown, Ferro & Robinson 1999), and National Institute of Standards (NIST) Hub - tasks aimed at spoken data (Pallett 1997).

Notable benefits arising from the strong conference competition approach included: agreed standards for defining named entity classes, standard data sets, comparable results from different communities and different methods, and readily available tools for scoring. There were, however, drawbacks caused by the conference competition approach, including: difficulty in new groups competing against groups that were better informed, the lack of diversity in tasks attempted, and the fact that systems get fine-tuned to maximise a particular 'score' - in the named entity case the F-score defined in section 3.4.

Competition results have tended to follow a certain trend (figure 2.1), with results showing initial rapid improvement but with the improvement then tailing off in accordance with the law of diminishing returns (Malthus 1798). Once the improvements have tailed off, a new task is designed to be harder than the previous task and the trend is then repeated.

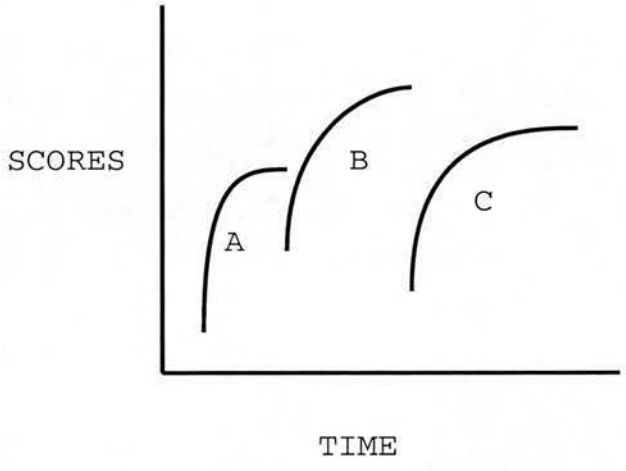


Figure 2.1: *Graph of typical learning curve for three successive competitions.*

## 2.2 The uses of named entity extraction

Named entity extraction is important not only in its own right, but also because of its application to other fields. By extracting what are essentially the most important words of a document and correctly identifying their class we can:

- search more effectively, whether by way of an internet search or an offline search (Mihalcea & Moldovan 2001)
- enable the automatic indexing of books (Tulic 2002)
- take a step towards producing good summarization (Nobata, Sekine, Isahara & Grishman 2002)
- help in the task of question answering (Srihari & Li 1999)
- aid in the process of machine translation (Babych & Hartley 2003)
- add an essential component to the more advanced stages of information extraction e.g. database construction (Grover, McDonald, NicGearailt, Karkaletsis, Farmakiotou, Samaritakis, Petasis, Pazienza, Vindigni, Vichot & Wolinski 2002)

## 2.3 Named entities in speech

The most prominent source of information extraction systems from speech was the 1998 Hub4 evaluation (Przybocki, Fiscus, Garofolo & Pallett 1999). In this evaluation, five systems (each provided with identical training material) had their results compared on the same test set. BBN Technologies entered one system (Kubala, Schwartz, Stone & Weishedel 1998), the MITRE Corporation entered one system (Palmer, Burger & Ostendorf 1999), the University of Sheffield Department of Computer Science entered two systems (Renals, Gotoh, Gaizauskas & Stevenson 1999) and SRI International Artificial Intelligence Center entered one system (Appelt & Martin 1999).

Two of the five systems that entered the Hub4 named entity evaluation were rule-based, the remainder used statistical methods. All five systems took as input the single most likely word sequence produced by a speech recogniser. Some systems used multiple transcriptions, but the use of the multiple transcriptions was to compare final output, rather than to try and improve final output.

### 2.3.1 Rule-based

The rule-based systems were SPRACH-R (Renals et al. 1999) entered by Sheffield University and TextPro (Appelt & Martin 1999) entered by SRI International.

SPRACH-R uses a rule-based approach that was ported from the rule-based system used in the MUC7 competition (Humphreys, Gaizauskas, Azzam, Huyck, Mitchell, Cunningham & Wilks 1998). The rule-based approach relied upon finite state matching against lists of single or multi-word names and named entity cue words, part of speech (POS) tagging, and specialised named entity parsing based on phrasal grammars for the named entity classes.

The SPRACH-R process consisted of a pipeline of components to produce: pseudo sentences, lookup in gazetteers, POS tags and finally a named entity parse. Each of the components is described below.

The pseudo-sentence segmenter split the speech transcripts into suitable length phrases. Typically, phrases needed to be no more than 40 words in duration due to efficiency constraints of the POS tagger and the parsing components of the pipeline.

The pseudo sentence segmenter was not required to find actual sentence breaks but simply to break the text into suitable size chunks without splitting any named entities; clearly it would be impossible to correctly identify a named entity split over multiple pseudo sentences.

The gazetteer comparison stage involved the lookup of typical single and multiword names. The gazetteers included Christian names, personal titles, common locations and organisations, location cue words such as "quay" and company designators such as "corporation". The SPRACH-R system used the Tipster architecture (Grishman 1995) which allowed multiple tags to be assigned per word by the gazetteers. (XML would not have allowed the overlap of markup in the same way.)

The POS tagger was a version of the Brill transformation-based tagger (Brill 1995) retrained on a single case version of the Penn Treebank.

The named entity parsing was performed initally by a bottom-up partial chart parse based on a number of regular grammars. The rules were based upon the already associated markup from gazetteers and POS tags, whereby named entities are associated to parts of sequences of markup. Due to the non-uniqueness of previous markup, it is possible for multiple named entity sequences to be generated by the initial parsing. It is therefore necessary to have a best parse selected by a "best-parse" algorithm. Full details of the named entity parsing component are given in (Wakao, Gaizauskas & Wilks 1996).

TextPro, the system used by SRI in the Hub4 evaluation, is a lightweight interpreter of cascaded finite-state transducers. This system also uses the Tipster architecture (Grishman 1995) that SPRACH-R used. TextPro was developed for standard text documents and was adapted to use the "noisy text" of speech transcriptions.

TextPro, rather than FASTUS (Hobbs, Appelt, Bear, Israel, Kameyama, Stickel & Tyson 1996), SRI's earlier named entity extraction system, was adopted due to its effectiveness for the task, together with its comparative size and speed of operation. The grammar used, however, was adapted from the FASTUS grammar used in the 1996 Message Understanding Conference. The adaptation was necessary due its optimisation on mixed case text. To help the system deal with single case text, an additional four lexicons were added to those of the FASTUS system: a list of US place names with some manually selected non-US place names added; a list of person names from many nationalities; a list of multi-national companies; a list of US government agencies and departments.

One significantly valuable rule was the use of identifying people names when, for example, only the Christian name was present. By means of the context, from which the full name could be identified, it was possible to check for the presence of any part of the name. It was also important, however, to be able to avoid this happening in the case of name fragments or repairs due to the speaker's error. The rules therefore took into account both immediate and surrounding context. Rules were tweaked by iteratively testing the system output against the training data.

### 2.3.2 Stochastic

The remaining three systems that were entered into the 1998 Hub4 competition were all stochastic. BBN entered IdentiFinder (Miller, Schwartz, Weischedel & Stone 1999), a system previously designed for text, adapted to work on speech transcripts. The MITRE Corporation entered NYMBLE (Palmer et al. 1999), a similar system based roughly on the design of IdentiFinder with a few differences. Sheffield University entered SPRACH-S (Renals et al. 1999).

All three of these systems were based on a stochastic finite state machine structure making use of multiple language models to predict the named entity sequence. The general topology of the systems is described in detail in chapter 4, where we also detail what distinguishes the three systems from each other. In this thesis we adopt the general structure of these three systems, as described in chapter 5.

A further notable statistical system which has been designed for speech which was not entered into the Hub4 competition was Ji-Hwan Kim's system (Kim 2001). This system, unlike the others, was based on more advanced transcriptions of speech data. The input to this system was still speech transcripts, but whereas the other systems assumed that speech transcripts were noisy text, Ji-Hwan Kim's system showed how it was possible to convert the noisy text to less noisy text. Where other systems argued that the reasons for lower F-score on speech transcripts included lack of punctuation, lack of capitalisation etc. this system focussed on adding this information to the transcripts. By the examination of pauses and prosody, this information was added into the transcripts.

The named entity extraction was performed by rules which were automatically generated from training material. All rules fitted one of the 53 templates shown in table 2.1. The approach for generating and selecting rules was similar in style to the generation of rules for Eric Brill's Transformation-Based POS tagger discussed below (Brill 1995).

The final aim of the named entity extraction was to further improve the noisy text by correctly dealing with named entities; that is, for example, having decided that "marks and spencer" is an organisation, altering the transcription to present it as "Marks and Spencer" as would appear in written text.

## 2.4 Part of speech tagging

POS tagging is the process of assigning the correct selection, from prespecified syntactic groupings (eg. NN1, DET), to a word. POS tagging is essentially a form of shallow parsing. Accuracy for automatic POS tagging on text documents is exceptionally high (with error rate comparable to human error rate).

POS tagging has value both in its own right and also as a pre-processing step in many other technologies, such as parsing. Indeed, many of the named entity extraction systems for text data make use of the POS in a variety of ways.

A great number of POS tagging systems have been implemented, as the task is not a new one. The most famous POS tagging system being the Brill (Brill 1995) Transformation-Based POS tagger, which compared favourably with another stochastic POS tagger in (Brill 1994) where the Brill tagger used only 267 simple, learned rules compared with approximately 10,000 learned probabilities of the other system.

Another notable tagger is Trigrams'n'Tags (TnT) (Brants 2000) which has been used extensively due to its portability to new domains, ease of use, speed and availability; for example, (Tufis, Dienes, Oravecz & Varadi 2000), (van Eynde, Zavrel & Daelemans 2000) and (Dzeroski, Erjavec & Zavrel 2000) make use of this tagger for a variety of languages and tasks. TnT is not optimized for any particular language and incorporates several methods for smoothing and handling unknown words. TnT is an implementation of the Viterbi algorithm, detecting weights for linear interpolation by using deleted

| Rule & Range | | |
|---|---|---|
| $w_0f_0[0,0]$, | $w_0f_{-1}[-1,0]$, | $w_0f_1[0,1]$ |
| $w_0w_1[0,1]$, | $w_0w_{-1}[-1,0]$, | $w_0t_1[0,1]$ |
| $w_0t_{-1}[-1,0]$, | $w_1t_0[0,1]$, | $w_{-1}t_0[-1,0]$ |
| $t_0t_1[0,1]$, | $t_0t_{-1}[-1,0]$, | $w_0f_{-1}[-1,0]$ |
| $w_0f_1[0,1]$ | | |
| $w_0w_{-1}w_{-2}[-2,0]$, | $w_0w_1w_2[0,2]$, | $w_0w_{-1}w_1[-1,1]$ |
| $w_0t_1[0,1]$, | $w_0t_{-1}[-1,0]$ | $w_1t_0[0,1]$ |
| $w_{-1}t_0[-1,0]$, | $w_0w_1t_2[0,2]$, | $w_0w_1t_{-1}[-1,1]$ |
| $w_0t_1w_2[0,2]$ | | |
| $w_0f_0b_0[0,0]$, | $w_0f_0b_0b_1[0,0]$ | |
| $w_0w_{-1}t_0t_{-1}[0,0]$, | $w_0w_1t_0t_1[0,0]$ | |
| $w_0f_0[0,0]$ | | |
| $w_0t_0t_{-1}[-1,0]$, | $w_0t_0t_1[0,1]$ | |
| $w_{-1}w_{-2}t_0f_0[0,0]$, | $w_1w_2t_0[0,0]$, | $w_{-1}t_0[0,0]$ |
| $w_1t_0[0,0]$ | | |
| $w_{-1}f_{-1}f_0[-1,0]$, | $w_1f_1f_0[0,1]$, | $w_0f_0t_{-1}[-1,0]$ |
| $w_0f_0t_1[0,1]$ | | |
| $w_{-1}f_{-1}f_0[0,0]$, | $w_1f_1f_0[0,0]$, | $w_0f_0t_{-1}[0,0]$ |
| $w_0f_0t_1[0,0]$ | | |
| $w_0t_{-1}t_1f_0[-1,1]$, | $w_0f_{-1}f_1f_0[-1,1]$, | $w_0f_1w_2[0,0]$ |
| $w_{-1}f_{-1}w_{-2}[0,0]$, | $w_1f_1ft_2[0,0]$, | $w_0f_{-1}t_{-2}[0,0]$ |
| $w_0w_{-1}[0,0]$, | $w_0w_1[0,0]$, | $w_0w_{-1}w_{-2}[0,0]$ |
| $w_0w_1w_2[0,0]$, | $w_0w_{-1}w_1[0,0]$ | |

Table 2.1: *Templates of rules for Kim's rule induction (w:words; f:word features; t:named entity classes). Subscripts define the distance from the current word and bracketed numbers indicate the range of rule application.*

interpolation (Brown, Pietra, deSouza, Lai & R 1992). More recent POS tagging systems (e.g. (Curran & Clark 2003)) have effectively used Max Ent for POS identification.

In (Spilker, Weber & Görz 1999) POS has been used for the correction of speech repairs within word lattices. The paper describes a method for finding potential speech repairs within word lattices using only acoustic level features. Then, having detected potential positions of speech repairs, the POS tags associated with reparandum and reparans are compared. Provided the POS tags match, the "repair" is considered plausible. This method doesn't require the tagging of whole lattices, but rather the tagging of n-best lists corresponding to subsections of the lattices.

In (Heeman & Allen 1997) POS has also been used in conjunction with word lattices. This paper focuses on speech recognition directly rather than on improving the lattices. The paper describes a method for reducing the perplexity of the speech recogniser. Instead of using the standard approach of finding the word sequence which maximises the probability of the words given the acoustic evidence ($\arg\max_{W} P(W|A)$), an approach which maximises the probability of the words and POS tags given the acoustic evidence ($\arg\max_{W,T} P(W,T|A)$) was considered. Classification and regression trees (CART) are then used to model the probabilities and thus generate the best sequence of words and POS tags for the given speech.

## 2.5 Summary

In this chapter we have been concerned with the methods that have been used for the extraction of named entities. We have briefly considered research which has used rules, statistics or a combination of these to identify named entities. We have explained that previous named entity work on speech data has focussed entirely on transcripts of the data (both manual and automatic), and we have discussed briefly the systems that have been used on speech transcripts.

Finally we have discussed POS tagging and in particular the ways in which POS tagging has been used in connection with named entity extraction and separately used in connection with word lattices.

# Chapter 3

# Data preparation and evaluation methods

In this chapter we introduce the data used throughout this thesis and then describe the evaluation metrics that are used to score the named entity extracted data. Most of the data used in this thesis is available to the public from the Linguistic Data Consortium; some, however, was obtained from Cambridge University Engineering Department (CUED) and is not available publicly.

There are essentially two distinct formats of data used in this thesis: text and lattices. At its simplest, text refers to sequences of words, whilst lattices (word-lattices) contain information with respect to time (in particular possible words at particular times). Within this broad categorisation, text may be coherent, may contain xml markup, may be single case, and may have punctuation, but for this general categorisation none are required. Similarly, lattices may offer only a single path, may contain part-of-speech information, and may contain accurate time stamps or may not. Examples of a lattice and some text are given in figures 3.1 and 3.2 respectively.

There are three distinct sets of data; namely training data, development data and test data. The training data is used to produce language models and to observe any patterns. The development data is a held-out set of data, which is not used for the purpose of training - but rather as a means of testing any named entity extraction system. The development data is effectively a data set on which named entity extraction systems are

```
N=10   L=11
J=0    S=0   E=1   W=!SENT_START   a=-129.05    l=0.00
J=1    S=1   E=2   W=TO            a=-3036.62   l=-5.622
J=2    S=2   E=3   W=WRECK         a=-3046.72   l=-9.542
J=3    S=2   E=5   W=RECOGNISE     a=-4032.03   l=-15.219
J=4    S=3   E=4   W=A             a=-636.12    l=-6.408
J=5    S=4   E=5   W=NICE          a=-1298.77   l=-4.815
J=6    S=5   E=6   W=SPEECH        a=-1083.14   l=-5.312
J=7    S=5   E=7   W=BEACH         a=-1071.32   l=-5.567
J=8    S=6   E=8   W=!SENT_END     a=-1424.97   l=-1.092
J=9    S=7   E=8   W=!SENT_END     a=-1424.97   l=-1.854
J=10   S=8   E=9   W=!NULL         a=0.0        l=0.0
```

Figure 3.1: *An example speech word-lattice.*

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

"Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!?"

Figure 3.2: *An example of text (from Lewis Carroll's Jabberwocky).*

optimised before finally being tested on the test set. After testing on the test set, results are recorded - the models are not re-adjusted. This systematic approach, illustrated in figure 3.3, is followed throughout the thesis. The precise details about each data set are given in section 3.3.

## 3.1 General data preparation

The two sources of data, the Linguistic Data Consortium (LDC) and Cambridge University, provided data in different formats. The LDC provided transcripts of broadcast news in Universal Transcription Format (UTF); an example of UTF format broadcast news is given in appendix C, together with the sgml document type definition. The Cambridge data were word-lattices and were in the HTK binary lattice format. A HTK-based tool was built to convert between HTK binary format and HTK ascii format lattices.

The data in UTF format was corrected for consistency of abbreviations both internally and also for consistency with the HTK lattices. For example, the one source used the format _I _B _M whereas the other used I. B. M. The UTF data then had all non-entity specific markup removed. All word fragments, background noise, non-speech (e.g. %breath), and punctuation were also removed. Finally the case of the data was changed to uppercase. These changes were to facilite consistency with the word-lattices - since word-lattices do not contain background noise, non-speech or punctuation.

Some repeat files existed in the training data; the purpose of these was presumably inter-annotator comparison. In such cases the decision was made to use those files annotated by BBN rather than those annotated by NIST; duplicate NIST files were ignored.

The document that remained was effectively a key, a manual transcription of the speech with manual named entity markup, suitable for scoring the system output against. The next stage was the removal of the named entity markup, which in turn produced a document which could be used as input to our named entity extraction system. The output of the system, the hypothesis, could be scored against the document with manual named entity markup, the key.

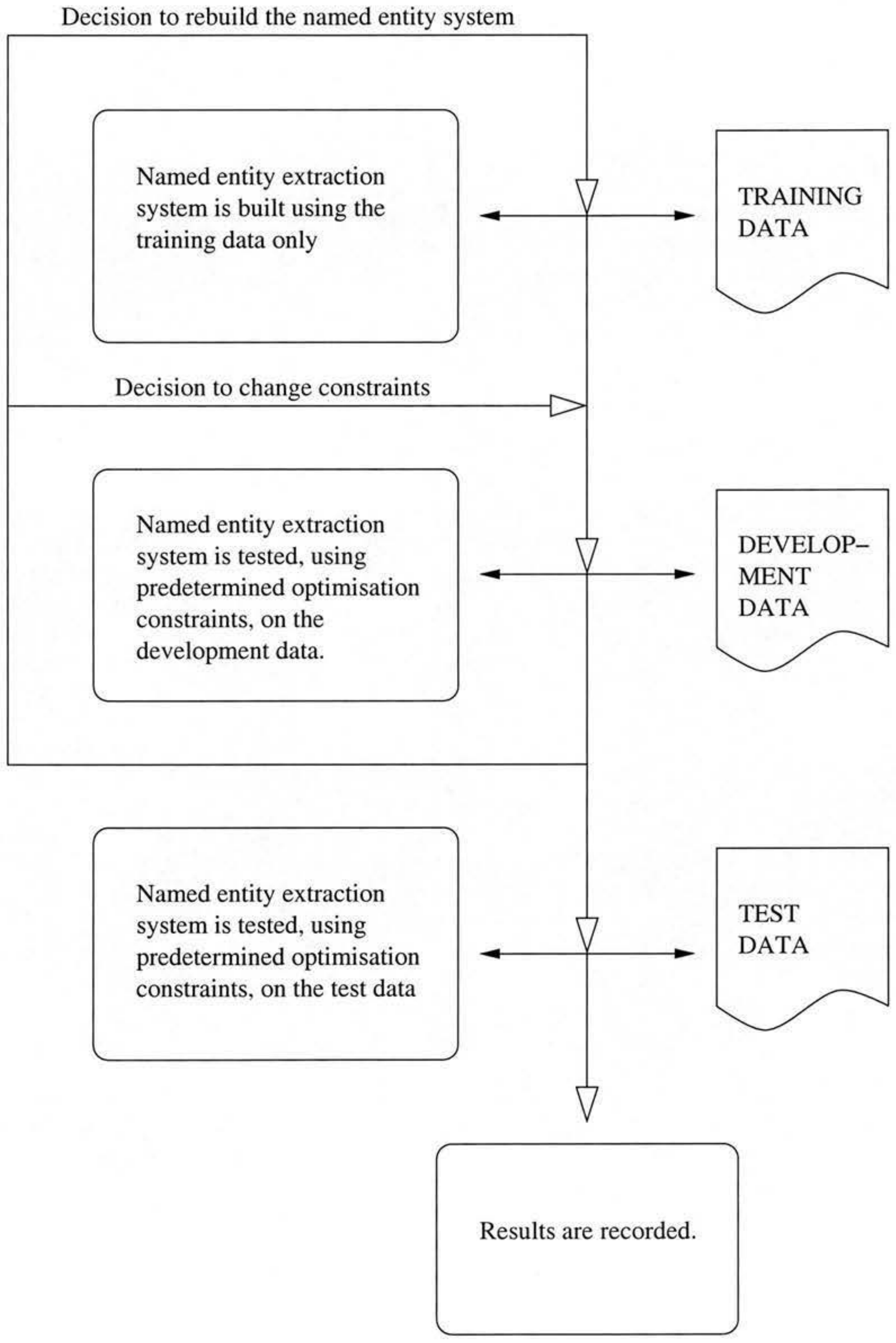

Decision to rebuild the named entity system
Named entity extraction system is built using the training data only
TRAINING DATA
Decision to change constraints
Named entity extraction system is tested, using predetermined optimisation constraints, on the development data.
DEVELOP–MENT DATA
Named entity extraction system is tested, using predetermined optimisation constraints, on the test data
TEST DATA
Results are recorded.

Figure 3.3: *The systematic approach to training, re-training and testing any system using the relevant data sets.*

## 3.2 Converting between lattices and text

Throughout the thesis there is a reliance upon maps from text to lattices and vice versa. The following subsections set out a number of maps for both of these.

### 3.2.1 Text to Lattices

We use an intuitive method for converting from a general text to a general lattice. Our method adds "non-information", ie content that carries no information. There is therefore no real significance to the time stamps or to the acoustic and language model probabilities. We simply map the first word of the text to start at time stamp 0 of the lattice and end at time stamp 1, the second word to start at time stamp 1 and end at time stamp 2, the nth word to start at time stamp n-1 and end at time stamp n. We add a null word to the end of the lattice for compatibility with the other lattices. We also add any necessary word features even if the values are irrelevant - the lattices that we use require acoustic and language model log probablities and so these are added. The log probabilities for words we associate with -1 allow us to keep track of how many words have been processed. There is, however, no mathematical significance to these numbers (and consequently the value is irrelevant).

For example, the sentence:

!SENT_START THE GOVERNMENT REPORTED SOME NEWS THAT SUGGESTS LOWER INFLATION AHEAD !SENT_END

becomes:

| | | | | | |
|---|---|---|---|---|---|
| N=14 | L=13 | | | | |
| J=0 | S=0 | E=1 | W=!SENT_START | a=-1 | l=0.00 |
| J=1 | S=1 | E=2 | W=THE | a=-1 | l=-1 |
| J=2 | S=2 | E=3 | W=GOVERNMENT | a=-1 | l=-1 |
| J=3 | S=3 | E=4 | W=REPORTED | a=-1 | l=-1 |
| J=4 | S=4 | E=5 | W=SOME | a=-1 | l=-1 |
| J=5 | S=5 | E=6 | W=NEWS | a=-1 | l=-1 |
| J=6 | S=6 | E=7 | W=THAT | a=-1 | l=-1 |
| J=7 | S=7 | E=8 | W=SUGGESTS | a=-1 | l=-1 |
| J=8 | S=8 | E=9 | W=LOWER | a=-1 | l=-1 |
| J=9 | S=9 | E=10 | W=INFLATION | a=-1 | l=-1 |
| J=10 | S=10 | E=11 | W=AHEAD | a=-1 | l=-1 |
| J=11 | S=11 | E=12 | W=!SENT_END | a=-1 | l=-1 |
| J=12 | S=12 | E=13 | W=!NULL | a=0.0 | l=0.0 |

In a general lattice: N is the total number of nodes; L is the total number of links between nodes; J refers to a specific link; S refers to the start time stamp of link J; E refers to the end time stamp of link J; W refers to the word that is associated with link J; a is the accoustic likelihood, and l is the language model log probability of the word as calculated by the speech recogniser (or, in the above example, simply added for completeness).

### 3.2.2 Lattices to Text

Conversions from lattices to text are not as trivial as conversion from text to lattices. This is because it is not always possible to represent all the information of a lattice within a text. Not only is there additional information (for example in the form of log probabilities) to be dealt with, but, as can be seen from the example in figure 3.1, it is not possible to convert the lattice into a single text while maintaining all of the word information. This is because there is no unique ordering of the words which is in keeping with the lattice: for example, both 'beach' and 'speech' should start simultaneously.

It is possible to visualise a lattice pictorally. Figure 3.4 sets out the the pictoral

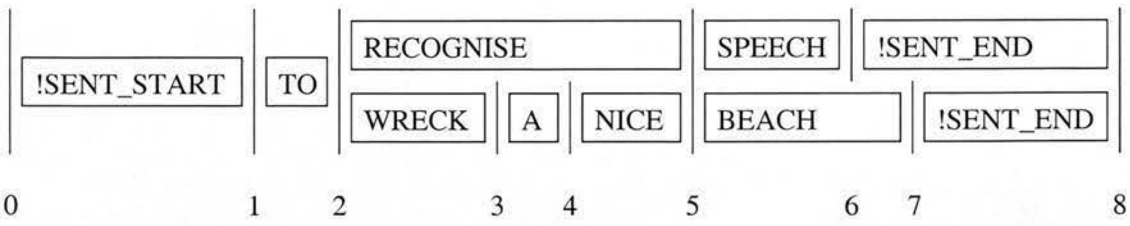


Figure 3.4: *Pictoral representation of the lattice in figure 3.1. Vertical bars represent time stamps.*

representation of figure 3.1. It is not possible, however, to represent it in standard text. Consequently a number of methods are used to convert from lattice to text.

### Trivial Lattices to Text

These refer to lattices with a unique path through them, as in the example in subsection 3.2.1. The solution is simply to take the words in the order that they occur within the lattice.

### Lattices to Text by Probability (1-best)

This is the process that speech recognisers use to find the 1-best path through the lattice. Essentially, a weighted log-likelihood of the acoustic likelihoods and the language model likelihoods, together with a fixed penalty per word, is used to generate a likelihood associated with each path through the lattice. The possible path with the maximum likelihood is selected. The lattices supplied by Cambridge University came with the best relative weightings: 1 and 14 respectively for acoustic likelihoods and language model likelihoods, and a word insertion penalty of 10.

By using a Viterbi search (Viterbi 1967), it is possible to efficiently find the 1-best path through each of the lattices. Having found the 1-best path, it was stored for later experiments.

For the lattice in figure 3.1 the 1-best path is "TO RECOGNISE SPEECH" with a log probability of -10137.234 = -129.05 + -3036.62 + -4032.03 + -1083.14 + -1424.97 + 14x(-1.092 + -5.312 + -15.219 + -5.622) + -10x5.

### Lattices to Text by Reference (lattice-best)

Another way of creating a text from the lattice is to compare the lattice with the key text, the idea being to see how close a transcription (path through the lattice) is possible to the text of the key. The lattice-best path is useful as it provides an absolute lower bound on the word error rate (WER) for any text generated by the lattice. Correspondingly, the best named entity marked up version of this lattice-best text provides an absolute upper bound on the F-measure possible from the lattice.

An example, using the lattice in figure 3.1 would be to identify the closest text possible to the text "TO WRECK THE NICE BEACH"; the correct output would be "TO WRECK A NICE BEACH".

The process is similar to the dynamic alignment of two texts to compare the differences between them. The difference is that there may now be multiple ways of getting a correct match, and correspondingly far more ways of getting an incorrect match.

The method relies on the use of an N x n grid, where 'N' refers to the number of nodes in the lattice, and 'n' refers to the number of words in the manual transcription. Starting at the bottom left corner of the grid and propagating through the end nodes of words in the lattice it is possible to keep track of insertion errors, deletion errors and substitution errors. Deletion errors work horizontally, insertion errors work vertically and substitution errors work diagonally.

This is illustrated for our toy example in figure 3.5: an 8 x 7 grid. There are multiple word alternatives where multiple words end at the same time stamp. The dynamic alignment is always the path with the least errors. A maximum of two alternative ways of reaching any given square has been shown to aid clarity in the diagram. In a number of places there are a number of alternative ways which have not been shown. The algorithm ensures that only one hypothesis needs to be stored for each square in the grid.

In practice, after the dynamic alignment, all squares in the grid will contain tokens, each of which points back in the direction of the best path through the grid (that is, the best path through the lattice), rather than just those illustrated.

Similarly each of those illustrated will contain only a single token, although there

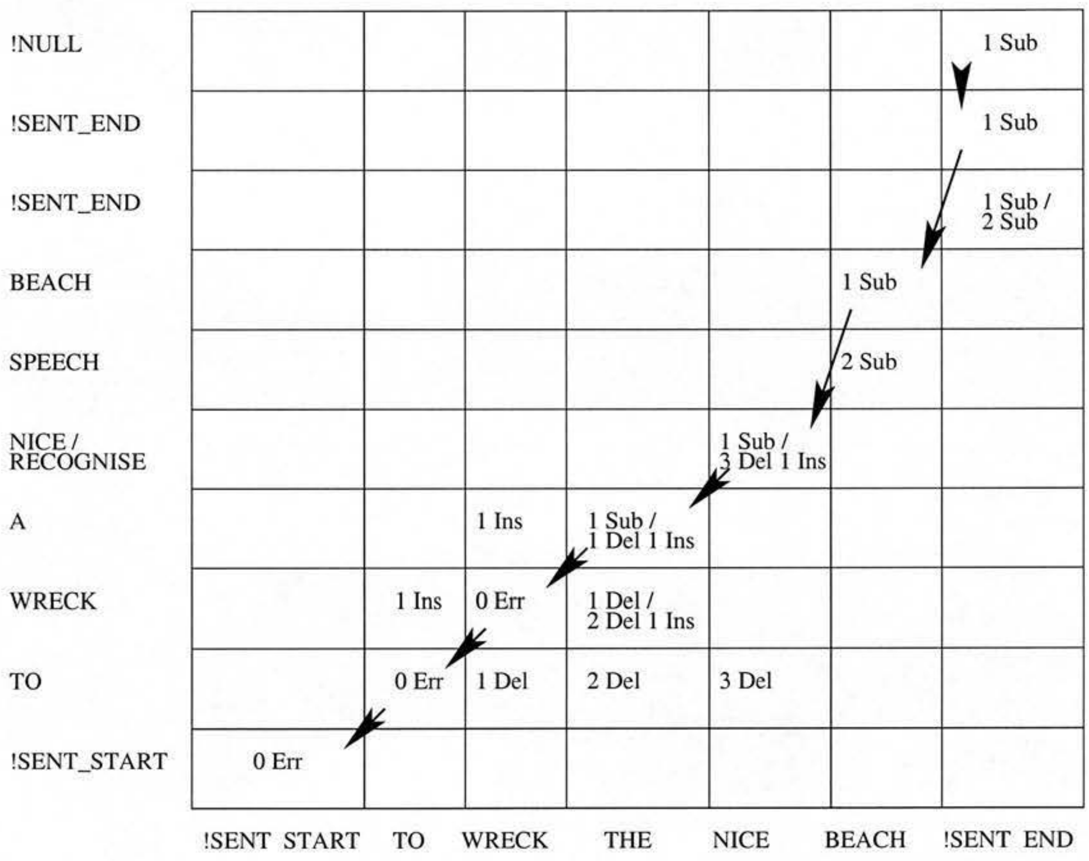


Figure 3.5: *Illustration of the dynamic alignment of a lattice with a text. 'Ins' represents insertion errors, 'Del' represents deletion errors, 'Sub' represents substitution errors, '0 Err' represents perfect match. The best path can be traced by following the arrows.*

may be many alternatives. For example:

- It is possible to have 1 insertion error and 1 deletion error where the word WRECK matches the word WRECK; in practice only the 0 error would be stored.
- It is possible to have 3 insertion errors or just 1 insertion error where the word NICE/RECOGNISE matches the word TO; in practice only the 1 insertion error would be stored.

Finding the dynamic alignment, and thus the corresponding text, is simply a case of tracing the path backwards from the top right hand corner of the grid. It is also possible to find the best path to any node in the lattice (with respect to the key text) simply by finding the least number of errors in the respective row of the grid and tracing the path backwards. Similarly, it is possible to find the best path corresponding to any section of the key text by finding the least number of errors in the respective column of the grid and tracing the path backwards. In this thesis we only find complete paths. Appendix B shows, step by step, the dynamic alignment of the example lattice and text.

Having found the lattice-best transcript of test data, when scored against the key, the word error rate (WER) was found to be 5.4%.

## 3.3 Data sets

### 3.3.1 Training data

The training data we used were the utf transcripts of broadcast news available from the Linguistic Data Consortium with catalogue references LDC98E10 and LDC98E11. There were 53 files which contained named entity markup in LDC98E10 and 114 files in LDC98E11. There were 12 duplicate files, which were ignored, 7 were used for development data, leaving a total of 148 files. Each file contained a broadcast news program transcript, each program having a duration of between 30 minutes and 1 hour. These were all preprocessed, as described in section 3.1 (general data preparation), to produce keys and input texts. Lattices were also produced from the input texts using the method described in section 3.2.1.

In total the training data contained just over one million words; within these words were almost fifty thousand named entities.

### 3.3.2 Development data

The data refered to as the development data is simply a held-out subset of the training data. A random sample of 7 files (approximately 10% of the training data - 10 hours) was held out for this purpose; the exact files were a960624_.ref.utf d960604a.ref.utf ea980120.bbn.utf eh971017.bbn.utf f960603b.ref.utf i960604_.ref.utf k960524_.ref.utf. This data was processed in an identical manner to the training data. As already explained this data was not used when training the system. Again, lattices were produced from the input texts using the method described in section 3.2.1.

### 3.3.3 Testing data

A number of test sets have been used in various tests on named entity extraction from speech. The main test set used was h4e_97.ref.utf, which was available within the distribution of LDC98E10 from the LDC. The principal reason for treating this as the main test set was that this data corresponded with the word-lattices offered by Cambridge University.

Processing of the test set h4e_97.ref.utf yielded our evaluation material, and also a manual transcription of the speech in the form of a lattice. We also had the HTK lattices corresponding to this file. This test set comprised four 30-minute news programmes, two from television and two from radio. The test data was distinct from the training and development data, but had been recorded at a similar time. The test data contained just under 35,000 words, and a target of 1,905 named entities.

By evaluation of the HTK lattices, using the accoustic and language model likelihoods, we were able to determine the 1-best transcription of the speech data according to HTK. Then by the text-to-lattice processing we produced the lattices corresponding to the 1-best transcription. The reason for converting this to a lattice was to enable the 1-best transcription to be handled by the system for named entity extraction from lattices once this system was built.

| | Words | Entities | ENAMEX | TIMEX | NUMEX | Duration |
|---|---|---|---|---|---|---|
| Training | 989258 | 46504 | 40851 | 3508 | 2145 | $\approx$ 90 hours |
| Development | 105267 | 2749 | 2334 | 320 | 95 | $\approx$ 10 hours |
| Testing | 33582 | 1905 | 1694 | 138 | 73 | 4 hours |

Table 3.1: *Frequency of words and entities within the data.*

By dynamic alignment of the HTK lattices with the manual transcription we were also able to ascertain a theoretical best possible path through the lattices (lattice-best) and generate the simplified lattices corresponding to this best possible path through the lattices.

Table 3.1 shows the exact frequency of words and named entities for each of the data sets.

For the purpose of the thesis each data set has been named to reflect its content and type. Table 3.2 summarises the names used for each data set and details briefly how each set was obtained and prepared.

## 3.4 Evaluation measures

We have already referred to F-scores, the most common score for evaluating named entity extraction. We now introduce F-scores more formally, together with an alternative measure, a slot error rate (SER), akin to a WER.

The task of scoring the output of a named entity recognition system against a key or reference transcript is a fairly simple process for text documents. A comparison is made between which named entities have been found correctly and which have not. For recognised speech input the task is more difficult because there is no simple one-to-one correspondence between hypothesis and key. In order to obtain a fair comparison between the hypothesis and the key a dynamic alignment of the two texts is necessary to match corresponding words within the transcripts.

Once the alignment has been completed, comparisons between the individual named entity slots (that is each individual piece of markup) are made. There are essentially four possible outcomes for any given slot: (i) the markup can be correct; (ii) the markup

| Type | Name | Description |
|---|---|---|
| Text | Training Key | The text that corresponds to the 148 files that contained markup. (155 - 7 development files) |
| | Development Key | The text that corresponds to the 7 files that contained markup. |
| | Manual Key | The text that contains a manual transcription of what was said with named entity markup. |
| | 1-Best Text | The text that corresponds to decoding the word lattices from Cambridge. |
| | Lattice-Best Text | The text that corresponds to finding the best possible path through the lattice by dynamic alignment with the manual transcription. |
| | Training Text | The text that corresponds to Training Text with the named entities removed - used to find word error rate (WER). |
| | Development Text | The text that corresponds to Development Text with the named entities removed - used to find WER. |
| | Testing Text | The text that corresponds to Testing Text with the named entities removed - used to find WER. |
| Lattice | Training Lattice | The lattices that correspond to Training Text. |
| | Development Lattice | The lattices that correspond to Development Text. |
| | Manual Lattice | The lattices that correspond to Testing Text. |
| | Speech Lattice | The lattices that were received from Cambridge University. |
| | 1-Best Lattice | The lattices that correspond to 1-Best Text. |
| | Lattice-Best Lattice | The lattices that correspond to Lattice-Best Text. |

Table 3.2: *A summary of the data used throughout the thesis*

can be incorrect - eg the wrong type, the wrong duration or the wrong words inside the markup; (iii) the markup can be missing or (iv) the markup can be spurious - ie not in the key.

We made use of the scoring software that was developed by the National Institute of Standards and Technology (NIST) (Fisher 1998) for the purpose of scoring the data in the HUB4 named entity task. In order to explain the method of calculation of F-scores and SER, the notation used by the scoring software to refer to the relative frequencies of these types of data has been adopted:

| | | |
|---|---|---|
| $cor$ | = | # of correct slots |
| $inc$ | = | # of incorrect slots |
| $mis$ | = | # of missing slots |
| $spu$ | = | # of spurious slots |
| $act$ | = | # of slots in the hypothesis |
| $pos$ | = | # of slots in the key |

By simple mathematics, $act = cor + inc + spu$ and $pos = cor + inc + mis$. Two values which are often referred to in the literature are the precision of the system (pre) - how accurate any slot is likely to be - and the recall of the system (rec) - the ratio of how many of the actual slots the system is likely to find. These values are calculated respectively by equations 3.1 and 3.2.

$$pre = \frac{cor}{act} \tag{3.1}$$

$$rec = \frac{cor}{pos} \tag{3.2}$$

The F-score which we will be referring to throughout the thesis is the uniformly weighted harmonic mean of precision and recall; as set out in equation 3.3. The F-score is the primary figure that is used to evaluate named entity extraction systems. The greater this value the better the system. Although it is possible to gain improvement in 'rec' at the expense of 'pre', and vice versa, F-score tends to decrease at such attempts. F-score is therefore a reliable estimate of how well the system performs.

$$F - score = \frac{2 \times pre \times rec}{pre + rec} = \frac{2 \times cor}{act + pos} \tag{3.3}$$

There is another fairly standard metric for evaluating named entity extraction - the SER. The SER, which is not unlike the WER in definition, is defined in equation 3.4.

$$SER = \frac{inc + mis + spu}{pos} \tag{3.4}$$

Throughout the thesis we refer to the F-score, this is the more generally accepted metric and has the added advantage that the function is symmetrical. It is symmetrical because, if a hypothesised document is scored against a reference document with the result that the F-score equals f, then, if the roles are reversed (the hypothesised document becomes the reference document and the reference document becomes the hypothesised document), the new F-score also equals f.

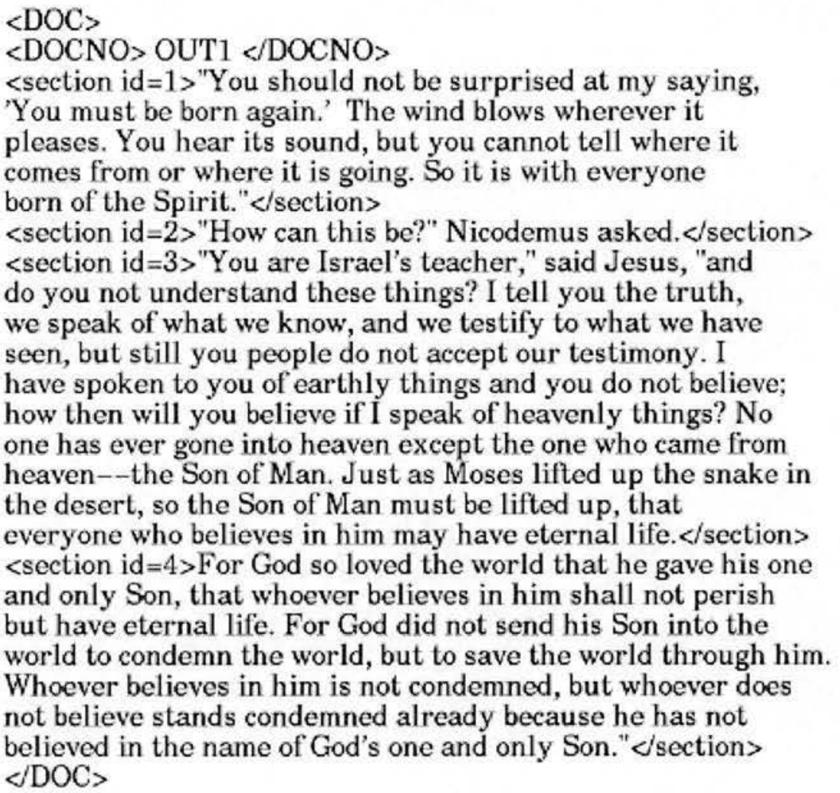
**Evaluation metrics by example**

In order to illustrate the evaluation metrics, a piece of text from the Bible[1] has been used. Figure 3.6 shows the Bible text in it's original format; figure 3.7 shows the bible text correctly marked with named entities (punctuation has been maintained to aid clarity); figure 3.8 shows an incorrect transcript - with incorrect markup.

The scoring software provides two separate output files when used to compare two documents. These files are a "tag-by-tag" explanation of how the scores are obtained and a table of calculated scores. The "tag-by-tag" breakdown for the comparison of figure 3.7 (the reference file) and figure 3.8 (the hypothesised file) is shown in table 3.3. The final score file for the comparison is shown in table 3.4; with scores per paragraph given in tables 3.5, 3.6, and 3.7 respectively.

The first paragraph contains no named entity information and consequently bears no impact on the score; indeed the scoring software completely ignores the paragraph and numbers the paragraphs commencing at the second paragraph. In this way the system

[1]Scripture taken from the HOLY BIBLE, NEW INTERNATIONAL VERSION®. Copyright ©1973, 1978, 1984 International Bible Society. Used by permission of Zondervan. All rights reserved.



```
<DOC>
<DOCNO> OUT1 </DOCNO>
<section id=1>"You should not be surprised at my saying,
'You must be born again.'  The wind blows wherever it
pleases. You hear its sound, but you cannot tell where it
comes from or where it is going. So it is with everyone
born of the Spirit."</section>
<section id=2>"How can this be?" Nicodemus asked.</section>
<section id=3>"You are Israel's teacher," said Jesus, "and
do you not understand these things? I tell you the truth,
we speak of what we know, and we testify to what we have
seen, but still you people do not accept our testimony. I
have spoken to you of earthly things and you do not believe;
how then will you believe if I speak of heavenly things? No
one has ever gone into heaven except the one who came from
heaven--the Son of Man. Just as Moses lifted up the snake in
the desert, so the Son of Man must be lifted up, that
everyone who believes in him may have eternal life.</section>
<section id=4>For God so loved the world that he gave his one
and only Son, that whoever believes in him shall not perish
but have eternal life. For God did not send his Son into the
world to condemn the world, but to save the world through him.
Whoever believes in him is not condemned, but whoever does
not believe stands condemned already because he has not
believed in the name of God's one and only Son."</section>
</DOC>
```

Figure 3.6: *John 3:7-18 (NIV)*

```
<DOC>
<DOCNO> OUT1 </DOCNO>
<section id=1>"You should not be surprised at my saying,
'You must be born again.' The wind blows wherever it
pleases. You hear its sound, but you cannot tell where it
comes from or where it is going. So it is with everyone
born of the Spirit."</section>
<section id=2>"How can this be?" <b_ENAMEX TYPE="PERSON">
Nicodemus<e_ENAMEX> asked.</section>
<section id=3>"You are <b_ENAMEX TYPE="LOCATION">Israel
<e_ENAMEX>'s teacher," said <b_ENAMEX TYPE="PERSON">Jesus
<e_ENAMEX>, "and do you not understand these things? I tell
you the truth, we speak of what we know, and we testify to
what we have seen, but still you people do not accept our
testimony. I have spoken to you of earthly things and you
do not believe; how then will you believe if I speak of
heavenly things? No one has ever gone into <b_ENAMEX
TYPE="LOCATION">heaven<e_ENAMEX> except the one who came from
<b_ENAMEX TYPE="LOCATION">heaven<e_ENAMEX>--the <b_ENAMEX
TYPE="PERSON">Son of Man<e_ENAMEX>. Just as <b_ENAMEX
TYPE="PERSON">Moses<e_ENAMEX> lifted up the snake in the
desert, so the <b_ENAMEX TYPE="PERSON">Son of Man<e_ENAMEX>
must be lifted up, that everyone who believes in him may have
eternal life.</section>
<section id=4>For God so loved the <b_ENAMEX TYPE="LOCATION">
world<e_ENAMEX> that he gave his one and only Son, that whoever
believes in him shall not perish but have eternal life. For
God did not send his Son into the <b_ENAMEX TYPE="LOCATION">
world<e_ENAMEX> to condemn the world, but to save the <b_ENAMEX
TYPE="LOCATION">world<e_ENAMEX> through him. Whoever believes
in him is not condemned, but whoever does not believe stands
condemned already because he has not believed in the name of
God's one and only Son."</section>
</DOC>
```

Figure 3.7: *John 3:7-18 (NIV) with named entity markup for scoring software*

```
<DOC>
<DOCNO> OUT1 </DOCNO>
<section id=1>"You should not be surprised at my saying,
'You must be born again.' The wind blows wherever it
pleases. You hear it's sound, but you cannot tell where it
comes from or where it is going. So it is with every one
born of the Spirit."</section>
<section id=2>"How can this be?" <ENAMEX TYPE="PERSON">Nick
</ENAMEX> oh dean must ask.</section>
<section id=3>"You are <ENAMEX TYPE="LOCATION">Israel's
</ENAMEX> teacher," said Jesus, "and do you not understand
these things? I tell you the truth, we speak of what we
know, and we testify to what we have seen, but still you
people do not accept our testimony. I have spoken to you of
earthly things and you do not believe; how then will you
believe if I speak of <ENAMEX TYPE="LOCATION">heaven</ENAMEX>
 things? No one has ever gone into <ENAMEX TYPE="LOCATION">
heaven</ENAMEX> except the one who came from <ENAMEX
TYPE="LOCATION">heaven</ENAMEX>--the Son of Man. Just as
<ENAMEX TYPE="PERSON">Moe</ENAMEX> says lifted up the snake
in the desert, so the Son of Man must be lifted up, that
everyone who believes in him may have eternal life.</section>
<section id=4>For God so loved the <ENAMEX TYPE="LOCATION">
world</ENAMEX> that he gave his one and only Son, that whoever
believes in him shall not perish but have eternal life. For
God did not send his Son into the <ENAMEX TYPE="LOCATION">
world</ENAMEX> to condemn the world, but to save the <ENAMEX
TYPE="ORGANIZATION">world</ENAMEX> through him. Whoever
believes in him is not condemned, but whoever does not believe
stands condemned already because he has not believed in the
name of God's one and only Son."</section>
</DOC>
```

Figure 3.8: *John 3:7-18 with imperfections and incorrect markup*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Story '.001' | | | | | | | |
| SUBTASK | TYPE | XTNT | CONT | KEY_TYPE | RSP_TYPE | KEY_CONT | RSP_CONT |
| ENAMEX | cor | cor | inc | PERSON | PERSON | "Nicodemus" | "Nick" |
| Story '.002' | | | | | | | |
| SUBTASK | TYPE | XTNT | CONT | KEY_TYPE | RSP_TYPE | KEY_CONT | RSP_CONT |
| ENAMEX | cor | cor | cor | LOCATION | LOCATION | "Israel " | "Israel ' s " |
| ENAMEX | spu | spu | spu | | LOCATION | "" | "heaven" |
| ENAMEX | mis | mis | mis | PERSON | | "Jesus " | "" |
| ENAMEX | cor | cor | cor | LOCATION | LOCATION | "heaven" | "heaven" |
| ENAMEX | cor | cor | cor | LOCATION | LOCATION | "heaven" | "heaven" |
| ENAMEX | mis | mis | mis | PERSON | | "Son of Man" | "" |
| ENAMEX | cor | cor | inc | PERSON | PERSON | "Moses" | "Moe" |
| ENAMEX | mis | mis | mis | PERSON | | "Son of Man" | "" |
| Story '.003' | | | | | | | |
| SUBTASK | TYPE | XTNT | CONT | KEY_TYPE | RSP_TYPE | KEY_CONT | RSP_CONT |
| ENAMEX | cor | cor | cor | LOCATION | LOCATION | "world" | "world" |
| ENAMEX | cor | cor | cor | LOCATION | LOCATION | "world" | "world" |
| ENAMEX | inc | cor | cor | LOCATION | ORGANIZATION | "world" | "world" |

Table 3.3: *The "tag-by-tag" output from comparing figure 3.7 with figure 3.8.*

| all_stories | cor | inc | mis | spu | pos | act | pre | rec | f | ser |
|---|---|---|---|---|---|---|---|---|---|---|
| all_entities | | | | | | | | | | |
| TYPE | 7 | 1 | 3 | 1 | 11 | 9 | 77.78 | 63.64 | 70.00 | 45.45 |
| XTNT | 8 | 0 | 3 | 1 | 11 | 9 | 88.89 | 72.73 | 80.00 | 36.36 |
| CONT | 6 | 2 | 3 | 1 | 11 | 9 | 66.67 | 54.55 | 60.00 | 54.55 |
| XTNT+CONT+TYPE | 21 | 3 | 9 | 3 | 33 | 27 | 77.78 | 63.64 | 70.00 | 45.45 |
| enamex | | | | | | | | | | |
| TYPE | 7 | 1 | 3 | 1 | 11 | 9 | 77.78 | 63.64 | 70.00 | 45.45 |
| XTNT | 8 | 0 | 3 | 1 | 11 | 9 | 88.89 | 72.73 | 80.00 | 36.36 |
| CONT | 6 | 2 | 3 | 1 | 11 | 9 | 66.67 | 54.55 | 60.00 | 54.55 |
| XTNT+CONT+TYPE | 21 | 3 | 9 | 3 | 33 | 27 | 77.78 | 63.64 | 70.00 | 45.45 |
| location | | | | | | | | | | |
| TYPE | 5 | 1 | 0 | 1 | 6 | 7 | 71.43 | 83.33 | 76.92 | 33.33 |
| XTNT | 6 | 0 | 0 | 1 | 6 | 7 | 85.71 | 100.00 | 92.31 | 16.67 |
| CONT | 6 | 0 | 0 | 1 | 6 | 7 | 85.71 | 100.00 | 92.31 | 16.67 |
| XTNT+CONT+TYPE | 17 | 1 | 0 | 3 | 18 | 21 | 80.95 | 94.44 | 87.18 | 22.22 |
| person | | | | | | | | | | |
| TYPE | 2 | 0 | 3 | 0 | 5 | 2 | 100.00 | 40.00 | 57.14 | 60.00 |
| XTNT | 2 | 0 | 3 | 0 | 5 | 2 | 100.00 | 40.00 | 57.14 | 60.00 |
| CONT | 0 | 2 | 3 | 0 | 5 | 2 | 0.00 | 0.00 | 0.00 | 100.00 |
| XTNT+CONT+TYPE | 4 | 2 | 9 | 0 | 15 | 6 | 66.67 | 26.67 | 38.10 | 73.33 |

Table 3.4: *The scores from comparing figure 3.7 with figure 3.8.*

| .001 | cor | inc | mis | spu | pos | act | pre | rec | f | ser |
|---|---|---|---|---|---|---|---|---|---|---|
| all_entities | | | | | | | | | | |
| TYPE | 1 | 0 | 0 | 0 | 1 | 1 | 100.00 | 100.00 | 100.00 | 0.00 |
| XTNT | 1 | 0 | 0 | 0 | 1 | 1 | 100.00 | 100.00 | 100.00 | 0.00 |
| CONT | 0 | 1 | 0 | 0 | 1 | 1 | 0.00 | 0.00 | 0.00 | 100.00 |
| XTNT+CONT+TYPE | 2 | 1 | 0 | 0 | 3 | 3 | 66.67 | 66.67 | 66.67 | 33.33 |
| enamex | | | | | | | | | | |
| TYPE | 1 | 0 | 0 | 0 | 1 | 1 | 100.00 | 100.00 | 100.00 | 0.00 |
| XTNT | 1 | 0 | 0 | 0 | 1 | 1 | 100.00 | 100.00 | 100.00 | 0.00 |
| CONT | 0 | 1 | 0 | 0 | 1 | 1 | 0.00 | 0.00 | 0.00 | 100.00 |
| XTNT+CONT+TYPE | 2 | 1 | 0 | 0 | 3 | 3 | 66.67 | 66.67 | 66.67 | 33.33 |
| person | | | | | | | | | | |
| TYPE | 1 | 0 | 0 | 0 | 1 | 1 | 100.00 | 100.00 | 100.00 | 0.00 |
| XTNT | 1 | 0 | 0 | 0 | 1 | 1 | 100.00 | 100.00 | 100.00 | 0.00 |
| CONT | 0 | 1 | 0 | 0 | 1 | 1 | 0.00 | 0.00 | 0.00 | 100.00 |
| XTNT+CONT+TYPE | 2 | 1 | 0 | 0 | 3 | 3 | 66.67 | 66.67 | 66.67 | 33.33 |

Table 3.5: *The scores from comparing the second paragraph of figure 3.7 with figure 3.8.*

| .002 | cor | inc | mis | spu | pos | act | pre | rec | f | ser |
|---|---|---|---|---|---|---|---|---|---|---|
| all_entities | | | | | | | | | | |
| TYPE | 4 | 0 | 3 | 1 | 7 | 5 | 80.00 | 57.14 | 66.67 | 57.14 |
| XTNT | 4 | 0 | 3 | 1 | 7 | 5 | 80.00 | 57.14 | 66.67 | 57.14 |
| CONT | 3 | 1 | 3 | 1 | 7 | 5 | 60.00 | 42.86 | 50.00 | 71.43 |
| XTNT+CONT+TYPE | 11 | 1 | 9 | 3 | 21 | 15 | 73.33 | 52.38 | 61.11 | 61.90 |
| enamex | | | | | | | | | | |
| TYPE | 4 | 0 | 3 | 1 | 7 | 5 | 80.00 | 57.14 | 66.67 | 57.14 |
| XTNT | 4 | 0 | 3 | 1 | 7 | 5 | 80.00 | 57.14 | 66.67 | 57.14 |
| CONT | 3 | 1 | 3 | 1 | 7 | 5 | 60.00 | 42.86 | 50.00 | 71.43 |
| XTNT+CONT+TYPE | 11 | 1 | 9 | 3 | 21 | 15 | 73.33 | 52.38 | 61.11 | 61.90 |
| location | | | | | | | | | | |
| TYPE | 3 | 0 | 0 | 1 | 3 | 4 | 75.00 | 100.00 | 85.71 | 33.33 |
| XTNT | 3 | 0 | 0 | 1 | 3 | 4 | 75.00 | 100.00 | 85.71 | 33.33 |
| CONT | 3 | 0 | 0 | 1 | 3 | 4 | 75.00 | 100.00 | 85.71 | 33.33 |
| XTNT+CONT+TYPE | 9 | 0 | 0 | 3 | 9 | 12 | 75.00 | 100.00 | 85.71 | 33.33 |
| person | | | | | | | | | | |
| TYPE | 1 | 0 | 3 | 0 | 4 | 1 | 100.00 | 25.00 | 40.00 | 75.00 |
| XTNT | 1 | 0 | 3 | 0 | 4 | 1 | 100.00 | 25.00 | 40.00 | 75.00 |
| CONT | 0 | 1 | 3 | 0 | 4 | 1 | 0.00 | 0.00 | 0.00 | 100.00 |
| XTNT+CONT+TYPE | 2 | 1 | 9 | 0 | 12 | 3 | 66.67 | 16.67 | 26.67 | 83.33 |
| .002 | cor | inc | mis | spu | pos | act | pre | rec | f | ser |

Table 3.6: *The scores from comparing the third paragraph of figure 3.7 with figure 3.8.*

| .003 | cor | inc | mis | spu | pos | act | pre | rec | f | ser |
|---|---|---|---|---|---|---|---|---|---|---|
| all_entities | | | | | | | | | | |
| TYPE | 2 | 1 | 0 | 0 | 3 | 3 | 66.67 | 66.67 | 66.67 | 33.33 |
| XTNT | 3 | 0 | 0 | 0 | 3 | 3 | 100.00 | 100.00 | 100.00 | 0.00 |
| CONT | 3 | 0 | 0 | 0 | 3 | 3 | 100.00 | 100.00 | 100.00 | 0.00 |
| enamex | | | | | | | | | | |
| TYPE | 2 | 1 | 0 | 0 | 3 | 3 | 66.67 | 66.67 | 66.67 | 33.33 |
| XTNT | 3 | 0 | 0 | 0 | 3 | 3 | 100.00 | 100.00 | 100.00 | 0.00 |
| CONT | 3 | 0 | 0 | 0 | 3 | 3 | 100.00 | 100.00 | 100.00 | 0.00 |
| XTNT+CONT+TYPE | 8 | 1 | 0 | 0 | 9 | 9 | 88.89 | 88.89 | 88.89 | 11.11 |
| location | | | | | | | | | | |
| TYPE | 2 | 1 | 0 | 0 | 3 | 3 | 66.67 | 66.67 | 66.67 | 33.33 |
| XTNT | 3 | 0 | 0 | 0 | 3 | 3 | 100.00 | 100.00 | 100.00 | 0.00 |
| CONT | 3 | 0 | 0 | 0 | 3 | 3 | 100.00 | 100.00 | 100.00 | 0.00 |
| XTNT+CONT+TYPE | 8 | 1 | 0 | 0 | 9 | 9 | 88.89 | 88.89 | 88.89 | 11.11 |

Table 3.7: *The scores from comparing the fourth paragraph of figure 3.7 with figure 3.8.*

ignores all speech recognition errors outside of named entity markup, since in this case the first paragraphs are non-identical.

The second paragraph contains one named entity. The data is collected from the relevant line within table 3.3.

ENAMEX cor cor inc PERSON PERSON "Nicodemus" "Nick"

The line tells us the entity being referred to - in this case "ENAMEX"; whether the type of entity in the hypothesis and the type of entity in the key matched - in this case "correct"; whether the boundaries of the named entity have not been crossed - in this case "correct"; whether the text within the named entity is the same in the hypothesis and the key - in this case "incorrect"; the type of named entity in the key - in this case "PERSON"; the type of named entity in the hypothesis - in this case also "PERSON"; the text within the markup in the key - in this case "Nicodemus"; and finally the text within the markup of the hypthesis - in this case "Nick". We are able to confirm that this line has been genuinely evaluated as the type (TYPE) of named entity is correct; a person has been labelled as a person. The extent (XTNT) of the named entity is correct; the named entity is one word long and has been labelled as one word long. The content (CONT) of the named entity is incorrect, however, Nick has been found in place of Nicodemus; there is therefore a content error. The respective precision, recall and F-measure for type, extent and content are calculated using equations 3.1, 3.2, 3.3 and 3.4.

It is important to note that the extent was marked correct because the boundaries occurred in a potentially correct place; the criteria for checking is that the words inside should not have occurred outside the boundaries - it does not involve a word count. Each of the following examples would therefore have been marked as correct boundaries because there are no forbidden words inside the named entity markup.

<ENAMEX TYPE="PERSON">Nick<ENAMEX> oh dean must ask

<ENAMEX TYPE="PERSON">Nick oh<ENAMEX> dean must ask

<ENAMEX TYPE="PERSON">Nick oh dean<ENAMEX> must ask

<ENAMEX TYPE="PERSON">Nick oh dean must<ENAMEX> ask

<ENAMEX TYPE="PERSON">Nick oh dean must ask<ENAMEX>

Both of the following examples would have been marked as incorrect, however, as far as extent is concerned, since in each case they contain a forbidden word.

<ENAMEX TYPE="PERSON">Be?' Nick<ENAMEX> oh dean must ask

<ENAMEX TYPE="PERSON">Nick asked<ENAMEX>

The software then uses this information to fill in table 3.5. For each line in table 3.3, there is a single count for each of TYPE, XTNT and CONT, which will occur in one of the columns cor, inc, mis or spu. Having found the individual scores for type, extent and content, the totals are found (referred to by the software as XTNT+CONT+TYPE). The respective precision, recall and F-measure for each row is then calculated. For this first paragraph there is only one named entity, and therefore the total for the type of named entity (PERSON), the total for the group of named entities (ENAMEX) and the total for all entities are identical. In general this is not the case, as can be seen in the next paragraph. When multiple different types of entity and groups of entity occur, the table becomes more specific deeper into the table.

The third paragraph contains a number of named entities; the scores are shown in table 3.6. Each named entity corresponds to one line from table 3.3. Addressing each line individually:

ENAMEX cor cor cor LOCATION LOCATION "Israel " "Israel ' s "

The first named entity is marked as correct. This is the only exception to the extent boundary which allows "'s" to be either inside or outside of the markup.

ENAMEX spu spu spu LOCATION "" "heaven"

An incorrect recognition of a word as a location results in a spurious error. All spurious errors incur a full penalty as it is not possible to have the correct word in spurious markup.

ENAMEX mis mis mis PERSON "Jesus " ""

Failing to identify an entity results in a similar error, and full penalty always results. In terms of F-score, the insertion errors and deletion errors carry the same overall penalty as each other.

ENAMEX cor cor cor LOCATION LOCATION "heaven" "heaven"

Totally correct markup. Scores correct for type, extent and content.

ENAMEX mis mis mis PERSON "Son of Man" ""

Another omission shows that the penalty is the same irrespective of length of the named entity.

ENAMEX cor cor inc PERSON PERSON "Moses" "Moe"

A further example of only falling short on the content. Had the hypothesised text read:

<ENAMEX TYPE="PERSON">Moses lifted<ENAMEX>

the type would have still been correct, but the extent and content would have been reversed. The content would have been correct because the word markup contains the desired "Moses". However the extent would have been wrong because the markup contains the word 'lifted'.

Finally, the scores for the fourth paragraph are shown in table 3.7, which shows how it is possible to obtain an error in the type of entity while still achieving the correct extent and content. If the word had been marked incorrectly:

<NUMEX TYPE="MONEY">world</NUMEX>,

rather than incurring a single type error, it would have incurred a full insertion error plus a full deletion error.

Having calculated scores for each of the paragraphs in the document, table 3.4 is produced by tallying up the individual section scores.

# Chapter 4

# Description of the standard statistical model

## 4.1 Introduction

In this chapter we introduce the standard statistical model that is used for the extraction of named entities from speech. A number of statistical attempts have been made at named entity recognition from speech (Robinson, Brown, Burger, Chinchor, Douthat, Ferro & Hirschman 1999). It is, however, notable how many of these have revolved around a single type of model, namely a Hidden Markov Model (HMM).

In the 1998 Hub-4 named entity evaluation, four sites submitted systems capable of named entity extraction from speech. These sites were:

- GTE Internetworking's BBN Technologies (BBN)
- A collaborative effort involving Cambridge University's Engineering Department, Sheffield University, and the International Computer Science Institute. (SPRACH)
- SRI International (SRI)
- A collaborative effort involving Boston University and MITRE Corporation. (MITRE)

With the exception of SPRACH, each site provided a single system for the extraction of named entities. BBN supplied a statistical system, SRI supplied a rule-based system, and MITRE supplied a statistical system. SPRACH supplied two separate systems: SPRACH-R a rule-based system and SPRACH-S a statistical system. All three statistical systems used as their basis what is referred to here as the standard statistical model.

In this chapter we introduce this model both graphically and mathematically. In the first part of this chapter we introduce the standard model that the BBN, MITRE and SPRACH-S systems have in common. In section 4.5 we describe the differences between the systems. In section 4.6 we describe our implementation of this standard model, which we later amend in subsequent chapters. Finally in section 4.7 we record our baseline experiment using the system described in section 4.6 which in turn is used for comparison purposes throughout the remainder of the thesis.

## 4.2 The mathematical approach

In the task of named entity extraction, finding which words correspond to named entities is equivalent to finding the sequence of named entities ($E_1^L$) corresponding to the sequence of words ($W_1^L$). We use a maximum likelihood approach to solve the problem. The task of named entity extraction is therefore formulated as finding the sequence of entities $E_1^L$ which maximises $P(E_1^L|W_1^L)$.

$$P(E_1^L|W_1^L) = \frac{P(E_1^L, W_1^L)}{P(W_1^L)} \tag{4.1}$$

Since $P(W_1^L)$ is constant for any given word sequence, without loss of generality, the task is to find the sequence of entities ($E_1^L$) which maximises $P(E_1^L, W_1^L)$.

$$\hat{E}_1^L = \arg\max_{E_1^L} P(E_1^L|W_1^L) = \arg\max_{E_1^L} P(E_1^L, W_1^L) \tag{4.2}$$

Although there are minor variations between systems, the problem is generally solved by splitting the joint probability of entities and words - $P(E_1^L, W_!^L)$ - into the product of the probability of the entities and the probability of the words given those entities using Bayes' theorem (Kim 2001).

The task of named entity extraction therefore becomes that of finding the entities which maximise the product of the component probabilities.

$$P(E_1^L, W_1^L) = P(E_1^L).P(W_1^L|E_1^L) \tag{4.3}$$

It is possible to break down further the component probabilities of equation 4.3 thus:

$$\begin{aligned} P(E_1^L) &= P(e_1).P(e_2|e_1).P(e_3|e_1,e_2).....P(e_L|E_1^{L-1}) \\ &= \prod_{i=1}^{L} P(e_i|E_1^{i-1}) \end{aligned} \tag{4.4}$$

$$\begin{aligned} P(W_1^L|E_1^L) &= P(w_1|E_1^L).P(w_2|w_1,E_1^L).P(w_3|w_1,w_2,E_1^L).....P(w_L|W_1^{L-1},E_1^L) \\ &= \prod_{i=1}^{L} P(w_i|W_1^{i-1},E_1^L) \end{aligned} \tag{4.5}$$

These probabilities can be estimated from data, provided certain independence assumptions are made. It is apparent that $P(w_L|W_1^{L-1}, E_1^L)$ cannot be estimated from data without making some independence assumptions since it is not possible for the training data to contain every possible sequence $W_1^L$, $E_1^L$ $\forall L$. If, however, independence assumptions are made (for example independence over $W_1^{i-2}$ and $E_1^L$), it is not unreasonable to estimate from data $P(w_i|w_{i-1})$.

Kim details these standard assumptions (Kim 2001): equation 4.6 uses a bigram assumption, where independance over $E_1^{i-2}$ is assumed; similarly equation 4.7 uses a crude approximation, where independence over $W_1^{i-1}$, $E_1^{i-1}$ and $E_{i+1}^L$ is assumed.

$$\begin{aligned} P(E_1^L) &= P(e_1).P(e_2|e_1).P(e_3|e_1,e_2).....P(e_L|E_1^{L-1}) \\ &\simeq e_1 \prod_{i=2}^{L} P(e_i|e_{i-1}) \end{aligned} \tag{4.6}$$

$$\begin{aligned} P(W_1^L|E_1^L) &= P(w_1|E_1^L).P(w_2|w_1,E_1^L).P(w_3|w_1,w_2,E_1^L).....P(w_L|W_1^{L-1},E_1^L) \\ &\simeq \prod_{i=1}^{L} P(w_i|e_i) \end{aligned} \tag{4.7}$$

It is concluded that equation 4.3 may be approximated by equation 4.8 and hence the named entity sequence is found by finding the sequence of entities ($E_1^L$) which maximise the right hand side of equation 4.8

| $W_1^6$ | $w_1$ peter | $w_2$ attended | $w_3$ a | $w_4$ conference | $w_5$ in | $w_6$ mexico |
|---|---|---|---|---|---|---|
| $E_1^6$ | $e_1$ <PERSON> | $e_2$ not-an-entity | $e_3$ not-an-entity | $e_4$ not-an-entity | $e_5$ not-an-entity | $e_6$ <PLACE> |

Table 4.1: *The phrase 'peter attended a conference in mexico' in mathematical notation.*

$$P(E_1^L, W_1^L) \simeq \prod_{i=1}^{L} P(e_i|e_{i-1}). \prod_{i=1}^{L} P(w_i|e_i) \tag{4.8}$$

In practice, both the MITRE (Palmer et al. 1999) and BBN (Bikel, Schwartz & Weischedel 1999) systems made fewer independence assumptions than those of equation 4.7 and used $P(w_i|w_{i-1}, e_i)$, rather than $P(w_i|e_i)$ as in equation 4.7. This is discussed later in this chapter.

### 4.2.1 Example

Having detailed the mathematical theory above, we now work through an example:

'peter attended a conference in mexico'

The correct markup for this text is:

'<PERSON>peter</PERSON> attended a conference in <PLACE>mexico</PLACE>'

In the mathematical notation described above, this phrase would correspond to table 4.1, which also has the correct markup associated with it. The probabilities corresponding to this entity sequence are given in table 4.2. For the statistical model to correctly mark up the text, the product of the sequence of probabilities shown in table 4.2 is required to be the largest of any possible sequence of entities - that is, that there is no sequence of entities ($E_1^6$) for which $\prod_{i=1}^{6} P(e_i|e_{i-1}). \prod_{i=1}^{6} P(w_i|e_i)$ is higher than for the sequence given in table 4.1.

| $\prod_{i=1}^{L} P(e_i \vert e_{i-1})$ | | $\prod_{i=1}^{L} P(w_i \vert e_i)$ | |
|---|---|---|---|
| P($e_1$) | P(<PERSON>) | P($w_1 \vert e_1$) | P(peter\| <PERSON>) |
| P($e_2 \vert e_1$) | P(not-an-entity\| <PERSON>) | P($w_2 \vert e_2$) | P(attended\|not-an-entity) |
| P($e_3 \vert e_2$) | P(not-an-entity\|not-an-entity) | P($w_3 \vert e_3$) | P(a\|not-an-entity) |
| P($e_4 \vert e_3$) | P(not-an-entity\|not-an-entity) | P($w_4 \vert e_4$) | P(conference\|not-an-entity) |
| P($e_5 \vert e_4$) | P(not-an-entity\|not-an-entity) | P($w_5 \vert e_5$) | P(in\|not-an-entity |
| P($e_6 \vert e_5$) | P(<PLACE> \|not-an-entity) | P($w_6 \vert e_6$) | P(mexico\| <PLACE>) |

Table 4.2: *The probabilities associated with the correct markup of the sentence 'peter attended a conference in mexico'.*

## 4.3 The graphical approach

We have in the previous section given mathematical justification for named entity extraction being equivalent to solving equation 4.9.

$$\hat{E}_1^L = \arg\max_{E_1^L} \prod_{i=1}^{L} P(e_i|e_{i-1}) . \prod_{i=1}^{L} P(w_i|e_i) \tag{4.9}$$

In this section we show how this is equivalent to a finite state machine. The method of solving equation 4.9 is that of a Viterbi search through the finite state machine in much the same way as a standard Hidden Markov Model (HMM) functions in speech recognition. Initially we introduce the simplest finite state automaton capable of generating our example and then we extend this model to deal with the general case of any sequence of words.

Figure 4.1 shows a concise finite state machine (FSM) which would allow the generation of the example phrase "<PERSON>peter</PERSON> attended a conference in <PLACE>mexico</PLACE>". To understand this finite state machine we consider transitions between states to be associated with the probabilities of the entities, i.e. transitions correspond to $P(e_i|e_{i-1})$; whereas the states themselves represent entities and are where the words are generated - thus states compute $P(w_i|e_i)$.

In figure 4.2 we make a few minor improvements to the FSM of figure 4.1. Although figure 4.1 would generate the phrase "<PERSON>peter</PERSON> attended

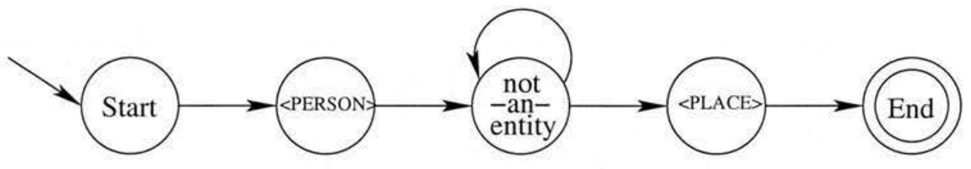


Figure 4.1: *A simple FSM which allows the generation of "<PERSON>peter</PERSON> attended a conference in <PLACE>mexico</PLACE>". Transitions represent P(entity|context); states represent named entities - which generate the corresponding words.*

a conference in <PLACE>mexico</PLACE>", it would be unable to generate the phrase "<PERSON> peter irvine</PERSON> attended a conference in <PLACE>mexico city</PLACE>". The changes to accept this sentence are clearly trivial (adding self-transitions to both the <PERSON> state and also the <PLACE> state). The other main difference is in the layout of the FSM, which otherwise is identical to that of figure 4.1.

By extending the finite state machine of figure 4.1 to figure 4.2 we have made one important change. The new model is no longer deterministic with respect to the input sequence. The FSM in figure 4.1 was deterministic and had only one possible outcome for strings of length 3 or more; the first word would be a person and the last word would be a place. The FSM in figure 4.2, although deterministic when the input string contains only 3 words, is not deterministic if the input is longer than 3 words. The final decision between which path to take can be found by a Viterbi search through the FSM.

There is still a fundamental flaw in the automaton shown in figure 4.2; this automaton only allows phrases of the form <PERSON> (<PERSON>)*not-an-entity(not-an-entity)*<PLACES> (<PLACES>)*. We, however, require the ability to generate phrases of the form ((<PERSON>)*(not-an-entity)*(<PLACES>)*...)*. This is because it is clearly the case that not all word sequences start with a person and end with a place. Our model should be capable of generating "mexico has had a huge tourist trade a friend was there last year peter", and indeed any sequence of entities must be possible. Expanding to such a model is again trivial. All that is required is to add transitions from the end of all states (excluding the final end state) to the start of all other states (excluding the initial start state). This fully connected FSM is shown in figure 4.3.

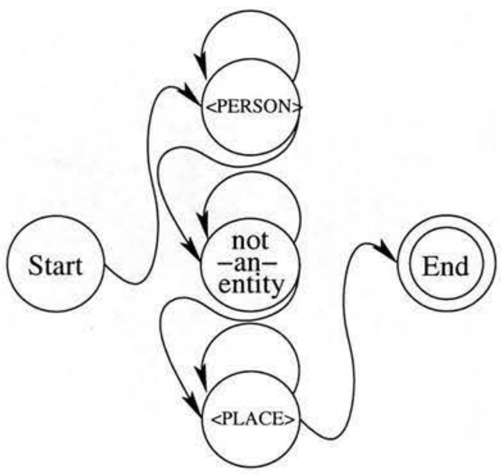


Figure 4.2: *Improved FSM which allows the generation of both "<PERSON>peter</PERSON> attended a conference in <PLACE>mexico</PLACE>" and "<PERSON>peter irvine</PERSON> attended a conference in <PLACE>mexico city</PLACE>".*

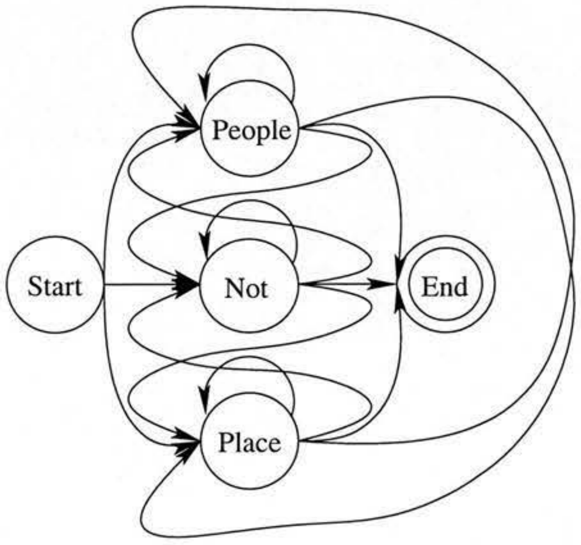


Figure 4.3: *Fully connected FSM which allows the generation of all word sequences containing people and places.*

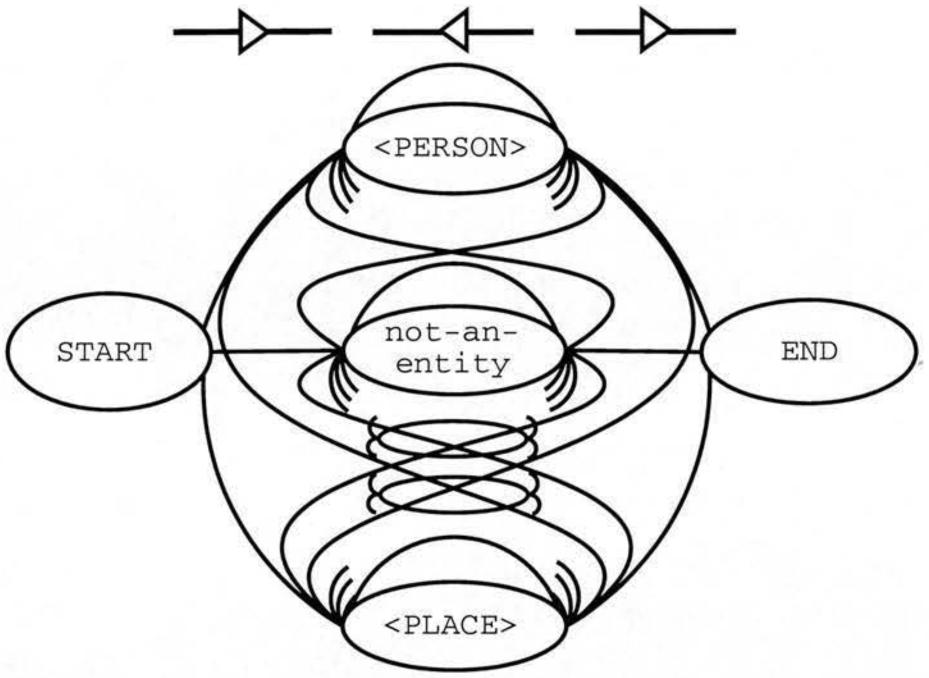


Figure 4.4: *Final topology used in the standard statistical model for the extraction of named entities. This topology allows for the markup of all sequences of words where words can either be from the same named entity class as the previous or in any of the alternative named entity classes. The arrows above each transition correspond to the direction that the transition takes.*

In order to produce the full FSM required, states for each of the named entities need to be created rather than just for people and places. Figure 4.4 is a simplified version of this final FSM topology. It is simplified to show only connections between the start state, the <PERSON> state, the not-an-entity state, the <PLACE> state, and the end state to avoid confusion caused by 8x9+8=80 (number of entity states multiplied by the number of input transitions to them plus the number of input transitions to the end state) transitions. Additional states have been indicated with fainter lines as have some starts and ends of transitions to and from these states. To further aid clarity the arrows on each transition have been removed from the transitions and three arrows showing the direction of transitions added physically allow them in the diagram - essentially the direction of the arrow runs from the output (right hand) side of a state to the input (left hand) side of a state.

in a turn of events <ENAMEX TYPE='PERSON'>john gillan</ENAMEX> originally from <ENAMEX TYPE='LOCATION'>northern ireland</ENAMEX> was recently involved ...

Figure 4.5: *Example of some marked up text.*

Although the models of both figure 4.2 and figure 4.4 are non-deterministic, it is apparent that the Viterbi search space is considerably larger for figure 4.4. More precisely there are $\frac{(L-1)(L-2)}{2}$ possible solutions to $W_1^L$ in figure 4.2, but there are $8^L$ possible solutions to figure 4.4.

## 4.4 Generating Markup

In chapter 1 we introduced the type of markup that is required as output from the named entity extraction system. Figure 4.5 shows an example of correctly marked up output.

The model, described in sections 4.2 and 4.3, produces output of the form $W_1^L, P_1^L$; ie the output corresponding to the text in figure 4.5 would be as shown in figure 4.6.

The specific systems use different methods for generating the markup. In general, however, the method is to insert the particular xml sequence which corresponds to the start of the entity (e.g. <NUMEX TYPE='MONEY'> for monetry expressions) wherever $e_i \neq e_{i-1}$; and to insert the particular xml sequence[1] which corresponds to the end of the entity (e.g. </TIMEX> for dates) wherever $e_i \neq e_{i+1}$.

This may be done efficiently at process time, rather than post-processing, by adding the start of entity markup every time a token is passed into a new state, and end of entity every time a token is passed out of any state. Markup is not generated within any state.

[1]the particular xml sequence which corresponds to both the start and end of the not-an-entity entity is the null or empty string ''.

| $W_1^{14}$ | | $E_1^{14}$ | |
|---|---|---|---|
| $w_1$ | in | $e_1$ | NOT–AN–ENTITY |
| $w_2$ | a | $e_2$ | NOT–AN–ENTITY |
| $w_3$ | turn | $e_3$ | NOT–AN–ENTITY |
| $w_4$ | of | $e_4$ | NOT–AN–ENTITY |
| $w_5$ | events | $e_5$ | NOT–AN–ENTITY |
| $w_6$ | john | $e_6$ | <PERSON> |
| $w_7$ | gillan | $e_7$ | <PERSON> |
| $w_8$ | originally | $e_8$ | NOT–AN–ENTITY |
| $w_9$ | from | $e_9$ | NOT–AN–ENTITY |
| $w_{10}$ | northern | $e_{10}$ | <PLACE> |
| $w_{11}$ | ireland | $e_{11}$ | <PLACE> |
| $w_{12}$ | was | $e_{12}$ | NOT–AN–ENTITY |
| $w_{13}$ | recently | $e_{13}$ | NOT–AN–ENTITY |
| $w_{14}$ | involved | $e_{14}$ | NOT–AN–ENTITY |

Figure 4.6: *Example of the system output corresponding to figure 4.5.*

## 4.5 Specific Systems

We now introduce the specific systems used in the IE-NE subtask of HUB 4 in 1998. The most simple, and correspondingly the worst performing, statistical system implemented for the HUB 4 task was SPRACH-S described in (Gotoh & Renals 1999) based on the mathematics detailed in (Gotoh, Renals & Williams 1999). The results of this system are presented in (Renals et al. 1999).

This system used equation 4.10 to solve for the best word and entity path. There are a few minor differences between this and equation 4.9. The first and most important is the difference of notation. SPRACH-S uses $t$ to refer to entities (tags) and $e$ to refer to combinations of words and tags. More explicitly a unique tag-word token $e$ is defined in 4.11.

$$\hat{T,W} = \arg\max_{T,W} f(e|e_1^{i-1}).f(w|e) \tag{4.10}$$

$$e_i = \left\{ \begin{array}{cc} <t,w>_i & if <t,w>_i \, \epsilon \, vocabulary \\ t_i & otherwise \end{array} \right\} \tag{4.11}$$

The advantage of the definition of $e$ in 4.11 is that it provides a means of estimating the probability of infrequently occurring words (and words that did not occur at all) within certain entities. No mathematical justification for this smoothing is given. The danger of smoothing lies in the information which is discarded, or, equivalently, the information which is assumed.

The system developed by MITRE obtained scores comparable to (though slightly lower than) that developed by BBN. We address them in that order. The MITRE system (Palmer, Ostendorf & Burger 2000) decomposes the probabilities in the fairly traditional method into state transition probabilities (the probabilities of entities given the context) and state dependent models (the probabilities of words given the context). What makes this system unique is the use of class-based smoothing, over classes $c_k$, in the estimation of probabilities. The principles described in (Iyer & Ostendorf 1997), namely equation 4.12, are used with automatically transcribed part of speech (POS) tags used as the classes. For textual data, other classes (such as capitalised word) were used, but for speech, since this information is not available, POS alone was used. Results of this system are presented in (Palmer et al. 1999).

$$P(w_i|w_{i-1}, e_i) = \sum_k P(w_i|w_{i-1}, c_k, e_i)P(c_k|w_{i-1}, e_i) \tag{4.12}$$

Finally in this section we deal with BBN (Miller et al. 1999), whose complete system is detailed in (Bikel et al. 1999). This was the highest scoring system (it produced the best F-measure) in the 1998 HUB 4 IE-NE task. This system was very similar to the MITRE system detailed above.

This system was able to deal with the "Simi Valley California" problem[2] by using separate statistical models for dealing with the first word in each entity and with subsequent words in each entity.

In terms of the FSM this equates to having two separate states for each named entity. The first of these states allows transitions from all other states, whereas the second state has only two possible transitions into it - from itself or from the first state. The BBN system therefore splits $P(w_i|W_1^{i-1}, E_1^L)$ into two cases: (i) The case of the first word of an entity in which case the approximation $P(w_i|e_i, e_{i-1})$ is used, and (ii) the case of non-first word of an entity when $P(w_i|w_{i-1}, e_i)$ is used. No mathematical justification is given for this, although an intuitive argument is given in (Bikel et al. 1999).

The use of the two separate states enables more advanced markup generation. Essentially markup is still generated in the same way as described in section 4.4. Now, however, it is possible to enter the second state without generating markup (equivalent to a self-transition previously) or to re-enter the first state generating markup - thus facilitating the correct markup of Simi Valley California.

The second important feature of this model is the back-off strategy. BBN have a sophisticated back-off strategy for all the probabilities that require estimating. The back-off strategy involves the use of word features. Details of the features used are given in (Bikel et al. 1999), together with an ordering for specifying priority of one feature

[2]The "Simi Valley California" problem is the difficulty of giving "Simi Valley California" the correct markup. (The same applies to numerous similar phrases). The problem stems from the fact that labelling the phrase as a single location named entity is wrong, but equally marking up the individual words as individual location named entities is also wrong. See (Gotoh & Renals 1999) for further details.

| Information | First-word Predictions | Non-first-word Predictions |
|---|---|---|
| Most known | $P(< w, f >_i \|e_i, e_{i-1})$ | $P(< w, f >_i \| < w, f >_{i-1}, e_i)$ |
| | $P(< w, f >_i \|e_i, < Not - Entity >)$ | $P(< w, f >_i \|e_i)$ |
| | $P(< w, f >_i \|e_i)$ | $P(w\|e_i).P(f\|e_i)$ |
| | $P(w\|e_i).P(f\|e_i)$ | $\frac{1}{\|V\|} \cdot \frac{1}{14}$ |
| Nothing known | $\frac{1}{\|V\|} \cdot \frac{1}{14}$ | |

Table 4.3: *The back-off strategy used by BBN to estimate the probability of words given context.*

over another.

BBN group words and features together in a similar way to SPRACH-S and then estimate $P(< w, f >_i |e_i, e_{i-1})$ and $P(< w, f >_i \mid < w, f >_{i-1}, e_i)$. The advantage of this added complexity is that it provides flexibility for the back-off strategy given in table 4.3.

## 4.6 Implementing a named entity recogniser

Having decribed systems based on this standard model, we now introduce our implementation of this Markov model approach. Initially we explain the theory behind our system and then briefly cover the technical details.

### 4.6.1 The theory

Equation 4.9 has shown that there are two distinct probabilities that need to be calculated, namely $P(w_i|e_i)$ and $P(e_i|e_{i-1})$. The methods for calculating each of these within the named entity recognition system are dealt with individually.

#### Calculating $P(w_i|e_i)$

In order to estimate $P(w_i|e_i)$, language models were created using the Carnegie Mellon University (CMU) statistical language modelling toolkit (Rosenfeld 1994). It was necessary for the training data to be reformatted into a form suitable for eight separate

!SENT_START this is <ENAMEX TYPE="ORGANIZATION">a. b. c. news</ENAMEX>
it's <NUMEX TYPE="DATE"/>tuesday the fifth of january</NUMEX> i'm
<ENAMEX TYPE="PERSON">ted kopel</ENAMEX> reporting from
<ENAMEX TYPE="LOCATION>new york city</ENAMEX> !SENT_END

Figure 4.7: *A example of the Training Key.*

language models to be trained on entity-specific data. An example of the Training Key in its original format is shown in figure 4.7.

This data needed to be converted to produce training material suitable for single entity language models. To do this, all named entity markup and the corresponding named entities were replaced within the text by an appropriate pseudo-word[3]. The remaining data was suitable for training the not-an-entity language model. The not-an-entity data correspondingly becomes:

!SENT_START this is <org/> it's <dat/> i'm <peo/> reporting from <pla/> !SENT_END

The training data for each respective entity language model is then created by treating each named entity as a sentence within the training data of that model. Each sentence uses <s> and </s> to delimit the start and end of the sentence. Taking the example set out in figure 4.7, we now add one sentence to each of: the organisation training data, the date training data, the person training data, and the place training data. These sentence are respectively as follows:

<s> a. b. c. news </s> - in the organisation data

<s> tuesday the fifth of january </s> - in the date data

[3]<s>, </s>, <org/>, <dat/>, <peo/>, <pla/>, and also <tim/>, <mon/>, and <per/> are pseudo-words; that is, they are not part of the original training material but are rather words that are added to the training material so that probabilities associated with these pseudo-words can be estimated. Standard XML style notation is used so that the document remains xml compliant; and the removal of the XML tags results in the words of the original document. These pseudo-words represent start of string tag, end of string tag, organisation tag, date tag, people tag, place tag, time tag, monetry expression tag and percentage tag respectively. The purpose of the tags is to allow, for example: the language models trained on the date data to predict $P(the| < s >, tuesday)$, and the language trained on the organisation data to predict $P(< /s > |c., news)$.

<s> ted kopel </s> - in the person data

<s> new york city </s> - in the place data

No data is added to percentage, monetary-expression, or time training data as there was no information about percentages, times are money within the original sentence.

The new data was used by the CMU toolkit to create eight individual language models. These were in turn used to generate the probabilities of words (ie the $P(w_i|e_i)$ from equation 4.9). Trigram language models were created, rather than simply unigram language models, which provided us with a means of generating probabilities of the form $P(w_i|e_i, W_{i-2}^{i-1})$, thereby enabling us to make fewer independence assumptions than Kim.

Not all required probabilities will exist within these eight language models. In some instances, named entity specific language models will not contain the full trigram probabilities. This is due in part to the sparseness of the data, and in part to the fact that some sequences cannot occur (for example, the word "James" will not occur in the percentage language model; nor would $P(sleep|green, ideas)$ occur in any language model). In these instances the language models use a standard back-off procedure, where the back-off weights were calculated by the CMU toolkit at the time of training. The language models also deal with out of vocabulary (OOV) words at this time.

The disadvantage of training by this method is that there is would be no accurate way of predicting the probability of a word in a new entity given the word in the previous entity, since the training data does not contain the words from previous entities. For example, the organization language model will not be able to predict the $P(a.|this\ is, organization)$ - the language model simply calculates the approximate probability based on the contextual word <s>, so the language model simply predicts the probability of $P(a.| < s >, organisation)$.

## Calculating $P(e_i|e_{i-1})$

A transition matrix of probabilities of transitioning between named entities was also constructed. To produce this transition matrix a count of transitions was taken for a pass through the training data. Using the data above we find that the first transition is for "!SENT_START" and "that" and the corresponding states are start and not-an-entity; the first word-word transition is for the words "this" and "is" where the

| From \ To | start | not-an-entity | location | person | organization | time | date | percentage | money | end |
|---|---|---|---|---|---|---|---|---|---|---|
| start | - | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |
| not-an-entity | - | 2 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| location | - | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| person | - | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| organization | - | 1 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| time | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| date | - | 1 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| percentage | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| money | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| end | 0 | - | - | - | - | - | - | - | - | - |

Table 4.4: *Frequency counts for named entity state transitions.*

| From \ To | start | not-an-entity | location | person | organization | time | date | percentage | money | end |
|---|---|---|---|---|---|---|---|---|---|---|
| start | - | $\frac{1}{21} = 0.048$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |
| not-an-entity | - | 0.095 | 0.048 | 0.048 | 0.048 | 0 | 0.048 | 0 | 0 | 0.048 |
| location | - | 0.048 | 0.095 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| person | - | 0.048 | 0 | 0.048 | 0 | 0 | 0 | 0 | 0 | 0 |
| organization | - | 0.048 | 0 | 0 | 0.143 | 0 | 0 | 0 | 0 | 0 |
| time | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| date | - | 0.048 | 0 | 0 | 0 | 0 | 0.190 | 0 | 0 | 0 |
| percentage | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| money | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| end | 0 | - | - | - | - | - | - | - | - | - |

Table 4.5: *Transition Matrix for named entity state transitions.*

transition is from the not-an-entity entity to the not-an-entity entity; the second word-word transition is from the not-an-entity entity to the organization entity, the third is from the organization entity to the organization entity, and so on. Table 4.4 shows the complete counts for all transitions for the example text.

From the count matrix, the transition matrix is calculated by simply dividing each figure by the grand total of all counts. The actual transition matrix used for the named entity extraction system based on table 4.4 is shown in table 4.5. In practice when counting the transitions of the million-word training data, all counts were increased by 1 to ensure that no theoretically possible transitions were completely ruled out due to lack of evidence in training data.

The named entity recogniser used the probabilities from the transition matrix to predict the $P(e_i|e_{i-1})$ from equation 4.9.

### 4.6.2 The technical details

The named entity recogniser was built in C++, and relies heavily on the use of library classes from the Edinburgh Speech Tools[4] (Taylor, Caley, Black & King 1999). There is also a small dependence on library classes from Espresso (King, Frankel & Richmond 2003) which is currently not available for public use.

Our statistical named entity recogniser uses the token passing algorithm (Young, Russell & Thornton 1989) to pass tokens around ten states corresponding to a start state, an end state, a state for each of the named entities, and a state for not-an-entity. Each token stores a triplet of information: the word that it represents, the total probability associated with the token being in that state (accumulated as the token has been passed from previous states), and a pointer back to where the token came from. By examining any token, it is possible to establish its current condition; that is, the word to which the token corresponds and the entity that the word is classified as (known because the state itself is known). It is also possible to trace the entire history of the token by considering the link back to its preceeding condition, to the condition before that, and so on right back to the start state.

Each of the named entity states, together with the not-an-entity state, house a named entity specific language model trained as per section 4.6.1 which is used to predict the probability of the word occurring within that state. Consequently, when a language model predicts the probability of a word occurring within the state, this corresponds mathematically to the probability of the word given the entity that the state represents. The language models are standard back-off trigram language models, which are trained using the CMU toolkit with its default settings.[5]

In our implementation of the named entity recogniser, as tokens pass along transitions between states, these transitions are weighted with the respective probabilities from the transition matrix, and the token's probability is updated to reflect this transition ($P(e_i|e_{i-1})$). Once the token reaches the new state, the language model within the state generates the probability of the word given the state, and the token's probability

[4] Freely available on the internet at http://www.cstr.ed.ac.uk/projects/festival/download.html.

[5] Witten Bell discounting was used rather than the default linear discounting method after a comparision revealed better results using this discounting strategy.

```
solo artist neil alton was yesterday reunited with fellow former nervous
passenger band members jamie wilson and stuart cockburn to ...
```

Figure 4.8: *A piece of text to be classified by the named entity extraction system*

```
solo artist <ENAMEX TYPE=''PERSON''>neil alton<ENAMEX> was yesterday
reunited with fellow former <ENAMEX TYPE=''ORGANIZATION''>nervous
passenger<ENAMEX> band members <ENAMEX TYPE=''PERSON''>jamie
wilson<ENAMEX> and <ENAMEX TYPE=''PERSON''>stuart cockburn<ENAMEX>
to ...
```

Figure 4.9: *A correctly classified piece of text*

is again updated to reflect this $(P(w_i|e_i, W_{i-2}^{i-1}))$.

By the time a token reaches the end state, the probability stored in the token has effectively accumulated $\prod_{i=1}^{L} P(e_i|e_{i-1}.P(w_i|e_i, W_{i-2}^{i-1})$. By comparing the probabilities of all tokens at the end state, it is possible to find the token which is most likely, and then, by examination of this token, it is possible to find the path that the token has taken as described above. Our system therefore finds the arguments which maximise $\prod_{i=1}^{L} P(e_i|e_{i-1}.P(w_i|e_i, W_{i-2}^{i-1})$ and solves the problem defined in equation 4.9.

### 4.6.3 Example

In order to named entity classify the text shown in figure 4.8 and correctly produce the text shown in figure 4.9 we need to calculate the named entity sequence $E_1^L$ that minimises the overall probability $\prod_{i=1}^{L} P(e_i|e_{i-1}). \prod_{i=1}^{L} P(w_i|e_i)$. The model described above in section 4.6.1 enables us to calculate each individual probability and therefore the combined probability.

If we consider the input document (figure 4.8), the output document (figure 4.9) and the simplified model topology - showing only the states for person, not-an-entity, and organisation (figure 4.10), we are able to infer the path through the FSM that needed to be taken by the winning token. Using the labels on figure 4.9, the winning token would have taken route EDBACDDDDDGHFDBACBAC...

Each letter on the route refers to a transition probability calculated from the transi-

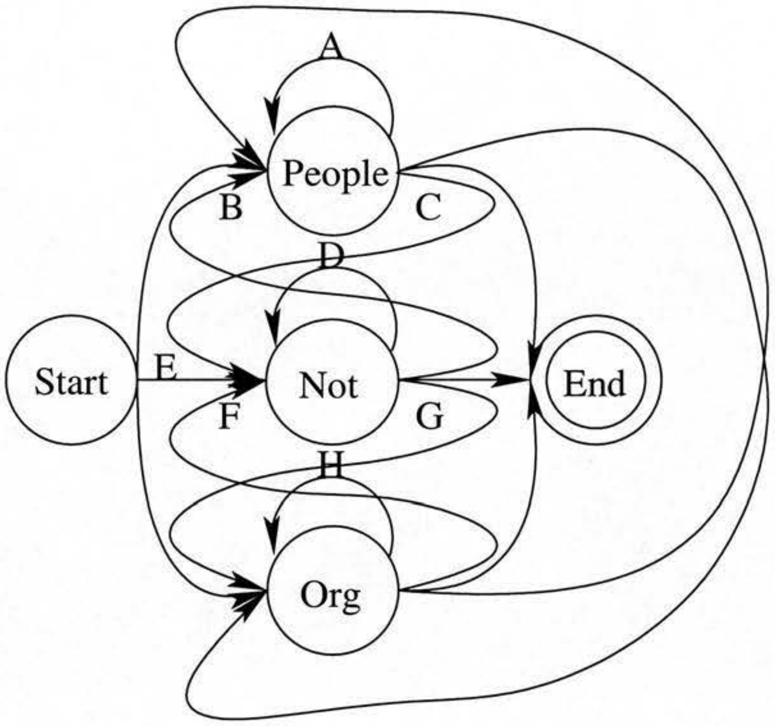


Figure 4.10: *Simplified model topology - showing only the states for person, not-an-entity, and organization.*

| Letter | Probability |
|---|---|
| A | $\mathrm{P}(e_i = \text{people} \mid e_{i-1} = \text{people})$ |
| B | $\mathrm{P}(e_i = \text{people} \mid e_{i-1} = \text{not-an-entity})$ |
| C | $\mathrm{P}(e_i = \text{not-an-entity} \mid e_{i-1} = \text{people})$ |
| D | $\mathrm{P}(e_i = \text{not-an-entity} \mid e_{i-1} = \text{not-an-entity})$ |
| E | $\mathrm{P}(e_i = \text{not-an-entity} \mid e_{i-1} = \text{start})$ |
| F | $\mathrm{P}(e_i = \text{not-an-entity} \mid e_{i-1} = \text{organization})$ |
| G | $\mathrm{P}(e_i = \text{organization} \mid e_{i-1} = \text{not-an-entity})$ |
| H | $\mathrm{P}(e_i = \text{organization} \mid e_{i-1} = \text{organization})$ |

Table 4.6: *Key to the letters on figure 4.10.*

tion matrix according to table 4.6 which enables the calculation of $\prod_{i=1}^{L} P(e_i|e_{i-1})$ from equation 4.9.

The word probabilities are calculated by the state-internal language models dependent on where the token is when words are generated. Table 4.7 summarizes which probabilities are calculated in which state-internal language model. $\prod_{i=1}^{l} P(w_i|e_i, W_{i-2}^{i-1})$ is calculated by taking the product of the probabilities within table 4.7.

In addition to the correct path through the FSM there will be $8^L - 1 = 8^{20} - 1 > 10^{18}$ incorrect paths through this FSM, using a Viterbi search enables an efficient search of these paths. The path with the maximum combined probability is chosen - hopefully the path which results in figure 4.9.

## 4.7 Baseline experiment

As a baseline for later work in this thesis, an experiment was conducted using the system described in section 4.6 with the language models trained on the complete set (almost one million words) of training data described in chapter 3. The language models were trained using the Witten Bell discounting strategy - the discounting strategy that produced the highest results, and the system run at optimum configuration for the development data.

The results of this experiment produced an F-score of 79.8. The full output file is shown in table 4.8.

| Word | State | Probability calculated |
|---|---|---|
| !SENT_START | start | - |
| Solo | not-an-entity | $P(Solo\|!SENT_START)$ |
| artist | not-an-entity | $P(artist\|!SENT_START, Solo)$ |
| Neil | people | $P(Neil\| < s >)$ |
| Alton | people | $P(Alton\| < s >, Neil)$ |
| was | not-an-entity | $P(was\| < peo/ >$ |
| yesterday | not-an-entity | $P(yesterday\| < peo/ >, was)$ |
| reunited | not-an-entity | $P(reunited\|was, yesterday)$ |
| with | not-an-entity | $P(with\|yesterday, reunited)$ |
| fellow | not-an-entity | $P(fellow\|reunited, with)$ |
| former | not-an-entity | $P(former\|with, fellow)$ |
| Nervous | organization | $P(Nervous\| < s >$ |
| Passenger | organization | $P(Passenger\| < s >, Nervous)$ |
| band | not-an-entity | $P(band\| < org/ >$ |
| members | not-an-entity | $P(members\| < org/ >, band)$ |
| Jamie | people | $P(Jamie\| < s >)$ |
| Wilson | people | $P(Wilson\| < s >, Jamie)$ |
| and | not-an-entity | $P(and\| < peo/ >$ |
| Stuart | people | $P(Stuart\| < s >)$ |
| Cockburn | people | $P(Cockburn\| < s >, Stuart)$ |
| to | not-an-entity | $P(to\| < /peo >$ |
| ... | ... | ... |

Table 4.7: *The word probabilities calculated by the state-internal language models.*

| all_stories | cor | inc | mis | spu | pos | act | pre | rec | f | ser |
|---|---|---|---|---|---|---|---|---|---|---|
| all_entities | | | | | | | | | | |
| TYPE | 1419 | 52 | 434 | 120 | 1905 | 1591 | 89.19 | 74.49 | 81.18 | 31.81 |
| XTNT | 1299 | 172 | 434 | 120 | 1905 | 1591 | 81.65 | 68.19 | 74.31 | 38.11 |
| CONT | 1468 | 3 | 434 | 120 | 1905 | 1591 | 92.27 | 77.06 | 83.98 | 29.24 |
| XTNT+CONT+TYPE | 4186 | 227 | 1302 | 360 | 5715 | 4773 | 87.70 | 73.25 | 79.82 | 33.05 |
| enamex | | | | | | | | | | |
| TYPE | 1247 | 52 | 395 | 88 | 1694 | 1387 | 89.91 | 73.61 | 80.95 | 31.58 |
| XTNT | 1135 | 164 | 395 | 88 | 1694 | 1387 | 81.83 | 67.00 | 73.68 | 38.19 |
| CONT | 1296 | 3 | 395 | 88 | 1694 | 1387 | 93.44 | 76.51 | 84.13 | 28.69 |
| XTNT+CONT+TYPE | 3678 | 219 | 1185 | 264 | 5082 | 4161 | 88.39 | 72.37 | 79.58 | 32.82 |
| numex | | | | | | | | | | |
| TYPE | 66 | 0 | 7 | 23 | 73 | 89 | 74.16 | 90.41 | 81.48 | 41.10 |
| XTNT | 62 | 4 | 7 | 23 | 73 | 89 | 69.66 | 84.93 | 76.54 | 46.58 |
| CONT | 66 | 0 | 7 | 23 | 73 | 89 | 74.16 | 90.41 | 81.48 | 41.10 |
| XTNT+CONT+TYPE | 194 | 4 | 21 | 69 | 219 | 267 | 72.66 | 88.58 | 79.84 | 42.92 |
| timex | | | | | | | | | | |
| TYPE | 106 | 0 | 32 | 9 | 138 | 115 | 92.17 | 76.81 | 83.79 | 29.71 |
| XTNT | 102 | 4 | 32 | 9 | 138 | 115 | 88.70 | 73.91 | 80.63 | 32.61 |
| CONT | 106 | 0 | 32 | 9 | 138 | 115 | 92.17 | 76.81 | 83.79 | 29.71 |
| XTNT+CONT+TYPE | 314 | 4 | 96 | 27 | 414 | 345 | 91.01 | 75.85 | 82.74 | 30.68 |
| date | | | | | | | | | | |
| TYPE | 93 | 0 | 20 | 9 | 113 | 102 | 91.18 | 82.30 | 86.51 | 25.66 |
| XTNT | 90 | 3 | 20 | 9 | 113 | 102 | 88.24 | 79.65 | 83.72 | 28.32 |
| CONT | 93 | 0 | 20 | 9 | 113 | 102 | 91.18 | 82.30 | 86.51 | 25.66 |
| XTNT+CONT+TYPE | 276 | 3 | 60 | 27 | 339 | 306 | 90.20 | 81.42 | 85.58 | 26.55 |
| location | | | | | | | | | | |
| TYPE | 531 | 24 | 176 | 25 | 731 | 580 | 91.55 | 72.64 | 81.01 | 30.78 |
| XTNT | 490 | 65 | 176 | 25 | 731 | 580 | 84.48 | 67.03 | 74.75 | 36.39 |
| CONT | 555 | 0 | 176 | 25 | 731 | 580 | 95.69 | 75.92 | 84.67 | 27.50 |
| XTNT+CONT+TYPE | 1576 | 89 | 528 | 75 | 2193 | 1740 | 90.57 | 71.87 | 80.14 | 31.55 |
| money | | | | | | | | | | |
| TYPE | 21 | 0 | 2 | 10 | 23 | 31 | 67.74 | 91.30 | 77.78 | 52.17 |
| XTNT | 21 | 0 | 2 | 10 | 23 | 31 | 67.74 | 91.30 | 77.78 | 52.17 |
| CONT | 21 | 0 | 2 | 10 | 23 | 31 | 67.74 | 91.30 | 77.78 | 52.17 |
| XTNT+CONT+TYPE | 63 | 0 | 6 | 30 | 69 | 93 | 67.74 | 91.30 | 77.78 | 52.17 |
| organization | | | | | | | | | | |
| TYPE | 261 | 23 | 139 | 48 | 423 | 332 | 78.61 | 61.70 | 69.14 | 49.65 |
| XTNT | 220 | 64 | 139 | 48 | 423 | 332 | 66.27 | 52.01 | 58.28 | 59.34 |
| CONT | 284 | 0 | 139 | 48 | 423 | 332 | 85.54 | 67.14 | 75.23 | 44.21 |
| XTNT+CONT+TYPE | 765 | 87 | 417 | 144 | 1269 | 996 | 76.81 | 60.28 | 67.55 | 51.06 |
| percent | | | | | | | | | | |
| TYPE | 45 | 0 | 5 | 13 | 50 | 58 | 77.59 | 90.00 | 83.33 | 36.00 |
| XTNT | 41 | 4 | 5 | 13 | 50 | 58 | 70.69 | 82.00 | 75.93 | 44.00 |
| CONT | 45 | 0 | 5 | 13 | 50 | 58 | 77.59 | 90.00 | 83.33 | 36.00 |
| XTNT+CONT+TYPE | 131 | 4 | 15 | 39 | 150 | 174 | 75.29 | 87.33 | 80.86 | 38.67 |
| person | | | | | | | | | | |
| TYPE | 455 | 5 | 80 | 15 | 540 | 475 | 95.79 | 84.26 | 89.66 | 18.52 |
| XTNT | 425 | 35 | 80 | 15 | 540 | 475 | 89.47 | 78.70 | 83.74 | 24.07 |
| CONT | 457 | 3 | 80 | 15 | 540 | 475 | 96.21 | 84.63 | 90.05 | 18.15 |
| XTNT+CONT+TYPE | 1337 | 43 | 240 | 45 | 1620 | 1425 | 93.82 | 82.53 | 87.82 | 20.25 |
| time | | | | | | | | | | |
| TYPE | 13 | 0 | 12 | 0 | 25 | 13 | 100.00 | 52.00 | 68.42 | 48.00 |
| XTNT | 12 | 1 | 12 | 0 | 25 | 13 | 92.31 | 48.00 | 63.16 | 52.00 |
| CONT | 13 | 0 | 12 | 0 | 25 | 13 | 100.00 | 52.00 | 68.42 | 48.00 |
| XTNT+CONT+TYPE | 38 | 1 | 36 | 0 | 75 | 39 | 97.44 | 50.67 | 66.67 | 49.33 |

Table 4.8: *The full scores from the baseline experiment using the full named entity extraction system trained on the training data alone.*

# Chapter 5

# Extension of the standard model

## 5.1 Preamble

In the previous chapter we introduced a standard statistical model that has proven effective for named entity extraction tasks. It has been successfully used both on written text and on transcribed speech (1-best recogniser output and manual transcriptions). In this chapter we introduce some improvements to the standard model and also some adaptations to make the model more portable to the general speech domain; in particular we use classification and regression trees (CART) rather than standard language model probabilities, and we add some simple rules to produce a basic hybrid system. We suggest an improvement to the traditional method of defining a joint probability - showing that the new derivation is not only better mathematically[1] but, as would be expected, it also leads to better named entity extraction. Having found this approach effective we continue to use it for the remainder of the thesis. Finally, we extend the model to work with word lattices rather than text (and from then on we always use the word lattices rather than speech transcript), and we identity a method to deal with out-of-vocabulary (OOV) words which occur as a result of using speech lattices.

[1]In that it requires fewer independence assumptions.

## 5.2 The CART before the horse

We ran one early experiment that used CART to predict $P(e_i)$ instead of using transition matrix $P(e_i|e_{i-1})$. The main reason for attempting to use CART was that it allowed more features to be used than solely the previous entity.

The full set of features that we used were:

- Whether the word had occurred before.
- The most common entity matched by the word.
- The second most common entity matched by the word.
- The third most common entity matched by the word.
- How many different entities the word corresponded to in the training data.
- What the previous word was classified as.
- What the second previous word was classified as.
- What the third previous word was classified as.
- Whether the word was a stop word.
- What the word was classified as the previous time.
- Whether the word occurred within the last 200 words.
- Whether the word occurred within the last 500 words.
- Whether the word has had multiple classifications since the start of the document.
- Whether the word was ever classified as Person in the training data.
- Whether the word was ever classified as Location in the training data.
- Whether the word was ever classified as Organization in the training data.
- Whether the word was ever classified as Time in the training data.
- Whether the word was ever classified as Date in the training data.

- Whether the word was ever classified as Percent in the training data.
- Whether the word was ever classified as Monetary expression in the training data.
- Whether the word was ever classified as not-an-entity in the training data.
- The frequency of the word in the training data.
- The entity that most frequently followed the word.
- The entity that was the second most frequent to follow the word.

Results from this experiment were disappointing. There was only a slight improvement in results over the previous model. There were a variety of reasons for the poor results which we discuss below.

### 5.2.1 Reasons for the failure

The first reason for the failure was a flaw in the mathematics. The mathematics involved the estimation of $P(e_i|E_1^{i-1})$. The CART used other information relating to the word itself; for example, e.g. its frequency. Thus the mathematics was unjustified.

The second reason for the failure was the use of the feature, "What the word was classified as the previous time". Whilst the feature is intuitively good, the problem lay with a misclassification, and more specifically the misclassification of the first occurence of the word. Intuitively, if the word under consideration is 'clinton', then the fact that the last time this word was classified it was classified as Person is a very good indication that this occurence also refers to a person. For this reason the CART placed high emphasis on the word's previous classification.

When the word was being classified for the first time, this feature was not available. The CART was therefore not well equipped to classify the word correctly - the word might have been incorrectly classified from the outset. When the word recurred, later classifications were based on its earlier classification and, if the earlier classification was wrong, the later classifications would likely be wrong also.

With poor results, we decided not to use CART, but instead to focus effort entirely on improving n-gram language models and the methods for using them.

## 5.3 A hybrid approach

In this section we describe a hybrid approach to detecting named entities from speech transcripts. The approach is based on a few basic rules and the HMM style statistical model described in chapter 4. The system consists of three component parts.

- Pre-Statistical Rules
- The Statistical Model
- Post-Statistical Rules

In order to obtain the final marked up data, the data is piped through all three as illustrated in figure 5.1.

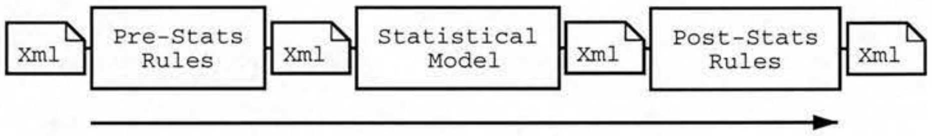


Figure 5.1: *Diagram of pipeline*

### 5.3.1 Pre-Statistical Rules

Each unit in the system takes XML as input and yields XML as output[2]. Thus the test data needed to be converted to XML prior to input to the Pre-Statistical Rules. The transformation was simple: a document type definition (DTD) describing the document was created and <DOC> markup and an attribute declaration to show the location of the XML DTD was added to the top of the text, and </DOC> markup was added to the bottom of the text.

The Pre-Statistical rules were written in XML, and the Text Tokenisation Toolkit (Grover et al. 2000) was used to parse them. There are two types of rules: those that mark up ENAMEX entities, and those that mark up NUMEX entities - the former being the simplest.

[2]Valid XML is required by the tools used for the rules.

The ENAMEX rules involve simple lookup in a lexicon. At this stage, '_u _s navy' would be marked up as Organisation, 'new york' would be marked up as Location and 'david' would be marked up as Person.

The NUMEX rules are also relatively simple. They involve the lookup of different currencies in a lexicon, and then a logical method of ensuring that all numbers, decimals, fractions, and even nonsense numbers such as 'million million' which are followed by a currency or 'percent' get correctly marked up as NUMEX.

Extra rules needed to be added to allow for the general case of 'three or four percent' and 'one point five to two million dollars' where the NUMEX entities do not contain key units. An example additional rule is given in figure 5.2.

```
<RULE name="money" match="SEQ" targ_sg="NUMEX[TYPE='MONEY']">
  <REL type="REF" match="number"></REL>
  <REL type="REF" match="to" m_mod="TEST"></REL>
  <REL match="NUMEX[TYPE='MONEY']" m_mod="TEST"></REL>
</RULE>
```

Figure 5.2: *An example additional rule for the Pre-Stats Rules*

### 5.3.2 Statistical Model

The standard statistical model described in chapter 4 is used here. Where the input text to the statistical model contained XML markup from the Pre-Statistical Rules, however, this information is used by the statistical model. The mark up is effectively used to adjust the probabilities on transitions in the statistical model. Thus the probabilities which correspond to a markup different to that already created by the Pre-Statistical rules are regarded as impossible and are multiplied by zero. For example, if 'three' has Percent markup when input to the statistical model, then the transition from state not-an-entity to state not-an-entity for the word 'three' will become zero; indeed all transitions to state not-an-entity from any entity for the word three will become zero. Similarly all transitions leading to any state other than the Percent state will also become zero.

Using this method, NUMEX entities which are classified by the Pre-Statistical rules are assumed to be completely correct. There is, however, a problem with using this method for ENAMEX. For example, "david" would be marked as Person, yet it could refer to a Location or Organization - as in the case where "david morgan" refers to the name of a department store, and should therefore be marked as an Organization. Similarly "new york" would be marked up as a Location, whereas "new york athletic commission" if it exists would actually be an Organization. For this reason a three-tier structure for ENAMEX is used in the Statistical Model.

- Organization is assumed to be Organization.
- Location is assumed to be Location or Organization.
- Person is assumed to be Person, Location or Organization.

Using this three-tier structure, an entity that has been marked as person, will have transitions entering the Monetary expression, Percentage, Time and Date states set to zero, while the probabilities of entering an ENAMEX state are allowed. Similarly if an entity has been marked as a Location, all entry to states other than Location and Organization are set to zero.

### 5.3.3 Post-Statistical Rules

Due to the simple nature of the statistical model there is no indication of length of individual entities within entity types. Thus the incorrect A in figure 5.3 would be obtained instead of the correct B. Although we address this issue properly in section 5.5, we need at this stage to set out the rules required to make the necessary adjustments

```
A <ENAMEX TYPE='LOCATION'>edinburgh great britain</ENAMEX>

B <ENAMEX TYPE='LOCATION'>edinburgh</ENAMEX>
  <ENAMEX TYPE='LOCATION'>great britain</ENAMEX>
```

Figure 5.3: *Example: actual (A) and correct (B) markup of locations by Statistical Model*

It is clear from the example that the solution is not as simple as splitting all the words with the named entity tags into separate named entities because this would cause multi-word entities to be incorrectly separated; for example, "great britian" would be split into two entities. It was therefore necessary to have lexical lookup of the locations within Location tags after the statistical model had completed. If any part of a tag matched a location in a lexicon, this part was separated from the remainder into a new tag. In this way Location named entities were corrected. This problem, as a general rule, only occurred for Locations and the method was therefore only used for Location tags.

### 5.3.4 Experiments

#### Experiment 1

The first experiment consisted of simply piping the test data through the three stages outlined in the description of the system. The F-scores of the results after each stage in the progression are outlined in table 5.1. As is clear from the results, each progressive step improves the overall result.

| Stage | F-score |
|---|---|
| Pre-Stats | 66.64 |
| Stats | 80.78 |
| Post-Stats | 83.17 |

Table 5.1: *Results at various stages in pipeline*

#### Experiment 2

The second experiment was designed to estimate the effectiveness of the initial markup stage. The method was to vary the certainty of the markup in the Pre-Statistical rules globally prior to the statistical model.

By regarding the certainty that the data was correct (i.e. the confidence $C$ in the rules) as a real number in the range [0,1], we calculated a factor $U$ equal to one minus the confidence.[3]

In the previous experiment, the Statistical Model acted definitively by multiplying the probabilities along all paths, other than those arriving at allowed states, by zero. In

[3] $U$ effectively is a measure of uncertainty about the rules.

this experiment, the paths were instead multiplied by $U$. Thus the Statistical Model was able to overrule the decision made by the Pre-Statistical rules if the probability of an alternative was high enough. The new markup was therefore accepted if it was chosen by the Statistical Model after the weighting was applied.

$U$ was kept constant throughout individual runs of the experiment, but was varied between zero and one in successive runs. It is important to note that during this stage the only varying factor between runs was $U$. The XML documents going into the Statistical Model were identical.

During this experiment, the markup of NUMEX entities was still assumed to be correct. The factor $U$ was therefore only used in the case of ENAMEX markup. Representative results are shown in table 5.2 below. These, together with some additional points, have been plotted on the graph in figure 5.4.

| $C$-level | $U$-level | F-measure |
|---|---|---|
| 0.0 | 1.0 | 80.07 |
| 0.9 | 0.1 | 82.35 |
| 0.99 | 0.01 | 83.28 |
| 0.999 | 0.001 | 83.46 |
| 0.9999 | 0.0001 | 83.37 |
| 0.99999 | 0.00001 | 83.23 |
| 1.0 | 0.0 | 83.17 |

Table 5.2: *Results at various confidence levels*

The graph shows the predictable steep growth as confidence is increased from zero ($U$ decreases), reflecting the clear advantage of having already marked up certain of the ENAMEX entities. As would be expected, as the confidence continues to grow the steepness of the increase decreases as we approach total confidence.

The graph then peaks (at $U$=0.001) and begins a steady but slow decline until plateauing out. The plateau has not been plotted on the graph, but the final point actually continues indefinitely. This decline indicates that the lexical markup was imperfect. This was not necessarily made clear by the low F-score of the Pre-Statistical

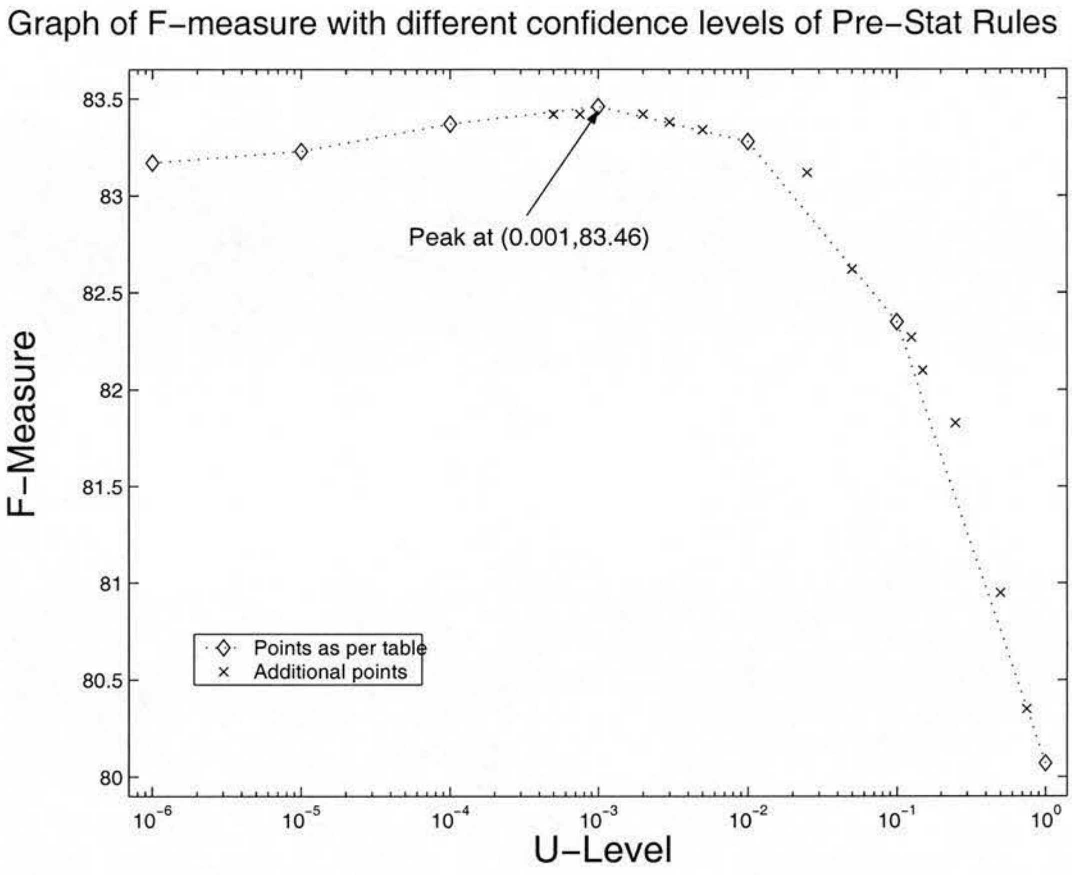


Figure 5.4: *Graph shows peak of 83.46*

rules, since the presence of spurious markup (especially in the case of Person entities) could be explained by single name markup (as illustrated in figure 5.5) and the fact that a Location which is marked up as Person could still be correct.

```
A <ENAMEX TYPE='PERSON'>james</ENAMEX>
  <ENAMEX TYPE='PERSON'>horlock</ENAMEX>

B <ENAMEX TYPE='PERSON'>james horlock</ENAMEX>
```

Figure 5.5: *Example: actual A and correct B markup of people by Pre-Stats Rules*

The results show that although using the output from the Pre-Stats Rules is beneficial, the inaccuracies mean that the results are not definitive. This suggests that higher results could be achieved if the individual markup contained probabilities that the Statistical Model could take into account.

### Experiment 3

The third experiment was designed to prove the benefit of associating probabilities with the markup of the first rules.

Probabilities for each phrase in the Organization lexicon were calculated by reading through the Organization lexicon phrase by phrase and counting the number of times that each phrase occurs within Organization markup of the training data (plus one - to avoid zero probabilities) and dividing by the total number of times the phrase occurs in the document (plus two - to avoid infinite probabilities). Thus a phrase that did not occur at all, but had been placed in the Organization lexicon, would have a probability of 0.5, one that occurred mostly outside of Organization markup would have a probability in the range (0,0.5); and one that occurred mainly inside markup would have a probability in the range (0.5,1). By storing these probabilities in the lexicon only one calculation was needed per lexical entry.

The new markup of Organization entities after the Pre-Statistical rules now contained probabilities as in figure 5.6.

```
<ENAMEX TYPE="ORGANIZATION">washington post</ENAMEX>
<ENAMEX PROB="0.9729" TYPE="ORGANIZATION">washington
                     post</ENAMEX>
```

Figure 5.6: *Pre-Stats Rules output in experiment 3 distinct from experiments 1&2*

The statistical system then used these probabilities, together with the general measure of certainty (necessary as not only the Organization entities needed to be treated as uncertain), to update the probabilities on the transitions between states in the model. The results of this experiment are shown in table 5.3 and have been plotted in figure 5.7 together with the results from experiment 2 to provide a baseline.

As can be seen from the graph, there is a slight improvement in the results because actual probability association is used instead of a single confidence measure.

In our view, the use of a more appropriate function of the probability would lead to further improvements in the critical places. We hypothesise that we would get even better results if the probabilities were associated with all ENAMEX rules rather than just organization rules.

| $C$-level | $U$-level | F-measure |
|---|---|---|
| 0.0 | 1.0 | 81.35 |
| 0.9 | 0.1 | 82.91 |
| 0.99 | 0.01 | 83.47 |
| 0.999 | 0.001 | 83.49 |
| 0.9999 | 0.0001 | 83.38 |
| 0.99999 | 0.00001 | 83.23 |
| 1.0 | 0.0 | 83.18 |

Table 5.3: *Results at various confidence levels*

We further conjecture that, as probabilities are now being associated individually with Organization entities, organisations which previously had been removed from the Organization lexicons (such as 'shell' and 'labor') because they were regarded as uncertain, could be returned. We predict that this addition would further improve results.

### 5.3.5 Conclusion

By combining several theories we have been able to present a hybrid system which shows reasonable performance on the test data. It is apparent that there are benefits from combining both statistical and rule-based approaches to the named entity task. The hybrid system is portable to other domains such as other languages because it does not rely on manually configured rules, but simply on lexical lookup.

At an early stage in the lexicon-making process, certain single word organizations were removed from the Organization lexicon. Words such as "labor" and "shell" were removed due to their ambiguity. It can be shown, however, that with the introduction of the probabilities associated with the markup at Pre-Statistical rule stage, by simply adding the Organization markup to each mention of "labor" with probability 0.34 (the calculated probability), the overall score is slightly increased.

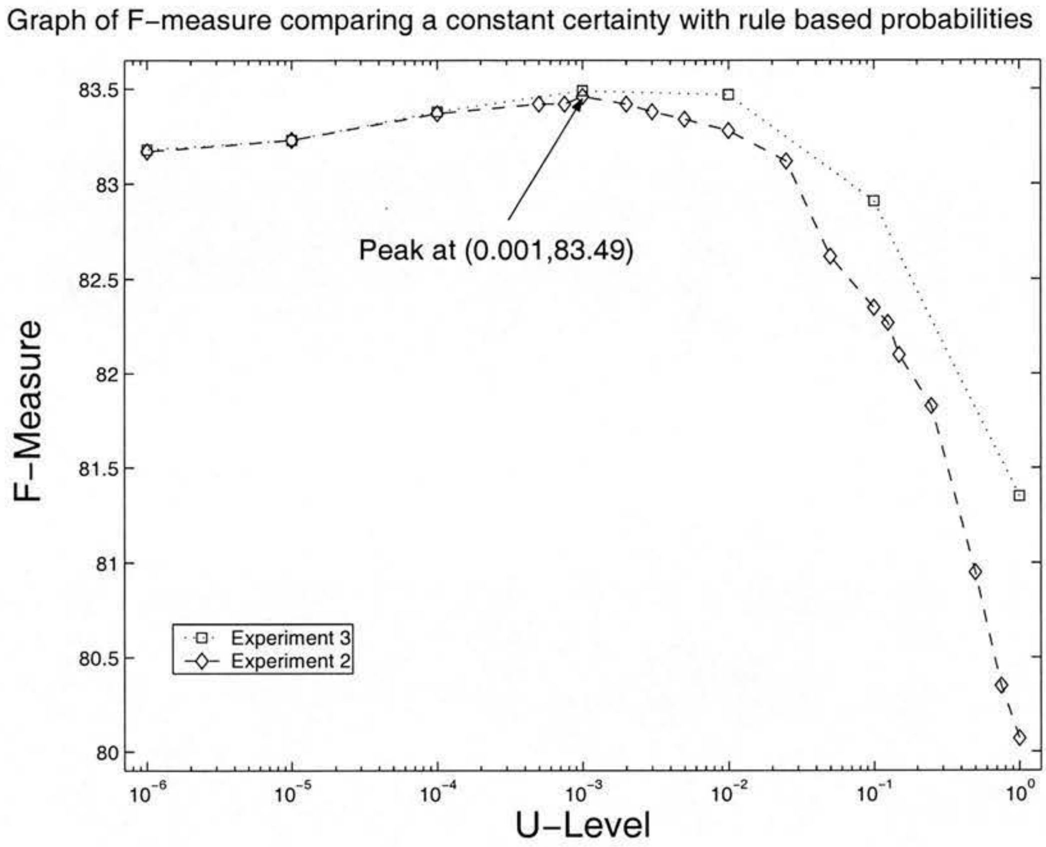


Figure 5.7: *Graph comparing results of experiment 2 with experiment 3*

## 5.4 What is the probability of A intersection B?

In chapter 4 we explored a generalisation of the most common statistical model used in the extraction of named entities. We noted that we were required to find the sequence which maximises $P(E, W)$; and subsequently that

$$P(E, W) = P(E).P(W|E) \tag{5.1}$$

or more explicitly

$$P(E_1^L, W_1^L) = P(E_1^L).P(W_1^L|E_1^L) \tag{5.2}$$

Thus

$$P(E_1^L, W_1^L) = \prod_{i=1}^{L} P(e_i|E_1^{i-1}). \prod_{i=1}^{L} P(w_i|W_1^{i-1}, E_1^L) \tag{5.3}$$

We were then able to approximate this probability to

$$P(E_1^L, W_1^L) \simeq \prod_{i=1}^{L} P(e_i|E_{i-2}^{i-1}). \prod_{i=1}^{L} P(w_i|W_{i-2}^{i-1}, E_{i-1}^{i}) \tag{5.4}$$

by making the following assumptions:

- $P(e_i|E_1^{i-1}) = P(e_i|E_{i-2}^{i-1})$; i.e. $e_i$ is independent of $E_1^{i-3}$.
- $P(w_i|W_1^{i-1}, E_1^L) = P(w_i|W_{i-2}^{i-1}, E_{i-1}^{i})$; i.e. $w_i$ is independent of $W_1^{i-3}, E_1^{i-2}, E_{i+1}^{L}$.

### 5.4.1 An alternative definition

There is an alternative way of viewing $P(E, W)$ which is equally valid, but does not require such strong independence assumptions for n-gram estimation.

$$\begin{aligned} P(E_1^L, W_1^L) &= P(e_1).P(E_2^L, W_1^L|e_1) \\ &= P(e_1).P(w_1|e_1).P(E_2^L, W_2^L|e_1, w_1) \\ &= P(e_1).P(w_1|e_1).P(e_2|e_1, w_1).P(E_3^L, W_2^L|e_1, w_1, e_2) \\ &\dots \\ &= \prod_{i=1}^{L} P(e_i|E_1^{i-1}, W_1^{i-1}).P(w_i|E_1^{i}, W_1^{i-1}) \end{aligned} \tag{5.5}$$

Thus

$$P(E_1^L, W_1^L) = \prod_{i=1}^{L} P(e_i|E_1^{i-1}, W_1^{i-1}). \prod_{i=1}^{L} P(w_i|E_1^{i}, W_1^{i-1}) \tag{5.6}$$

We are then able to approximate this probability to

$$P(E_1^L, W_1^L) \simeq \prod_{i=1}^{L} P(e_1|E_{i-2}^{i-1}, W_{i-2}^{i-1}). \prod_{i=1}^{L} P(w_i|E_{i-2}^{i}, W_{i-2}^{i-1}) \tag{5.7}$$

### 5.4.2 The differing n-gram assumptions

The assumptions required to use an n-gram language model to estimate the probabilities in equation 5.3 are

- $P(e_i|E_1^{i-1}) = P(e_i|E_{1+i-n}^{i-1})$
- $P(w_i|W_1^{i-1}, E_1^L) = P(w_i|W_{1+i-n}^{i-1}, E_{2+i-n}^{i})$

The assumptions required to use an n-gram language model to estimate the probabilities in equation 5.6 are only

- $P(e_i|E_1^{i-1}, W_1^{i-1}) = P(e_i|E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1})$
- $P(w_i|E_1^{i}, W_1^{i-1}) = P(w_i|E_{2+i-n}^{i}, W_{1+i-n}^{i-1})$

Both methods assume independence over $W_1^{i-n}$ and $E_1^{i-n}$. The first method, however, also assumes independence over $E_{i+1}^L$. It would therefore appear that the second method assumes less and is, therefore, a more accurate method of estimating the probabilities.

### 5.4.3 A backoff strategy

A suitable backoff strategy for the estimation of $P(w_i|E_{2+i-n}^{i}, W_{1+i-n}^{1+i-1})$ has been discussed at the end of section 4.6 in chapter 4. We now need a suitable strategy for the estimation of $P(e_i|E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1})$.

A bottom-up strategy was used to solve this problem by examining the language models that had already been constructed. It became apparent that these probabilities were already known without any need for development of additional language models. In the previous method a transition matrix (a simplified language model) was required. As is shown below, this is not necessary for the new method; the original language models which after being trained as per section 4.6 already contain trigrams containing the information required. For example, the not-an-entity language model already contains $P(< org\backslash > |you're\ watching)$ - which is clearly a better predictor of the probability of transitioning into the organization state (given the context "you're watching") than the probability stored in the transition matrix.

#### The "Simi Valley California" problem revisited

In general, there are two possible ways for each $e_i$ to be an assigned entity e. It can either be the same e as $e_{i-1}$, or it can be a different e than $e_{i-1}$. To illustrate, consider the following two examples.

1. $< PLACE >$ *Edinburgh* $< /PLACE >< PLACE >$ *Scotland* $< /PLACE >$

2. $< PERSON >$ *James Horlock* $< /PERSON >$

Using our notation these would be described:

1. $\begin{array}{ll} w_1 = \text{Edinburgh} & e_1 = \text{Location} \\ w_2 = \text{Scotland} & e_2 = \text{Location} \end{array}$

2. $\begin{array}{ll} w_1 = \text{James} & e_1 = \text{Person} \\ w_2 = \text{Horlock} & e_2 = \text{Person} \end{array}$

In both cases it is clear that the second entity $e_2$ is of the same type as the first entity $e_1$. However, only in the second case are the first and second entities contained within the same named entity tag. We can therefore write

$$P(e_i|E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1}) = P(e_i[same]|E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1}) + P(e_i[different]|E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1}) \tag{5.8}$$

Equation 5.8 does not apply when $e_i \neq e_{i-1}$, or in the special case when $e_i$ is not-an-entity. Alternatively, the formula may be considered to apply in these cases if it is assumed that in these cases $P(e_i[same]|E_{i-n}^{i-1}, W_{i-n}^{i-1}) = 0$.

The language models that we have built already contain enough information to calculate all these probabilities, since the probability of the entity being the same as the previous entity is simply one minus the probability of it not being the same, and the relevant language model contains the probability of the entity not being the same. The probability of it not being the same is simply $P(< /s > |W_1 + i - n)$. The language model can therefore predict $P(e_i[same]|E_{i-n}^{i-1}, W_{i-n}^{i-1})$ as per equation 5.9.

$$\begin{aligned} P(e_i[same]|E_{i-n}^{i-1}, W_{i-n}^{i-1}) &= 0 \text{ in the special cases} \\ &= 1 - P(< /s > |E_{i-n}^{i-1}, W_{i-n}^{i-1}) \text{ otherwise} \end{aligned} \tag{5.9}$$

Similarly, the probability of the entity being of the same type, but being a different entity, is simply the probability of not being the same entity ($P(< /s > |W_1 + i - n)$ - calculated as above) multiplied by the probability of entering in the particular entity state ($P(< e_i > |E_{i-n}^{i-1}, W_{i-n}^{i-1})$ - calculated the same way that any new entity is calculated, from the not-an-entity language model).

Where $P(< e_i > |E_{i-n}^{i-1}, W_{i-n}^{i-1}) = 1 - \sum_e P(< e > |E_{i-n}^{i-1}, W_{i-n}^{i-1})$ if $e_i$ is not-an-entity, and similarly $P(< /s > |E_{i-n}^{i-1}, W_{i-n}^{i-1}) = 1$ if $e_i$ is not-an-entity, the language models can therefore predict $P(e_i[different]|E_{i-n}^{i-1}, W_{i-n}^{i-1})$ as per equation 5.10.

$$P(e_i[different]|E_{i-n}^{i-1}, W_{i-n}^{i-1}) = P(< /s > |E_{i-n}^{i-1}, W_{i-n}^{i-1}).P(< e_i > |E_{i-n}^{i-1}, W_{i-n}^{i-1}) \quad (5.10)$$

There are only two probabilities that we need to obtain from our language models to solve equation 5.10; namely probabilities 5.11 and 5.12.

$$P(< /s > |E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1}) \quad (5.11)$$

$$P(< e_i > |E_{1+i-n}^{i-1}, W_{1+i-n}^{i-1}) \quad (5.12)$$

Probability 5.11 is estimable when $e_{1+i-n} = .. = e_{i-1}$ by finding the (possibly backed-off) $P(< /s > |W_{i-n}^{i-1})$ in the $e_{i-1}$ language model. When this is not the case, we back off equation 5.11 to $P(< /s > | < s >, W_{i-j}^{i-1})$ for maximal j such that $e_{i-j} = .. = e_{i-1}$.

Probability 5.12 is estimable when $e_{1+i-n} = .. = e_{i-1}$ is not-an-entity by finding the (possibly backed-off) $P(< e_i > |W_{i-n}^{i-1})$ in the not-an-entity language model. In the other cases 5.12 is estimated by $P(< e_i > |[E \vee W]_{i-n}^{i-1})$ in the not-an-entity language model; where $e_i \vee w_i = w_i$ if $e_i$ is not-an-entity and $= e_i$ otherwise, and $[E \vee W]_j^k$ is the sequence $e_j \vee w_j$ .. $e_k \vee w_k$ following the usual notation.

## 5.4.4 Graphical interpretation

We now present the graphical interpretation of the mathematics described above. The improved model topology is shown in figure 5.8. Although the new topology bears some resemblance to the old topology, there are some important distinctions. The first distinction is that, where originally there were simple states for each of the entities, these have been replaced by more complex objects comprising two separate states (nodes). In order to continue the terminology of the previous chapters we refer to these two states as "nodes", and to the objects containing the two nodes as "states". According to our terminology, each entity has a single "state", but that "state" is comprised of two "nodes".

The second distinction is that, where previously all states were inter-connected (apart from start and end states), this is no longer the case.

In this new topology (figure 5.8), the probabilities of words are still generated within states. However, they are not generated within nodes.

In the new topology there is no single transition between two different entity states. In order to transit from one entity state to another entity state it is necessary to pass through the not-an-entity state. Consequently the new model topology contains far fewer transitions than the old model topology did. The old model topology contained 8 states each with 9 transitions entering them and 1 end state with 8 transitions entering it; i.e. $(8 \times 9) + 8 = 80$ transitions in total. The new model contains 7 states with 1 transition entering them, 1 state with 8 transitions entering it, and 1 end state with 1 transition entering it. Eight of the states in the new model, however, have a more complex structure than the old model, requiring two additional transitions internal to each entity state; i.e. $(7 \times 1) + (1 \times 8) + 1 + (8 \times 2) = 32$ transitions in total.

Figure 5.8 shows a variety of arrows. We deal with these arrows in three stages: firstly, the probabilities associated with each type of arrow, secondly, what the arrows require as input, and thirdly, the output associated with each type of arrow. There are three types of arrows shown on the diagram: vertical arrows on curved lines (hereafter VC), horizontal arrows on curved transitions (hereafter HC), and horizontal arrows on horizontal transitions (hereafter HH).

Firstly, the probabilities associated with each type of arrow. VC are the probabilities associated with entering and leaving entities; that is, they are either $P(e_i|E_{1+i-n}^{i-1})$ or $P(< /s > |E_{1+i-n}^{i-1})$. HC are the probabilities within states; that is, those generated by the state specific language models $P(w_i|E_{1+i-n}^{i-1})$. HH are generally the probabilities

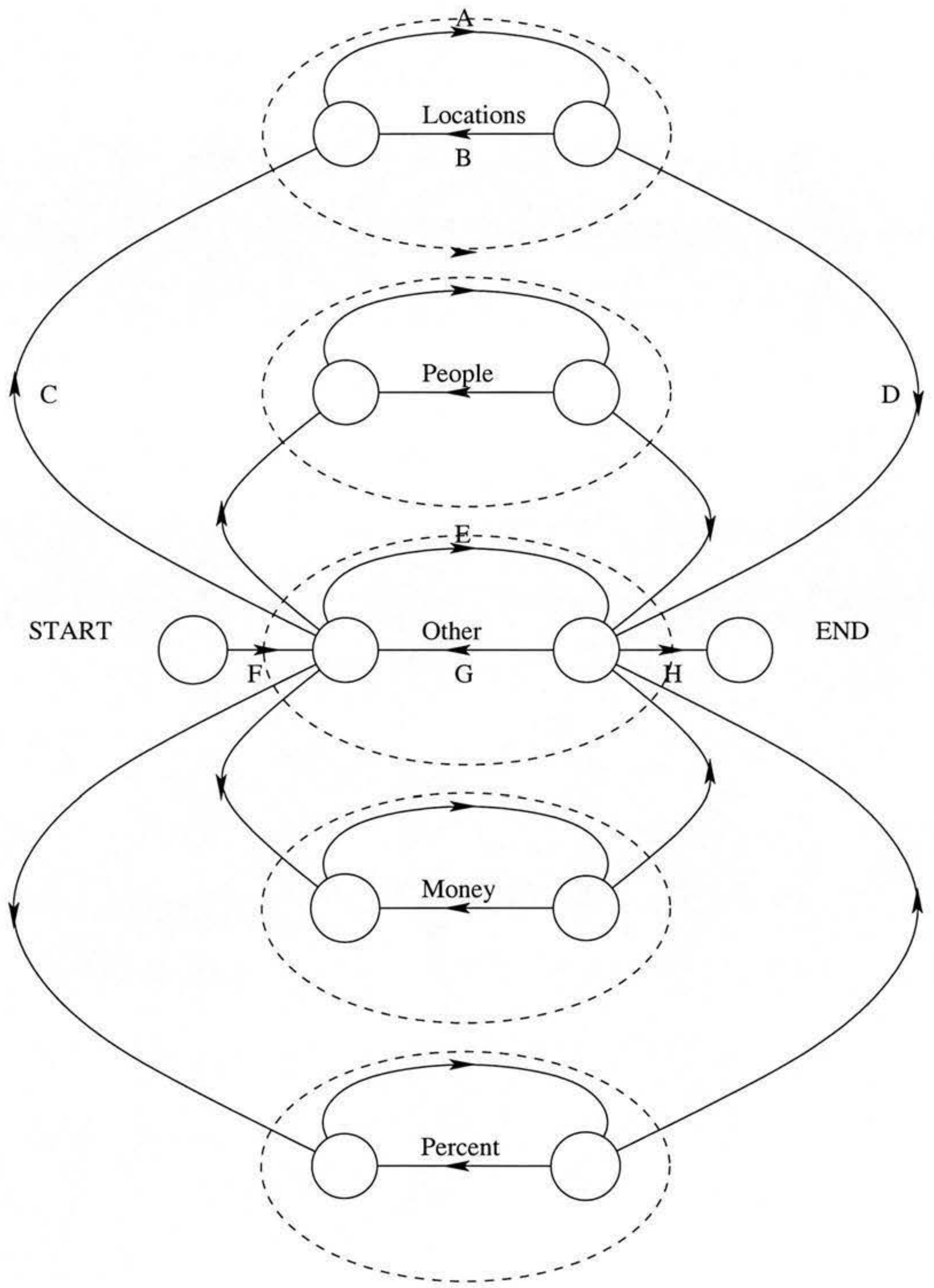


Figure 5.8: *Topology of the model: vertical arrows generate mark up; horizontal arrows on curved transitions generate words from respective language models; horizontal arrows on horizontal transitions generate nothing, but are possible transitions.*

associated with remaining in the current entity. In the entity state, the HH probabilities are simply found by subtraction. In the not-an-entity state, the probability is binary because there are only two possible transitions and one can only be taken when the end of the sentence has been reached[4] - and so generally the probability of the horizontal arrow in the not-an-entity state is 1. The probability of HH between the start state and the not-an-entity state is always 1, and between the end state and the not-an-entity state is binary.

Secondly, what the arrows require as input. VC require no input; that is, these transitions occur between words. HC require words as input. HH require no input. It is therefore possible for multiple transitions of type VC and HH to occur without requiring any words. By this means it is possible to transit from one entity state to another entity state between words - as is often required. It is also worth noting that HH are required between every pair of HC.

Thirdly, the output generated by each type of arrow. VC generate markup; that is, each VC will generate some markup: VC transitioning from the not-an-entity state will generate opening markup (eg <TIMEX TYPE="DATE">) and VC transitioning to the non-an-entity state will generate closing markup (eg </NUMEX>). HC generate words. HH generate nothing.

This can be illustrated if we consider the sentence

`the man from del monte he say no`

which when correctly marked up looks like

`the man from <ENAMEX TYPE=''LOCATION''>del monte</ENAMEX> he say no`

We are able to determine the types of arrows that need to be followed from comparison between the input and the output. These are HC, HC, HC, VC, HC, HC, VC, HC, HC, HC. As noted above, and made clear by figure 5.8, this sequence is impossible without having a single HH between each HC.

The correct sequence of arrows which should be followed to produce this output, using the labels from figure 5.8, is FEGEGEGCABADGEGEGEH . It is clear that this sequence is HH, HC, HH, HC, HH, HC, HH, VC, HC, HH, HC, VC, HH, HC, HH, HC,

[4]More specifically, if the !SENT_END word has been reached, as we are now working with lattices, but the model holds for text.

HH, HC, HH as expected.

Finally in this example, from this sequence we can see the probabilities that were used to generate the output. They are shown in table 5.4.

### 5.4.5 Examples

Having presented the mathematics together with the graphical interpretation, we now present some examples to complete our description of the model. We consider the following part of a named entity marked-up sentence:

.. the stock market rose by <NUMEX TYPE="PERCENT">fifteen percent</NUMEX> the <ENAMEX TYPE="ORGANIZATION">dow</ENAMEX> ..

We will consider both the actual markup and some possible alternatives to illustrate the probabilities used. We start by considering the probabilities associated with the word "rose" and the not-an-entity entity associated with the word "rose".

In the cases where a word does not occur in the vocabulary, the unknown word <UNK> is used to replace the word, and, if the probability needs to be backed-off, it is backed-off in the usual manner.

The correct trigram probability for the entity of "rose" (namely, not-an-entity) is $P(\textit{not-an-entity}|\textit{not-an-entity}, stock, \textit{not-an-entity}, market)$. There are seven alternatives to this probability as shown in equation 5.13, all of which are considered as the token transits from the not-an-entity state to the 7 respective entity states.

$$\begin{aligned}
P(<PEOPLE> |not-an-entity, stock, not-an-entity, market)\\
P(<PLACE> |not-an-entity, stock, not-an-entity, market)\\
P(<ORGANIZATION> |not-an-entity, stock, not-an-entity, market)\\
P(<TIME> |not-an-entity, stock, not-an-entity, market)\\
P(<DATE> |not-an-entity, stock, not-an-entity, market)\\
P(<MONEY> |not-an-entity, stock, not-an-entity, market)
\end{aligned}$$

| Arrow | Type | Probability |
|---|---|---|
| F | HH | 1 |
| E | HC | $P(the \mid not-an-entity, !SENT\_START)$ |
| G | HH | 1 |
| E | HC | $P(man \mid not-an-entity, !SENT\_START, the)$ |
| G | HH | 1 |
| E | HC | $P(from \mid not-an-entity, the, man)$ |
| G | HH | 1 |
| C | VC | $P(< PLA/ > \mid not-an-entity, man, from)$ |
| A | HC | $P(del \mid location, < s >)P(t_5 \mid location, < s >)$ |
| B | HH | $(1 - P(< /s > \mid location, < s >, del)$ |
| A | HC | $P(monte \mid location, < s >, del)$ |
| D | VC | $P(< /s > \mid location, del, monte)$ |
| G | HH | 1 |
| E | HC | $P(he \mid location, not-an-entity, from, < PLA/ >)$ |
| G | HH | 1 |
| E | HC | $P(say \mid not-an-entity, < PLA/ >, he)$ |
| G | HH | 1 |
| E | HC | $P(no \mid not-an-entity, he, say)$ |
| H | HH | $P(!SENT\_END \mid not-an-entity, say, no)$ |

Table 5.4: *The probabilities associated with the arrows in figure 5.8.*

$$P(< PERCENT > |not - an - entity, stock, not - an - entity, market) \tag{5.13}$$

Each of the probabilities in 5.13 is estimated in the not-an-entity language model. Since the not-an-entity language model contains $P(< tim/ > |stock, market)$ for example, we know the $P(< TIME > |not\text{-}an\text{-}entity, stock, not\text{-}an\text{-}entity, market)$, because the nature of the not-an-entity language requires that the words "stock" and "market" are of the not-an-entity 'entity'.

It is therefore a very simple process to deduce the $P(not\text{-}an\text{-}entity|not\text{-}an\text{-}entity, stock, not\text{-}an\text{-}entity, market)$ since the probabilities must sum to unity.

$$\begin{aligned} &P(not - an - entity|not - an - entity, stock, not - an - entity, market) \\ &\quad = 1 - \sum_e P(e|not - an - entity, stock, not - an - entity, market) \end{aligned} \tag{5.14}$$

The next stage is to approximate $P(rose|not\text{-}an\text{-}entity, stock, not\text{-}an\text{-}entity, market)$. When dealing with a best possible path there is only one possibility for what the word may be. However, in the lattice cases, there may well be other possibilities. For example, the lattice may offer an alternative of "rows" to the word "rose", in which case the $P(rows|not\text{-}an\text{-}entity, stock, not\text{-}an\text{-}entity, market, not\text{-}an\text{-}entity)$ also needs to be found.

Evaluating these probabilities is even simpler than that of evaluating the corresponding entity because, within the not-an-entity language model, $P(rose|stock, market)$ is equal to $P(rose|not\text{-}an\text{-}entity, stock, not\text{-}an\text{-}entity, market)$, and $P(rows|stock, market)$ is equal to $P(rows|not\text{-}an\text{-}entity, stock, not\text{-}an\text{-}entity, market)$.

We have therefore described the simplest set of probabilities - the probabilities of not-an-entity words given not-an-entity context. These are the simplest because they require no back-off strategy.

Finding the probability of the word and the entity for 'by' in the phrase we are examining is an identical process. That is, we simply evaluate $P(by|market, rose)$ directly from the not-an-entity language model and $P(not\text{-}an\text{-}entity|market, rose)$ by $1 - \sum_e P(e|market, rose)$ respectively.

The probability of the entity for the word 'fifteen' is straightforward as it simply requires the $P(< per/ > |rose, by)$ from the not-an-entity language model.

However, $P(fifteen|not\text{-}an\text{-}entity, rose, not\text{-}an\text{-}entity, by, < PERCENT >)$, requires back-off. In this instance, we would back-off to $P(fifteen| < s >)$ within the Percent language model, as the Percent language model has been trained on examples of percentages in this format.

The probability of the entity for the word 'percent' is reasonably straightforward again in the instance shown. We should however notice first that there are eight alternatives for this markup, as shown in 5.15.

$$\begin{aligned}
P(not - an - entity|not - an - entity, by, < PERCENT >, fifteen) \\
P(< PEOPLE > |not - an - entity, by, < PERCENT >, fifteen) \\
P(< PLACE > |not - an - entity, by, < PERCENT >, fifteen) \\
P(< ORGANIZATION > |not - an - entity, by, < PERCENT >, fifteen) \\
P(< TIME > |not - an - entity, by, < PERCENT >, fifteen) \\
P(< DATE > |not - an - entity, by, < PERCENT >, fifteen) \\
P(< MONEY > |not - an - entity, by, < PERCENT >, fifteen) \\
P(< PERCENT > |not - an - entity, by, < PERCENT >, fifteen)
\end{aligned} \tag{5.15}$$

The probabilities in 5.15 break down into three categories: $P(not - an - entity)$, $P(< PERCENT >)$ and $P('the\ rest')$.

We start with $P(< PEOPLE > |not\text{-}an\text{-}entity, by, < PERCENT >, fifteen)$. This probability is in effect a combination of two probabilities, namely the probability that the token is no longer in the Percent state, and that the token is now in the People state. The probability is given in equation 5.16 and calculated by $P(< /s > | < s >, fifteen)$ (from the Percent language model) $\times$ $P(< peo/ > |by, < per/ >)$ (from the not-an-entity language model).

$$P(< PEOPLE > |not - an - entity, by, < PERCENT >, fifteen) =$$

$$P(< /s > |not - an - entity, by, < PERCENT >, fifteen) \times$$
$$P(< PEOPLE > |not - an - entity, by, < PERCENT >, fifteen) \quad (5.16)$$

The second possibility we consider is the $P(< PERCENT >)$. There are two possible ways that the word could be marked up as Percent. First, it could be in the same entity as the previous entity (which in this example is correct) and the probability of this happening is simply $1 - P(< /s > \mid < s >, fifteen)$ from within the Percent language model. Second, it could be a new entity, in which case the probability of Percent is calculated in the same way as the $P(< PEOPLE >)$ (i.e. $P(< /s > \mid < s >, fifteen)$ (from the Percent language model) times $P(< per/ > |by, < per/ >)$ (from the not-an-entity language model).

The third and final possibility which we consider is the $P(not\text{-}an\text{-}entity)$. There is only one way in which this can happen and that is if it stops being a Percent, and doesn't start being anything else. The $P(not\text{-}an\text{-}entity)$ is therefore calculated by $1 - \sum_e P(e|by, < PERCENT >)$ (each probability calculated from the not-an-entity language model) times $P(< /s > \mid < s >, fifteen)$ (from the Percent language model).

These are examples of every probability that will be calculated by the language models (excluding the back-off done within the language models, which was dealt with in chapter 4).

### 5.4.6 Experiment

Having devised a way to find the named entity sequence by making less independence assumptions than are usually made, the next stage was to implement the changes. As we have already shown, the probabilities necessary to make the changes were already available from the language models, and so the code only required minor changes to remove the use of probabilities from the transition matrix and to add the use of additional probabilities from the language models.

Prior to the changes, the results on the test data were those of our baseline in section 4.7 with an F-scores of 79.82. Having implemented the changes, the new results - based on identical language models, identical weightings of probabilities and the same word lattices - produced an F-score of 83.21.

We conclude from this investigation that this alternative method of estimating the probability of $P(E_1^n, W_1^n)$ is both theoretically and experimentally an improvement on the traditional method of estimating probabilities.

Experimentally we have shown that a gain of over 3% absolute in F-score is obtained by this alternative method. Consequently we use this method of estimation in all future experiments detailed in this thesis.

## 5.5 Using lattices not text

The idea of using word lattices rather than text was inspired primarily by the statements of (Kubala et al. 1998) and (Kim 2001) that Slot Error Rate (SER) and (100 - F-score) were directly proportional to Word Error Rate (WER).

We considered the above statements to be true in a general sense, but also believed that there were other factors which needed to be considered before concluding that the only way to improve F-measure and SER was to improve WER. Indeed, we wanted to test the hypothesis that it was possible to obtain a trade-off between WER and SER/F-measure.

Word lattices are used in speech recognition prior to obtaining the final 1-best transcription of the speech. A small fraction of a speech lattice from the test set is shown in table 5.5. The lattice contains possible word matches to the speech signal for various times in chronological order, keeping a record both of language model probabilities and acoustic probabilities of the given word. In general the use of "word" here can refer to any unit: a phone, syllable, English word etc. In our case, the lattices contained English language words from an unknown fixed vocabulary. The vocabularly was clearly finite and was known by Cambridge University when the lattices were compiled; the information was not, however, conveyed with the lattices. Fortunately, as we explain in section 5.6, it was not required.

Obtaining the 1-best path from a speech lattice is a fairly simple process - it is simply a matter of generating all paths, together with their respective probabilities, and selecting the most probable one, as described in section 3.2.2. The Viterbi algorithm (Viterbi 1967) exploits recursion to reduce computational load. We were able to use

| | | | | | |
|---|---|---|---|---|---|
| J=15337 | S=6322 | E=6324 | W=JONES | a=-2133.28 | l=-1.115 |
| J=15342 | S=6323 | E=6326 | W=JONES | a=-2133.28 | l=-2.495 |
| J=15346 | S=6324 | E=6330 | W=INDUSTRIAL | a=-3698.22 | l=-0.585 |
| J=15347 | S=6325 | E=6330 | W=INDUSTRIAL | a=-3698.22 | l=-1.492 |
| J=15348 | S=6326 | E=6330 | W=INDUSTRIAL | a=-3698.22 | l=-0.779 |
| J=15349 | S=6327 | E=6330 | W=INDUSTRIAL | a=-3698.22 | l=-1.298 |
| J=15350 | S=6328 | E=6330 | W=INDUSTRIAL | a=-3698.22 | l=-2.072 |
| J=15351 | S=6329 | E=6330 | W=INDUSTRIAL | a=-3698.22 | l=-1.202 |
| J=15352 | S=6330 | E=6331 | W=AVERAGE | a=-2521.04 | l=-0.032 |
| J=15353 | S=6330 | E=6332 | W=AVERAGE | a=-2579.44 | l=-0.032 |
| J=15354 | S=6330 | E=6333 | W=AVERAGE | a=-2603.15 | l=-0.032 |
| J=15355 | S=6330 | E=6334 | W=AVERAGE | a=-2579.44 | l=-0.032 |
| J=15356 | S=6330 | E=6335 | W=AVERAGE | a=-2697.81 | l=-0.032 |
| J=15357 | S=6330 | E=6336 | W=AVERAGE | a=-2651.82 | l=-0.032 |
| J=15358 | S=6330 | E=6337 | W=AVERAGE | a=-2734.74 | l=-0.032 |
| J=15359 | S=6330 | E=6338 | W=AVERAGE | a=-2861.92 | l=-0.032 |
| J=15360 | S=6330 | E=6339 | W=AVERAGE | a=-2822.07 | l=-0.032 |
| J=15361 | S=6330 | E=6340 | W=AVERAGE | a=-2911.29 | l=-0.032 |
| J=15362 | S=6330 | E=6341 | W=AVERAGE | a=-2911.29 | l=-0.032 |
| J=15363 | S=6330 | E=6342 | W=AVERAGE | a=-2911.29 | l=-0.032 |
| J=15364 | S=6330 | E=6343 | W=AVERAGE | a=-2911.29 | l=-0.032 |
| J=15365 | S=6331 | E=6344 | W=A | a=-445.94 | l=-7.589 |
| J=15429 | S=6332 | E=6391 | W=CASH | a=-1552.63 | l=-10.460 |
| J=15442 | S=6332 | E=6400 | W=CAP | a=-1530.17 | l=-13.817 |
| J=15443 | S=6332 | E=6401 | W=CAP | a=-1562.18 | l=-13.817 |
| J=15418 | S=6333 | E=6381 | W=TEN | a=-1429.67 | l=-8.263 |
| J=15419 | S=6334 | E=6382 | W=PATH | a=-1392.95 | l=-14.275 |

Table 5.5: *Subsection of the first lattice corresponding to the first paragraph of the test data.*

Viterbi search techniques to find the best path efficiently, without the need to evaluate all paths, since the language model probabilities within the word lattices have been generated by n-gram models. Once two paths have overlapped for n words and are in the same state (i.e. have the same immediate history) the one with the lower probability can be discarded because it can never become more probable than a more probable path following the same route to the end of the lattice.

We also implemented some pruning because we would not expect that paths which start with very low probability to end more probable than paths which start with high probability. Although using no pruning gave best results, the system became too inefficient. For example, we investigated paths corresponding to phrases containing only the word "a". We found that there are 512 permutations of the 3 word sequence "a a a"[5]; and consequently 512 possibilities were being calculated for every trigram within the lattice. By fixing a beam (a value within which the probability of the word sequence must fall, compared to the current best for that state), efficiency was increased considerably. We experimented with various beam sizes and found that a beam width of 60 consistently showed identical results to results with an infinite beam - and a beam width of 20 offered comparable results. We therefore adopted 60 as a fixed beam for final results and 20 for comparisons requiring efficiency.

The task of named entity recognition of a single best path is easily scaled up to that for a speech lattice. In the previous chapter we showed that the named entity search was finite state and therefore we can use the Viterbi algorithm. The search for the best path through the lattice is also finite state thereby enabling the use of Viterbi. It is straightforward to combine these two searches. The task is simply to select the most probable path which corresponds to a single pass through both processes simultaneously.

As our method for the single best path for named entity tagging uses the token passing algorithm (Young et al. 1989), the increased complexity simply resulted in more tokens to pass around the FSM.

If we view the task as extending the model of the previous chapter, all we do is

[5]Any single word has 8 permutations (people, places, organizations, times, dates, percentages, money, or not-an-entity). Any pair of words therefore has $8^2$, and any word triplet has $8^3 = 512$ permutations. Fortunately, the Markov assumption renders an $8^n$ limit for n-grams.

propogate each node in the lattice through the statistical model in the same way as previously we had propogated each word instance in a text through the statistical model. That is, previously we treated words as tokens and propogated them through the FSM using the token passing algorithm; now we treat lattice nodes (which are instances of words) as tokens and propogate these through the FSM using the token passing algorithm.

Since the lattice is ordered, provided the nodes are propogated sequentially, tokens are guaranteed to store the best path[6] to that token. Since each state contains the optimum token we can safely propogate as in the case of text. Essentially the only difference between the text FSM and the lattice FSM is that all tokens arriving at any one state had the same word history in their path in the text FSM, whereas tokens arrive at a state from multiple lattice nodes, and thus have different nodes (and correspondingly different words) in their history. This doesn't affect what happens at the state, where essentially the probability of new tokens that arrive are compared with the probability of the current token in the state (if one exists). The new token replaces the current one only if the probability is greater than that of the current token; otherwise the token is deleted.

If we want to estimate the added complexity of using lattices rather than text, we simply need to compare the number of tokens for each. For example, if we consider a 30 second utterance of 75 words, it is not unreasonable for this utterance to produce a word lattice of 270,000 nodes.[7] The FSM therefore takes the same amount of time to process this 30 second utterance in lattice mode as to process a 1-best utterance of 270,000 words (equivalent in time to approximately 30 hours of spoken speech). It therefore takes the equivalent of one hour of text processing time to process 1 second of lattice based speech.

[6]In practice, as already explained, tokens do not store paths, they simply store pointers back to where they came from. This is, however, equivalent to storing the path.

[7]Such as the 74th utterance in the test data.

### 5.5.1 The Mathematics

It is evident that the new task of named entity extraction has changed somewhat from the old task. Previously the named entity extraction task was to find a suitable entity sequence which mapped to a known word sequence. In the new task there is no known word sequence and the task is therefore to find a sequence of words and the corresponding sequence of entities. Mathematically speaking it is no longer necessary to find the $\arg\max_{E_1^L} P(E_1^L|W_1^L)$; instead we require the $\arg\max_{E_1^L,W_1^L} P(E_1^L, W_1^L)$, or, more explicitly, the $\arg\max_{E_1^L,W_1^L} P(E_1^L, W_1^L|lattice)$.

When using a speech lattice instead of a single transcript, it is no longer possible to treat the probability of a word sequence, $P(W_1^L)$ as a constant, as is assumed between equations 4.1 and 4.3 in the previous chapter. We are not, however, required to make this assumption because in the case of word lattices we are not trying to maximise $P(E_1^L|W_1^L)$; we are rather trying to maximise $P(E_1^L, W_1^L)$. That is, we are seeking to maximise the joint probability of words and entities.

We are therefore required to find the word sequence and corresponding entity sequence which maximises $P(E_1^L, W_1^L)$ and can therefore follow the mathematics of the previous section (equations 5.5, 5.6, and 5.7) to arrive at equation 5.17. A more precise description of the calculation that we are performing is given in equation 5.18. We will use the shorter form (equation 5.17) for the remainder of this thesis since the lattice is always assumed. Equation 5.18 is introduced to demonstrate that the lattice is not ignored - in particular the lattice contains acoustic probabilities which need to be taken into account when making calculations. These lattice probabilities are taken into account in the ratio, stated within the lattices; language model log likelihoods are to be weighted 14 times as highly as acoustic likelihoods. These likelihoods are simply incorporated into the probabilities stored within the tokens as each lattice node gets propogated through the statistical model.

$$P(E_1^L, W_1^L) \simeq \prod_{i=1}^{L} P(e_1|E_{i-2}^{i-1}, W_{i-2}^{i-1}).\prod_{i=1}^{L} P(w_i|E_{i-2}^{i}, W_{i-2}^{i-1}) \qquad (5.17)$$

$$P(E_1^L, W_1^L|lattice) \simeq \prod_{i=1}^{L} P(e_1|E_{i-2}^{i-1}, W_{i-2}^{i-1}, lattice). \prod_{i=1}^{L} P(w_i|E_{i-2}^{i}, W_{i-2}^{i-1}, lattice) \tag{5.18}$$

We are therefore able to use the same approach we used in section 5.4.6 (to estimate the sequence of entities), to estimate the sequence of words and entities from a speech lattice. A trivial example is shown in appendix A.

### 5.5.2 Efficient Viterbi Decoding of Lattices

Similar to decoding named entities in ordinary text, speech lattice decoding does not require all possible paths to be evaluated. The word lattices from Cambridge University are n-gram lattices; that is, once a context of n words is established for any given time stamp, only the most probable sequence needs to be stored, or passed onwards in the token passing algorithm. If paths through a 3-gram word lattice were being examined and, at a particular time stamp, the following starts of sentence were possible:

- so he said to them (log prob = -14.2)
- so they said to them (log prob = -14.2)
- and he said to them (log prob = -14.3)
- and harry$_{PERSON}$ said to them (log prob = -14.3)
- so he said to them (log prob = -14.1)
- so he said to it (log prob = -17.9)

then only the last two sentences

- so he said to them (log prob = -14.1)
- so he said to it (log prob = -17.9)

need to be propogated. As illustrated, it is possible to have multiple sentences with exactly the same history, but for those with differing histories only those which differ in the immediate context need to be propogated. In the lattice FSM system that we are using, uniqueness is essential. If therefore the immediate context contains differing entities but the same words, both examples need to be propogated.

In practice, we also pruned if there was a large difference between the log probabilities. This was purely for time and memory efficiency to prevent strings like "$a_{PERSON}$ $a_{PLACE}$ $a_{ORGANIZATION}$ a $a_{PERSON}$ $a_{PLACE}$" being propogated. The 3-gram nature would have only discarded those sequences that would have ended "a $a_{PERSON}$ $a_{PLACE}$", which were less likely than this. If, however, the sequence is possible in the lattice, without pruning, the sequence will be followed to completion. Sequences like "$a_{PERSON}$ $a_{PLACE}$ $a_{ORGANIZATION}$ a $a_{PERSON}$ $a_{PLACE}$" can safely be pruned away, without any harm to the overall output of the lattice.

### 5.5.3 Experiment

An experiment was conducted to compare the results of a Viterbi search over named entity classes for a 1-best path through the lattice, with a Viterbi search over both named entity classes and all lattice paths to see if it was possible to improve named entity scores over those obtained on a 1-best path.

The experiment was also designed to establish whether the added named entity language models would improve the WER of the best transcript from the speech lattice. The three data sets that were used for this experiment were Lattice-Best Lattice, 1-Best Lattice and Speech Lattice as defined in table 3.2 in chapter 3.

The data set Lattice-Best Lattice was used as an upper bound on possible performance. Whereas the other two data sets were used for straightforward comparision of output.

All three data sets were used separately as input for the named entity recognition system, and three marked up texts were produced. For this experiment both F-scores and WER were recorded for each named entity transcription. F-scores and WER were calculated by comparison with the manual transcript (Manual Key). The best possible path, Lattice-Best Lattice, did not have a WER of 0%.

| Data Set | F-Score | WER |
|---|---|---|
| Lattice-Best Lattice | 78.87 | 5.4 |
| 1-Best Lattice | 74.04 | 19.7 |
| Speech Lattice | 74.34 | 20.1 |

Table 5.6: *Results of experiment comparing named entity recognition from word lattices and 1-best transcripts.*

### Results

The results for the three named entity evaluations are shown in table 5.6. Figure 5.9 shows the improvement in F-Scores as a result of using the word lattices rather than the 1-Best transcription of the lattices.

From figure 5.10 we can see how the WER is affected by this improvement in F-Score. The result confirms the hypothesis that named entity F-Scores can be improved without improving WER, and shows how a trade-off between F-Score and WER may be possible.

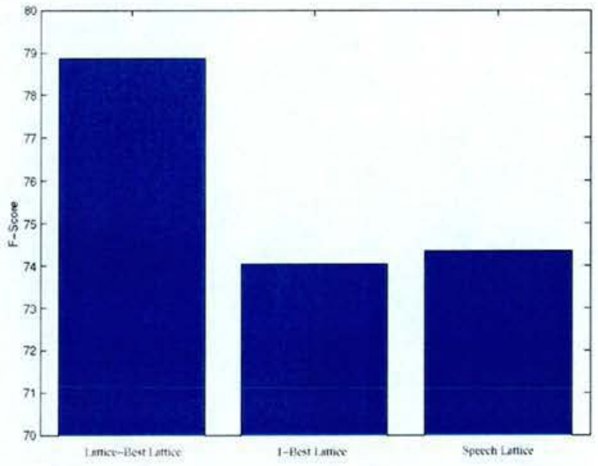


Figure 5.9: *Bar chart comparing the F-Scores for named entity recognition from word lattices and 1-best transcripts.*

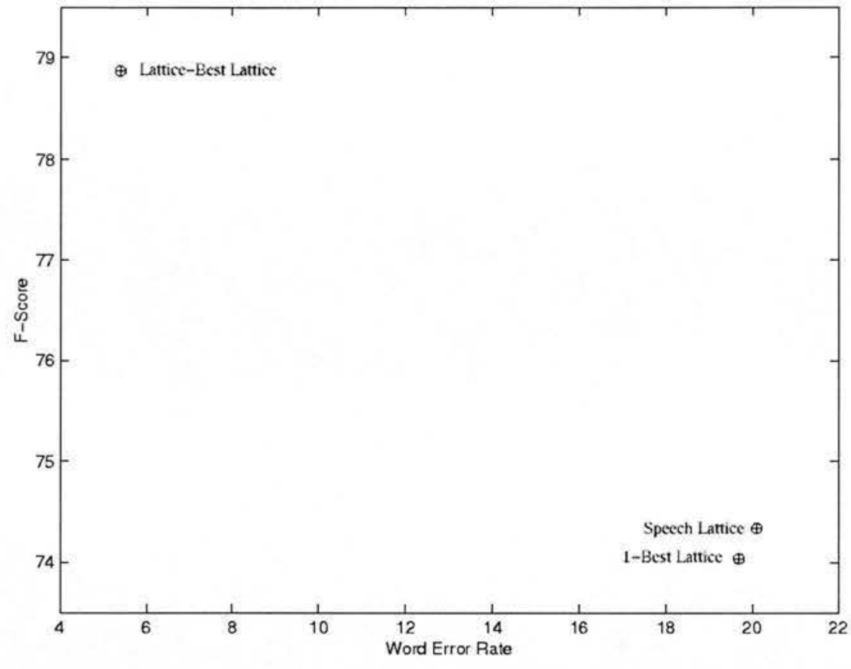


Figure 5.10: *Graph comparing the F-Scores for named entity recognition with the WER of the text generated by word lattices and 1-best transcripts.*

### 5.5.4 Conclusion

By a Viterbi search for words and named entities using word lattices, it has been shown that it is possible to obtain an improvement in named entity F-scores compared with a 1-best transcription of the same word lattices. We have also seen that this improvement in F-scores has corresponded with a decline in WER.

It follows that, although there is a general correlation between F-score and WER, it is possible to improve the one at the expense of the other. This is not to argue that degrading the quality of recognition is a sensible way of improving F-score but rather that the relationship between WER and F-score is not necessarily linear. We would suggest that the strong general correlation between F-score and WER does not imply that the only way to improve in F-score is to improve in WER.

The results suggest that there are limits on how far it is possible to improve F-score by trading the WER off in exchange for F-score.

Our conclusion that the relationship between F-score and WER is not necessarily linear is similar to the case of precision and recall. Although a comparison of systems presenting results in F-score would undoubtedly show that generally precision and recall have similar values, and one might be tempted to conclude therefore that there is a linear relationship (with unit gradient) between them, there is a known trade-off curve between precision and recall (figure 5.11) which has been shown to exist in almost all cases (Buckland & Gey 1994). We have shown that a trade off is similarly seen between WER and F-score, when the priority is F-score rather than WER.

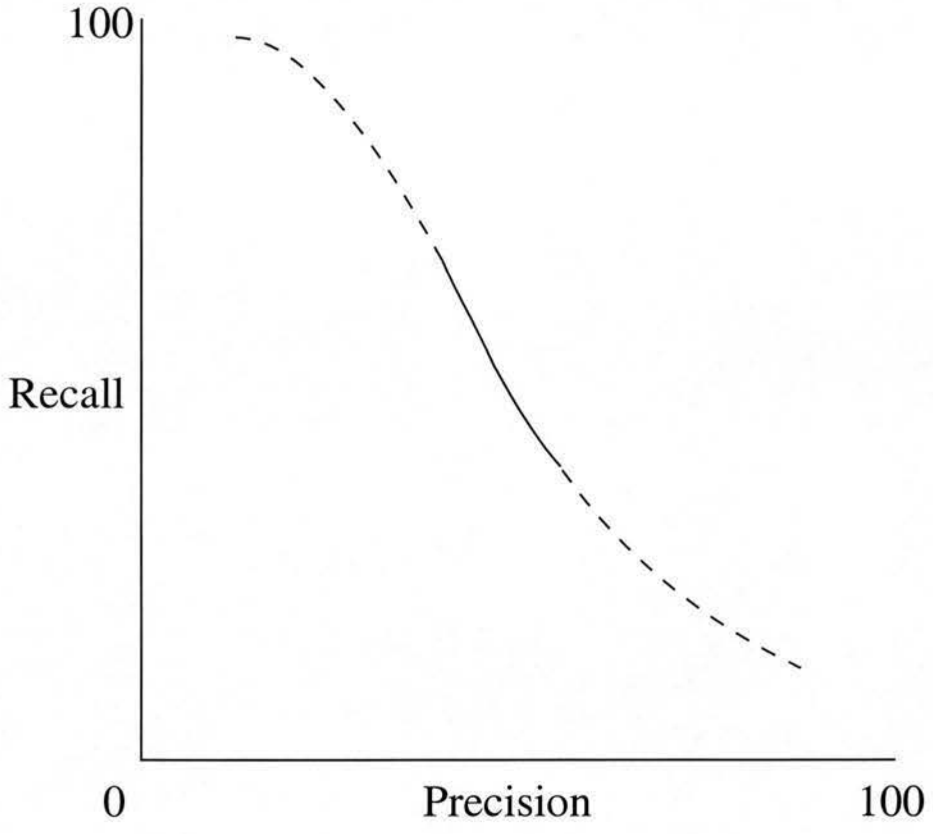


Figure 5.11: *Graph of the relationship between Recall and Precision, reproduced from (Cleverdon 1972).*

Contrary to the general trend of information extraction from speech, we conclude that it may be possible for named entity recognition to improve while WER increases - as a result of using speech rather than transcripts of speech.

## 5.6 Recognition and vocabulary errors

In chapter 3 we noted that not all correct paths were possible with the given lattices - hence the calculation of Lattice-Best lattices with a 5.4% WER. There are two reasons for this deficiency: "recognition" and vocabulary.

By "recognition" we refer to standard errors relating to insertion, deletion and substitution. The cause of these errors may lie in a fault of the speaker (mispronouncing words, etc), in a fault of the recording (background noise, etc) or, more often, in the a fault of the speech recogniser.

The *effect* of a vocabulary error will be the same as a standard "recognition" error (i.e. insertion, deletion, substitution or any combination), but the *cause* is different. A vocabulary error is caused by the speaker using a word that the speech recogniser doesn't know. For example if the speaker was quoting a verse from the Bible - Isaiah 8:1 (NIV)

> The Lord said to me, "Take a large scroll and write on it with an ordinary pen: Maher-Shalal-Hash-Baz."

The speech recogniser could not get the name "Maher-Shalal-Hash-Baz" correct. Instead, there would be something wrong with all sentence possibilities within the word lattice.

As previously stated, the size and content of the speech recogniser vocabulary used to generate the word lattices was unknown. We explain below that the vocabulary was not required.

The language models that we generated from the training data (in order to predict $P(w_i|..)$ and $P(e_i|..)$) also required a vocabulary. There were a number of possibilities for selecting this vocabulary:

- use the vocabulary of the speech recogniser - this was not available because the vocabulary was unknown;
- generate a vocabulary of all words that occurred in the lattices, and use this;
- generate a list of all words that occurred in the training data, and use this; or

- generate a list of the most common words in the training data, and use this.

The first method was impossible since the speech recogniser vocabulary was unknown. The second method was possible and was essentially equivalent to the first method since it enabled the prediction of all the words in the unknown vocabulary that were required. There was, however, a fundamental flaw with this method in that some of the words in the lattices did not occur in the training data. Therefore, when the language models were trained with this vocabulary, those words which did not occur in the training data received extremely low likelihoods in the language models.

The third method proved satisfactory for the training data - but using this vocabulary did not allow the language models to predict the probability of those words which occurred within the lattices but not within the training data (or of entities given words which occurred within the lattices but not within the training data).

The final method proved the best because, although some words which occurred in the training data were no longer predicted, it was possible to generate probabilities of unknown words. Wherever the word in the lattice was not in this vocabulary, the probability of the unknown word (labelled <UNK>) was used instead. Therefore, although the lattice words were known in advance (equivalent to knowing the speech recogniser vocabulary), results showed that it was far better to ignore this information when training the language models, in order to obtain a reasonable estimation of an unknown word.

## 5.7 Conclusions

In this chapter we have first investigated the use of CART to replace the $P(e_i|E_{1+i-n}^{i-1})$ and found that, although there was an improvement, the improvement was small for the amount of work required. Using CART required not only a lot of training, but having trained the trees, there was still a large computational overhead per experiment which was avoided without it. We therefore decided to abandon CART. (We return to it briefly in section 8.2.2 when discussing possible future work.)

Second, we introduced the idea of using a hybrid approach. Although results using the hybrid approach are good, there is no easy way of converting the hybrid approach to work with word lattices. Whilst writing rules for text is an established task, writing rules for word lattices is not - even the mark up of word lattices is non-trivial.

We have considered the traditional method of estimating a joint probability of two sequences and have developed a different method, which requires fewer independence assumptions. We have shown that this method provides better results than the traditional method.

Finally we have adjusted the system so that it is suitable for use with word lattices. Although the rule-based element of the hybrid system could not easily be converted to work on word lattices, we showed how the statistical component could be converted, and furthermore showed how the statistical component produced better results on word lattices than it did on the single best transcription of the lattice. One of the issues with which we were faced when moving from transcriptions to word lattices was the issue of vocabulary size - there will always be a far greater number and variety of words in a lattice than there will be in the relevant transcript. We addressed a number of issues relating to the vocabulary size in section 5.6.

The different method of estimating the joint probability has been shown to be applicable to both text and word lattices, but particularly to word lattices - where it is clear that the task is to find the joint probability of two sequences. (For words, the original task is to find the conditional probability of one sequence given another - although finding the conditional probability is generally still regarded as a joint probability calculation.) The improved statistical system for lattices is not only better than the original statistical system based on lattices; it is also better than the old statistical system based on automatic speech recognised transcripts and the new statistical system based on automatic speech recognised transcripts. Furthermore, when the improved statistical system is run on the manual lattices, it provides a better F-score than the hybrid system provided on manual speech transcripts.

# Chapter 6

# Discriminative Language Models

In this chapter we introduce the concept of discriminative language models and their application to the thesis. In the earlier chapters we described the eight language models: People, Place, Organization, Time, Date, Percent, Monetary expression and not-an-entity. Each language model was trained on named entity specific training data using the Carnegie Mellon University (CMU) language modelling toolkit (Rosenfeld 1994). Thus the probabilities of 'James' and 'October' were highest in the People and Dates language models respectively.

In this chapter we propose a method for iteratively making the language models discriminate against all data that occurs within the training material of any other language model in order that frequently occurring words within one language model become less likely in another language model. For example if the name "James" occurs 67 times in the People training data, 4 times in the Organisation training data, and nowhere else; we would not want the not-an-entity language model to predict $P(James|..) = P(< UNK >) = P(singleton)$, since $P(< UNK >)$ may be quite high in the not-an-entity language model. Having described a way of discriminately training the data, we provide our results on a test corpus having trained our discriminative language models both on the original training data and on the held-out development set.

## 6.1 Method

In order to develop the discriminative language models we require manually marked up data. Two separate experiments were conducted: experiment 1 (section 6.3) on the original training data and experiment 2 (section 6.4) on a held out development set.

We first strip all markup from the data (either training or development) and then use the current models to predict the best transcription of the data. We then take advantage of the fact that there is a one-to-one mapping from final transcriptions[1] to the probabilities used in the generation of the final transcriptions to create the list of probabilities used. In a similar way we treat the manual text as if it had been generated from the statistical model and calculate the probabilities that would have been used to produce the manual transcription. Having created two separate lists of probabilities, one corresponding to the manual transcript and the other corresponding to the hypothesised transcript, we compare the lists of probabilities.

If a probability occurs in the list corresponding to the manual transcription (hereafter 'manual probabilities') which does not occur in the same place[2] in the list corresponding to the hypothesised transcription (hereafter 'hypothesised probabilities'), it is assumed that the language models do not estimate this probability as highly as they should. Similarly, if a probability occurs in the hypothesised probabilities but does not occur in the same place in the manual probabilities, it is assumed that the language models are over-estimating this probability. By comparing the complete lists we are able to establish a superset of the set of incorrect probabilities.[3] We then adjust the language model probabilities listed in the superset by a predetermined amount and renormalise the language models. The process is repeated until a predetermined condition is met. The process is illustrated in figure 6.1. An explanation is given by way of example in section 6.1.1.

[1]Final transcriptions refer to the output of model

[2]The same place once the two lists have been dynamically aligned.

[3]The mathematical definition of 'superset' is used, referring to a set which contains a set. Mathematically, therefore, all sets are supersets of themselves since they contain themselves, and all sets are supersets of $\phi$ the empty set.

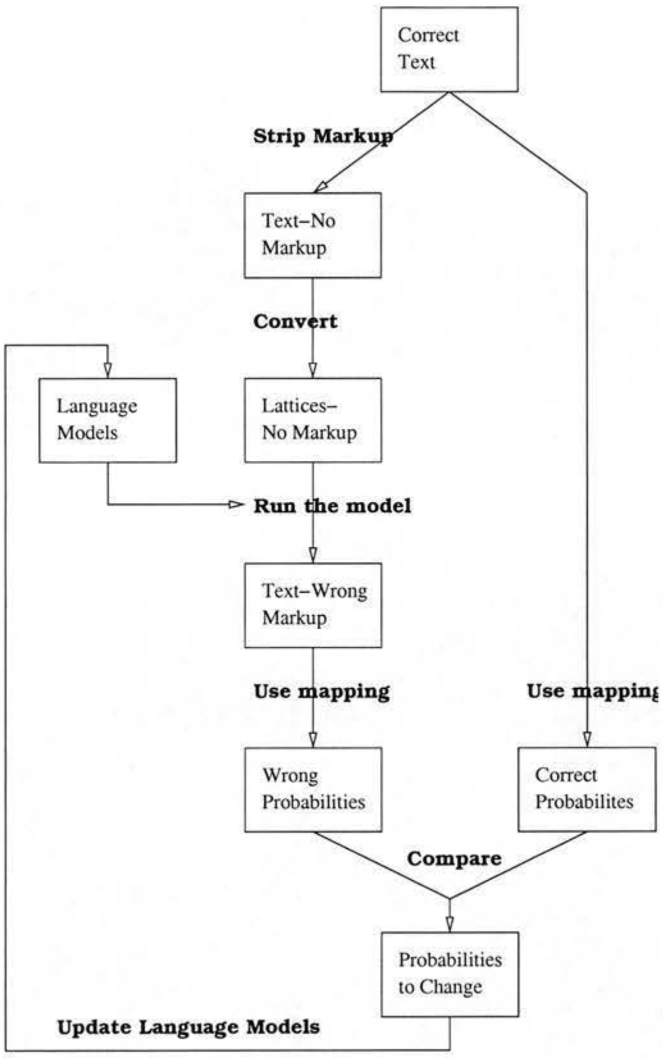


Figure 6.1: *Schematic diagram of language model iteration.*

### 6.1.1 Example

We consider two transcriptions: (i) a manual transcription, and (ii) a hypothesised transcription to compare with the manual transcription.

*(i) The manual transcription:*

```
..the stock market rose by <numex type='percent'>fifteen percent</numex>
the <enamex type='organisation'>dow</enamex>..
```

*(ii) The hypothesised transcription:*

```
..that stock market rose by <numex type='percent'>fifty percent</numex>
```

| Estimating | Probability | Language Model |
|---|---|---|
| Word | $P(the\|..)$ | `not-an-entity` |
| Entity | $1-\sum_{e} P(e\|..,the)$ | `not-an-entity` |
| Word | $P(stock\|..,the)$ | `not-an-entity` |
| Entity | $1-\sum_{e} P(e\|the,stock)$ | `not-an-entity` |
| Word | $P(market\|the,stock)$ | `not-an-entity` |
| Entity | $1-\sum_{e} P(e\|stock,market)$ | `not-an-entity` |
| Word | $P(rose\|stock,market)$ | `not-an-entity` |
| Entity | $1-\sum_{e} P(e\|market,rose)$ | `not-an-entity` |
| Word | $P(by\|market,rose)$ | `not-an-entity` |
| Entity | $P(<\texttt{per/}>\|rose,by)$ | `not-an-entity` |
| Word | $P(fifteen\|<\texttt{s}>)$ | `percent` |
| Entity | $1-P(<\texttt{/s}>\|<\texttt{s}>,fifteen)$ | `percent` |
| Word | $P(percent\|<\texttt{s}>,fifteen)$ | `percent` |
| Entity | $P(<\texttt{/s}>\|fifteen,percent)$ | `percent` |
| Word | $P(the\|by,<\texttt{per/}>)$ | `not-an-entity` |
| Entity | $P(<\texttt{ORGANIZATION}>\|<\texttt{per/}>,the)$ | `not-an-entity` |
| Word | $P(dow\|<\texttt{s}>)$ | `<ORGANIZATION>` |
| Entity | $P(<\texttt{/s}>\|<\texttt{s}>,dow)$ | `<ORGANIZATION>` |

Table 6.1: *The probabilities associated with the manual transcription.*

`<enamex type='person'>dow</enamex>..`

Since there is a bijection from transcriptions to paths through the statistical model it is possible to determine the probabilities which correspond to any given transcription. The probabilities that correspond to the manual transcription above are shown in table 6.1 and the probabilities used to generate the hypothesised transcription are shown in table 6.2.

It is therefore a simple task to obtain a superset of which probabilities are too high and which are too low. By subtraction of the hypothesised transcription probabilities (table 6.2) from the manual probabilities (table 6.1), we obtain a superset of the list of

| Estimating | Probability | Language Model |
|---|---|---|
| Word | $P(that\|..)$ | `not-an-entity` |
| Entity | $1 - \sum_e P(e\|..,that)$ | `not-an-entity` |
| Word | $P(stock\|..,that)$ | `not-an-entity` |
| Entity | $1 - \sum_e P(e\|that, stock)$ | `not-an-entity` |
| Word | $P(market\|that, stock)$ | `not-an-entity` |
| Entity | $1 - \sum_e P(e\|stock, market)$ | `not-an-entity` |
| Word | $P(rose\|stock, market)$ | `not-an-entity` |
| Entity | $1 - \sum_e P(e\|emarket, rose)$ | `not-an-entity` |
| Word | $P(by\|market, rose)$ | `not-an-entity` |
| Entity | $P(<\texttt{per/}>\|rose, by)$ | `not-an-entity` |
| Word | $P(fifty\|<\texttt{s}>)$ | `percent` |
| Entity | $1 - P(<\texttt{/s}>\|<\texttt{s}>, fifty)$ | `percent` |
| Word | $P(percent\|<\texttt{s}>, fifty)$ | `percent` |
| Entity | $P(<\texttt{/s}>\|fifty, percent).P(<\texttt{PEOPLE}>\|by, <\texttt{per/}>)$ | `percent, not-an-entity` |
| Word | $P(dow\|<\texttt{s}>)$ | `<PEOPLE>` |
| Entity | $P(<\texttt{/s}>\|<\texttt{s}>, dow)$ | `<PEOPLE>` |

Table 6.2: *The probabilities associated with the hypothesised transcription.*

probabilities required to be altered together with a direction (either increase or decrease) as shown in table 6.3.

This method allows us to find a superset of the probabilities that are incorrect. Possibly all of the probabilities within this superset are incorrect, but potentially only some of the probabilities are incorrect. It is hoped that, by iteratively repeating the process, those probabilities which are altered in the wrong direction will be corrected.

To illustrate this point we can consider the latter part of the sentence. If the only problem that caused the generation of e=Person to correspond to w=dow is that the $P(<org/> \mid <per/>, the)$ was too low, then, when we updated the language models to correct this, a superset of this probability[4] would be updated; in particular, the superset contained in table 6.4. It is hoped, however, that by iteratively calling the process the probabilities which should not have been updated (those in the superset which were already correct) will be returned to their correct values. As soon as one of these now-incorrect probabilities cause the model to make an incorrect decision, they will appear in either one or the other of the hypthesised probability list and the manual probability list, and should therefore be updated back to the original probability.

## 6.2 Factors to be considered

There were a number of factors which needed particular attention during these experiments. These are dealt with in the following subsections.

### 6.2.1 Dealing with new probabilities

In the language models, there are some probabilities which are calculated by backoff procedures. Consequently, not all trigrams which will occur in a new data set will exist within the language models. It is possible for the language model to estimate all trigram probabilities by using the backoff strategy, but, having estimated the probability, the language model does not store this number for future reference, so it cannot be updated. But, if a probability is not explicitly defined within a language model, how can this

[4]Technically, a superset of the set containing only this probability.

| Probability | Language Model | Direction |
|---|---|---|
| $P(the\|..)$ | `not-an-entity` | Increase |
| $P(that\|..)$ | `not-an-entity` | Decrease |
| $1 - \sum_e P(e\|.., the)$ | `not-an-entity` | Increase |
| $1 - \sum_e P(e\|.., that)$ | `not-an-entity` | Decrease |
| $P(stock\|.., the)$ | `not-an-entity` | Increase |
| $P(stock\|.., that)$ | `not-an-entity` | Decrease |
| $1 - \sum_e P(e\|the, stock)$ | `not-an-entity` | Increase |
| $1 - \sum_e P(e\|that, stock)$ | `not-an-entity` | Decrease |
| $P(market\|the, stock)$ | `not-an-entity` | Increase |
| $P(market\|that, stock)$ | `not-an-entity` | Decrease |
| $P(fifteen\|< \mathtt{s} >)$ | `percent` | Increase |
| $P(fifty\|< \mathtt{s} >)$ | `percent` | Decrease |
| $1 - P(< /\mathtt{s} >\|< \mathtt{s} >, fifteen)$ | `percent` | Increase |
| $1 - P(< /\mathtt{s} >\|< \mathtt{s} >, fifty)$ | `percent` | Decrease |
| $P(percent\|< \mathtt{s} >, fifteen)$ | `percent` | Increase |
| $P(percent\|< \mathtt{s} >, fifty)$ | `percent` | Decrease |
| $P(< /\mathtt{s} >\|fifteen, percent)$ | `percent` | Increase |
| $P(the\|by, < \mathtt{per/} >)$ | `not-an-entity` | Increase |
| $P(< \mathtt{org/} >\|< \mathtt{per/} >, the)$ | `not-an-entity` | Increase |
| $P(< /\mathtt{s} >\|fifty, percent).P(< \mathtt{PEOPLE} >\|by, < \mathtt{per/} >)$ | `percent, not-an-entity` | Decrease |
| $P(dow\|< \mathtt{s} >)$ | `organisation` | Increase |
| $P(dow\|< \mathtt{s} >)$ | `people` | Decrease |
| $P(< /\mathtt{s} >\|< \mathtt{s} >, dow)$ | `organisation` | Increase |
| $P(< /\mathtt{s} >\|< \mathtt{s} >, dow)$ | `people` | Decrease |

Table 6.3: *The probabilities to alter, together with the direction in which the probabilities need to be altered.*

| Probability | Language Model | Direction |
|---|---|---|
| $P(<\texttt{org/}>\|<\texttt{per/}>, the)$ | `not-an-entity` | Increase |
| $P(</\texttt{s}>\|fifty, percent).P(<\texttt{peo/}>\|by, <\texttt{per/}>)$ | `percent, not-an-entity` | Decrease |
| $P(dow\|<\texttt{s}>)$ | `organisation` | Increase |
| $P(dow\|<\texttt{s}>)$ | `people` | Decrease |
| $P(</\texttt{s}>\|<\texttt{s}>, dow)$ | `organisation` | Increase |
| $P(</\texttt{s}>\|<\texttt{s}>, dow)$ | `people` | Decrease |

Table 6.4: *The superset of* $P(< org/ > \mid < per/ >, the)$.

probability be updated within the language model? We illustrate this problem with a fictional example. We consider the sentence

... farmers have been having difficulties with wild cats in mozambique ...

The markup is incorrect because 'mozambique' has not been labelled as a location.

By using the method described in section 6.1, a superset of the set of incorrect probabilities will be generated. One of the probabilities within that superset would be $P(< pla/ > |cats, in)$ within the not-an-entity language model and it would be labelled that we should increase this probability. The problem only arises when $P(< pla/ > |cats, in)$ is not explicitly in the not-an-entity language model and instead is calculated by backing-off. How do we alter the probability of the trigram "cats in <pla/>", if the trigram doesn't exist within the not-an-entity language model?

One possibility is simply to add the probability to the language model. This method would certainly address the immediate problem that "cats in <PLACE>" had too low a probability. It would, however, probably lead to an overtrained language model which would not generalise well to new data. Indeed, the reason that many trigrams do not explicity occur within the language models is that including them at training time leads to bad generalisation later.

An alternative method would be to update the constituent probabilities that the language model uses to predict the probability of the trigram; for example, to update back-off weight(cats, in) (hereafter $bo(cats, in)$) and the $P(< pla/ > |in)$, if these n-grams exist within the language model. If these n-grams do not exist within the language model, it will be necessary to back-off further until we identify the n-grams which are used by the language model to estimate the trigram. Having found the component probabilities, these may be updated.

The question then arises, by how much should we update the component probabilities? We have been considering the case where $P(< pla/ > |cats, in)$ is not in the not-an-entity language model, but both $P(< pla/ > |in)$ and $bo(cats, in)$ are, with the back-off approximation that $P(< pla/ > |cats, in) = P(< pla/ > |in) \times bo(cats, in)$. If we have been updating trigram probabilities that do occur in the language models by $\delta$, one possibility is to update $P(< pla/ > |in) \times bo(cats, in)$ by $\delta$. There are still infinite possibilities for how to update the component parts; these are demonstrated by equations 6.1 and 6.2, since $\mu$ can take any value. The obvious value for $\mu$ is $\sqrt{\delta}$ since this will update the back-off weight and back-off probability equally.

$$bo_{new}(cats, in) = \frac{\delta}{\mu} \times bo_{old}(cats, in) \tag{6.1}$$

$$P_{new}(< pla/ > |in) = \mu \times P_{old}(< pla/ > |in) \tag{6.2}$$

We also need to decide what to do if $P(< pla/ > |in)$ is also not in the not-an-entity language model. We would simply proceed as before and split the unknown probability, as in equation 6.3.

$$P(< pla/ > |in) = bo(in) \times P(< pla/ >) \tag{6.3}$$

The new component parts are updated similarly to those in equations 6.1 and 6.2. These updates are shown in equations 6.4 and 6.5. Using the previous logic, the ideal value for $\nu$ is $\sqrt{\mu}$.

$$bo_{new}(in) = \frac{\mu}{\nu} bo_{old}(in) \tag{6.4}$$

$$P_{new}(< pla/ >) = \nu \times P_{old}(< pla/ >) \tag{6.5}$$

By using these weights, it is possible to update the language model probabilities such that any trigram probability, requested after an update of the language model with respect to that trigram probability, will have the desired output. This method will effectively update all component probabilities in such a way that the new probability will be equivalent to adding the new updated probability, while in practise all probabilities that depend on any of the constituent values will also have been updated. Equations 6.1, 6.2, 6.4 and 6.5 are all summarised in the derivation shown in equation 6.6.

$$\begin{aligned}
P(<pla/>|cats,in) &= \frac{\delta}{\mu}.bo_{old}(cats,in).\frac{\mu}{\nu}bo_{old}(in).\nu.P_{old}(<pla/>) \\
&= \frac{\delta}{\mu}.bo_{old}(cats,in).\frac{\mu}{\sqrt{\mu}}.bo_{old}(in).\sqrt{\mu}.P_{old}(<pla/>) \\
&= \frac{\delta}{\mu}.bo_{old}(cats,in).\sqrt{\mu}.bo_{old}(in).\sqrt{\mu}.P_{old}(<pla/>) \\
&= \frac{\delta}{\sqrt{\delta}}.bo_{old}(cats,in).\sqrt{\sqrt{\delta}}.bo_{old}(in).\sqrt{\sqrt{\delta}}.P_{old}(<pla/>) \\
&= \sqrt{\delta}.bo_{old}(cats,in).\sqrt[4]{\delta}.bo_{old}(in).\sqrt[4]{\delta}.P_{old}(<pla/>) \\
&= \delta.bo_{old}(cats,in).bo_{old}(in).P_{old}(<pla/>) \\
&= \delta.bo_{old}(cats,in).P_{old}(<pla/>|in) \\
&= \delta.P_{old}(<pla/>|cats,in)
\end{aligned} \tag{6.6}$$

Another alternative would be to use both of the previous methods to update the probabilities; that is, to add the probability to the language model, but also update the constituent parts of the probability - although to a lesser degree.

### 6.2.2 Dealing with unknown words

It is important to be able to deal with unknown words as many of the words which appear in new data will be regarded as unknown by the system. When the system has, for example "james horlock spoke" as input, although "james" and "spoke" will be in-vocabulary words, "horlock" will undoubtedly be out-of-vocabulary (OOV). If the system correctly classifies the named entity of OOV words, there is no problem, but if they are misclassified an issue arises as to what probabilities to adjust.

If the OOV word 'horlock' was misclassified and we want to increase $P(horlock|<s>, james)$ within the Person language model, we are able to approximate $P(horlock|<$

$s >, james)$ by $P(< UNK > \mid < s >, james)$. We therefore know what $P_{old}(horlock| < s >, james)$ was and what $P_{new}(horlock| < s >, james)$ should be.

We are therefore faced with two alternatives. The first was to add $P_{new}(horlock| < s >, james)$ to the language model.[5] This would have the desired result when this specific probability was required again, but would be bad for generalisation. The reason that this probability is not in the language model is due to the infrequency of the word. The second alternative was to update the probabilities referring to <UNK>; in this case, change $P(< UNK > \mid < s >, james)$. In cases where the n-gram is not in the language model, it is possible to use the method of section 6.2.1 above. This second approach is the one we used.

### 6.2.3 Dealing with repeated incorrect probabilities

If a probability is in the superset of those probabilities which are incorrect within the language model, we update the probability. If, however, the probability is truely wrong, we may expect the probability to occur multiple times within the list of probabilities to update per iteration of the process. We have to decide by how much to update a probability that occurs multiple times in this list.

By way of an example, suppose that "england" is so highly predictive of the location named entity tag, and that mentions of the "england cricket team" are incorrectly marked up as "<PLACE>england</PLACE> cricket team", instead of the correct markup "<ORGANIZATION>england cricket team</ORGANIZATION>. If this happens on only one occasion, we know how to deal with it. But in the case where a whole article is about the England cricket team, and therefore "england cricket team" occurs multiple times, we need to know how to update the respective probabilities. If the phrase occurs five times, we have five instances where we should reduce $P(team| < pla/ >, cricket)$. Should the probability be reduced each of the five times? Additionally, what should be done when the list of update probabilities requires both an increase and a decrease of a particular probability?

A preliminary experiment showed that updating the probabilities each time a mistake

[5] If probabilities are added to language models the language models need to be renormalised to ensure that probabilities sum to 1.

| Data set for test | Original | First iteration | Second iteration | Third iteration | Fourth iteration |
|---|---|---|---|---|---|
| Training | 97.77 | 98.35 | 98.50 | 98.86 | 98.79 |
| Testing | 74.34 | 74.10 | 74.42 | 74.32 | 74.42 |

Table 6.5: *F-scores when the language models are updated iteratively by log probability 0.05.*

occured resulted in very poor results. As an alternative to this scheme of equal addition per count, we used a tally scheme whereby, if the sum over all instances claiming that the probability should be increased, minus the sum over all instances claiming that the probability should be decreased, was positive then the probability was increased and, conversely, if it were negative the probability was decreased. This new scheme effectively counted multiple occurences the same as single occurences.

## 6.3 Experiment 1

The first experiment reused the training data to readjust the language model probabilities.

### 6.3.1 Experiment 1.1 (Multiple iterations)

In the first experiment, five iterations were made and the probabilities were adjusted by multiplying the probabilities by 1.12 and 0.89 respectively[6] for increasing and decreasing the probability. Table 6.5 shows the results on the test data, together with the changes in the accuracy on the training data. The results are plotted on the graph in figure 6.2. In each of these cases, the language models were retrained on the training data only.

It is possible to draw a number of conclusions from this initial experiment:

- Apart from improvements in the training data, the results show very slight improvement as a direct result of increased discriminative language model training on the training data.

[6]1.12 and 0.89 correspond to $\pm0.05$ in the log domain.

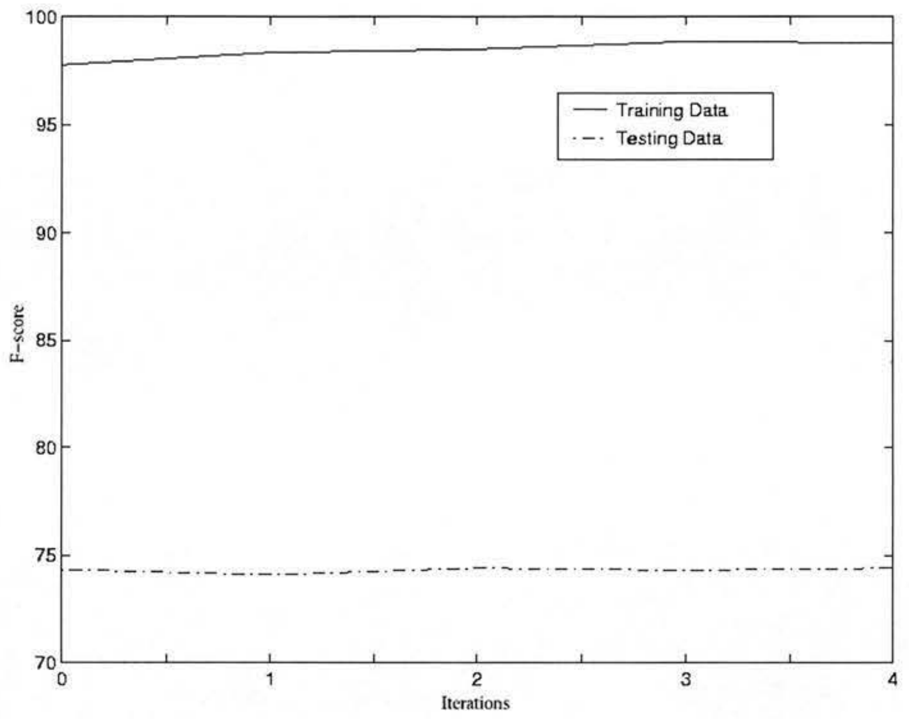


Figure 6.2: *Graph of results from experiment 1.1.*

- Improvements in results show fluctuations - in particular the second iteration results in a decrease in F-score for the development set.
- Improvements within one experiment are not necessarily reflected by improvements in the other experiments - in particular a decrease in the fourth iteration on the training data still produces an increase in F-score on the test set.

## 6.3.2 Experiment 1.2 (Comparing update weights)

A second experiment was conducted to see the effect of greater and lesser weighting of the changes to the language models. In this experiment, the amount that the probabilities were increased and decreased was varied from 1.58 to 1.06 and from 0.63 to 0.94 respectively, corresponding to variation in log probabilities from 0.2 to 0.025. In each case, the experiment was conducted with five iterations and results were noted for the training and test sets. The language models were adapted solely on the training material.

| Data set | Adjustment factor | Log factor | Original | First iteration | Second iteration | Third iteration | Fourth iteration |
|---|---|---|---|---|---|---|---|
| Training | 1.58 0.63 | ±0.2 | 97.77 | 97.98 | 98.47 | 97.72 | |
| | 1.26 0.79 | ±0.1 | 97.77 | 98.45 | 98.86 | 98.79 | 98.63 |
| | 1.12 0.89 | ±0.05 | 97.77 | 98.35 | 98.50 | 98.86 | 98.79 |
| | 1.06 0.94 | ±0.025 | 97.77 | 98.11 | 98.36 | 98.51 | 98.55 |
| Testing | 1.58 0.63 | ±0.2 | 74.34 | 74.30 | 74.27 | 72.26 | 69.72 |
| | 1.26 0.79 | ±0.1 | 74.34 | 74.27 | 74.30 | 74.02 | 74.17 |
| | 1.12 0.89 | ±0.05 | 74.34 | 74.10 | 74.42 | 74.32 | 74.42 |
| | 1.06 0.94 | ±0.025 | 74.34 | 74.44 | 74.23 | 74.41 | 74.21 |

Table 6.6: *F-Scores when the language models are updated iteratively by varying log probabilities.*

The results for this experiment are shown in table 6.6 and plotted on the graphs in figures 6.3 and 6.4. It is possible to make some observations from this experiment:

- It is possible to give too much weighting to the discriminative probabilities, which is evidenced by the adjustment of 0.2 showing far lower increase in F-score on the training data than smaller adjustment factors. This maximum weighting was to be expected, since, if the current estimation of a probability is $\epsilon$ from the correct probability, there is clearly a maximum weighting ($2\epsilon$) - such that subtracting more than this amount leaves the approximation worse than the original approximation. Also due to the relative instability of the model, once a probability is adjusted too far from the correct probability, it will have a knock-on effect on other probabilities which occur in close proximity to it in the data.

- There is a general trend for lower adjustment factor updates to give better increases in F-score on the test set, independent of the number of iterations.

- Improvement on the training data appears to follow parabolic graphs for each of the adjustment factors. The lower the adjustment factor, the more iterations before finding the global maximum - which is in keeping with expectations, providing no

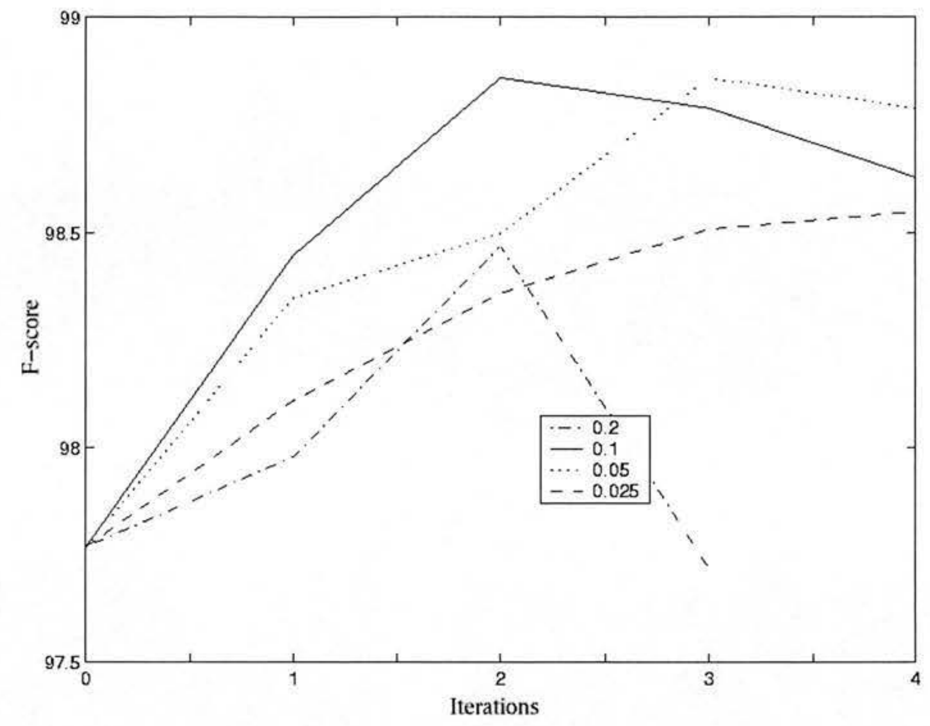


Figure 6.3: *Graph of results on the training data.*

local maxima exist and stop progress.

Table 6.7 shows how the size of the log probability used for the discriminative updates affects the number of probability updates in the following iteration. The figures in brackets show the difference from the previous iteration.

These results confirm how badly the high adjustment factor works by causing an increase in errors on the training data. They also show that the relative decline in the high probability updates is more serious than the F-scores reflected. It is interesting to note that the optimum F-score on the test set occurs when there are still a large number of probabilities to be updated.

## 6.4 Experiment 2

The second experiment used the development data to adjust the probabilities associated with n-grams. Although, in experiment 1.2, the F-score values for log factor $\pm 0.05$ were marginally better than those for the other log factors, the application of this to the

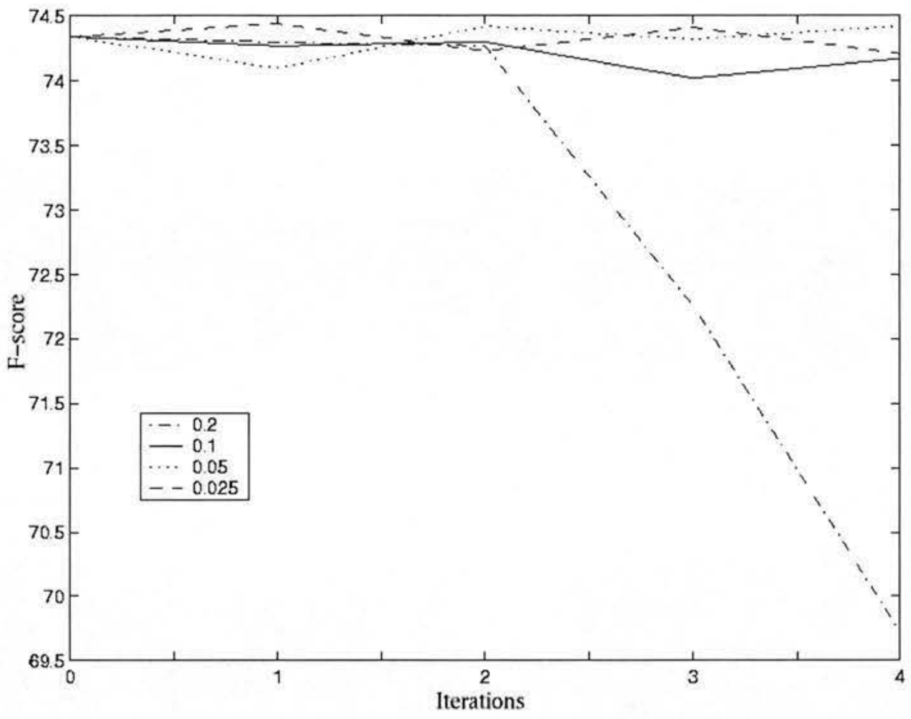


Figure 6.4: *Graph of results on the test data.*

development data produced better results with log factor ±0.25. The probabilities were therefore adjusted by 1.06 and 0.94 (equivalent to adjusting the log probabilities by ±0.025). We used the tally scheme used in experiment 1 for deciding how many times to adjust probabilities that occurred multiple times in the data and we ran only a single iteration, as that was the most effective for experiment 1. The results are presented in table 6.8.

The results of this experiment show that the decrease in F-score on the training data, which was to be expected since previously the models had been trained on this data alone, was negligible. The increase, however, on the development data, which was also to be expected, was large. The most important improvement to note was the increase in F-score on the test set. We are therefore able to conclude that this method improves named entity extraction on a test set when a development set is used for the discriminative training process.

Having run the system with these new language models on the testing set, a final evaluation of the system on the Manual Lattice was run. Our system was designed to

| Log probs | Original | First | Second | Third | Fourth |
|---|---|---|---|---|---|
| 0.2 | 8168 | 5689 (2497) | 5261 (428) | 7976 (-2715) | |
| 0.1 | 8168 | 5286 (2882) | 3899 (1387) | 3770 (129) | 3703 (67) |
| 0.05 | 8168 | 5903 (2265) | 5231 (672) | 4105 (1126) | 3852 (253) |
| 0.025 | 8168 | 6690 (1478) | 5911 (779) | 5261 (650) | 5137 (124) |

Table 6.7: *Number of changes required for each iteration with the respective log probabilities.*

| | Training | Development | Testing |
|---|---|---|---|
| Before discriminative training | 97.77 | 83.65 | 74.34 |
| After discriminative training | 97.76 | 85.29 | 74.59 |

Table 6.8: *Table of results for experiment 2.*

be run on speech word lattices, so it was not expected that the system would do as well as systems designed to be run on manual transcripts. The Manual Lattice did not contain any contextual clues (punctuation, capital letters, etc); each lattice was simply a lattice of width 1 containing a string of single case text, unlike the input used for the systems in the comparison (as explained in table 3.2). When using the discriminatively trained language models, the system produced an F-score of 84.01, in comparision with the basic system (without discriminative training) which produced an F-score 83.21; see section 5.4.6. It has therefore been shown that using discriminative training has further improved the system that had already improved on the baseline system, despite the developments being targeted at speech data rather than text.

These results are shown in the bar chart of figure 6.5, where it is clear that as well as seeing an improvement against the F-score of the system without discriminative training, there is a large improvement on both manual and speech transcripts of this method compared with both SPRACH-S and SPRACH-R (Renals et al. 1999).

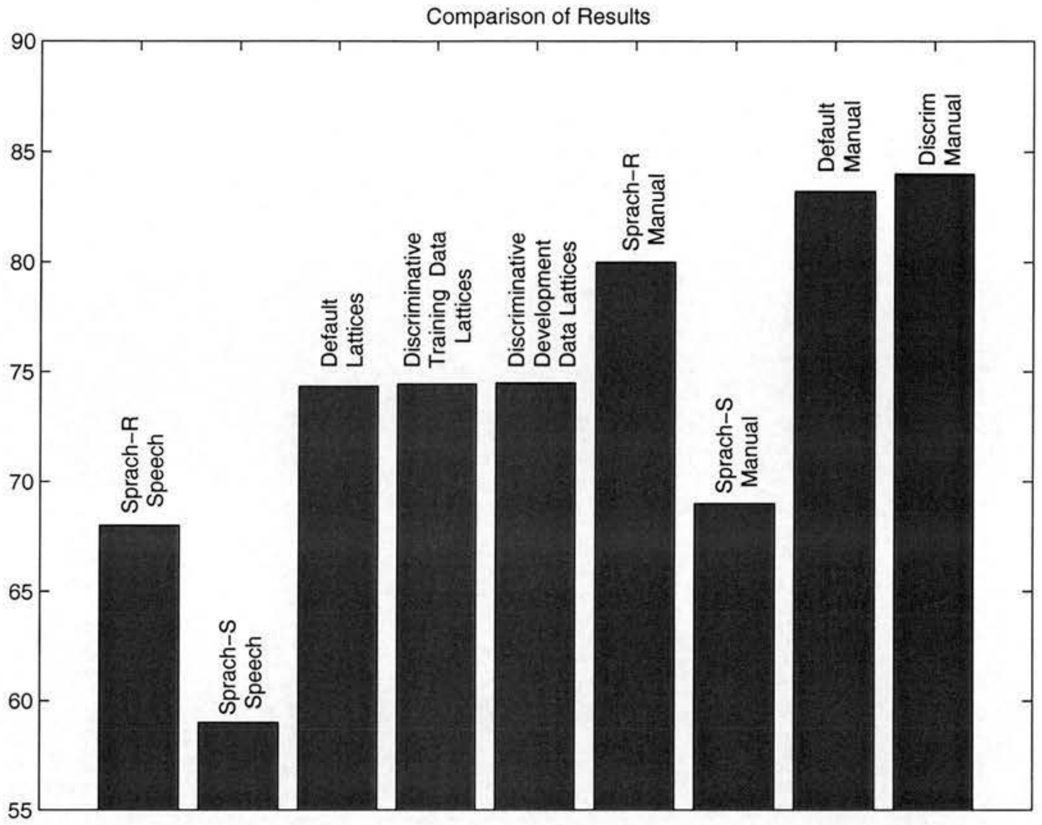


Figure 6.5: *Bar chart of results from experiment 2.*

## 6.5 Conclusions

We conclude that it is possible to use data to improve trained language models for the purpose of named entity extraction. We have shown that both the original training data and also a held-out development set can be used for this purpose.

The fact that the training data can be used to improve the results on the test set shows that the language models had not been trained to a point of over-fitting the data prior to the discriminative training.

It is likely that there is further room for improvement in these results. We have shown that a linear relation between the frequency of the probabilities to be updated and the amount by which this was updated was disadvantageous, but that always updating probabilities by the same amount, irrespective of the relevant frequencies of the evidence of probability being incorrect, proved successful. There may well be better formulae for deciding by how much to update; for example, a function of the log frequency or a sigmoidal function. A further experiment would be to discard any probabilities that only occur once in the list of probabilities to update - on the grounds that, if the probability were truly incorrect, we would expect it to appear in multiple contexts.

# Chapter 7

# Using part-of-speech in word lattices

In this chapter we show a method for using part-of-speech (POS) tags to improve named entity recognition within word lattices. We present a method for POS tagging a word lattice and then a means of utilising the POS tags within the lattice to improve named entity F-scores. This chapter is split into five sections. Section 7.1 gives a brief justification for an expectation of positive results. Section 7.2 details a mathematical derivation, showing a method for finding the named entity sequence. Section 7.3 describes how we assigned the POS tags to the lattices. Section 7.4 explains the method for generating the probabilities required by section 7.2 and relates the overall model back to the standard model used throughout the thesis. Finally, section 7.5 details the experiment and results, showing a small improvement in F-score by using POS information.

## 7.1 Why should POS tags help?

There were three main reasons why we might expect an increase in F-score as a result of using POS information.

Firstly, POS tags have been used in numerous named entity systems for text data over the years. In some instances there has been considerable value in using POS information, reflected in the F-scores of the respective systems, as we discussed in chapter 2. The successful use in other similar projects suggests a potential for use in this thesis.

Secondly, we ran the TnT (Brants 2000) POS tagger, trained on the standard Suzanne Corpus (Sampson 1995), over the training data and compared the distributions of POS tags over each of the named entity classes. The distributions are shown in figure 7.1, from which it is apparent that there is great variation over the different classes, even if some classes show resemblances to others. The differing distributions of POS tags over the different named entity classes provides evidence that both $P(e|t)$ and $P(t|e)$ are non-uniform.

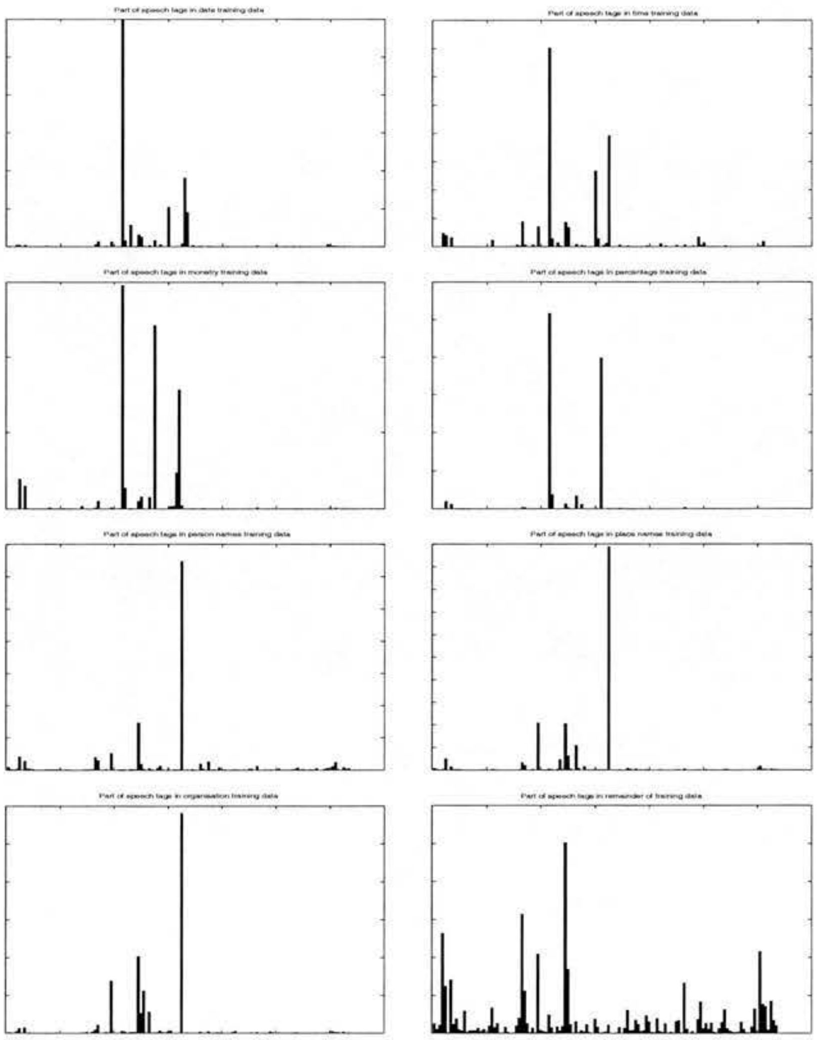


Figure 7.1: *Graphs comparing the distribution of POS tags over the different named entity data sets.*

Finally, the mathematics (with which we deal in the next section) shows how POS information can be used in the generation of the entity sequence. The existence of definitive mathematics is in itself supporting evidence, particularly since the new method makes use of additional information.

## 7.2 The mathematics

In the progression from text to word lattices in section 5.5, we noted the need for a joint probability over words and named entities. Since there was no fixed word sequence, it was incorrect to attempt to calculate $P(E_1^L|W_1^L)$. This is similarly true for adding POS information. We are not able to assume any single POS sequence - any more than we were able to assume a single word sequence. We are therefore left with the task of calculating the joint probability of entities, words and POS tags; namely $P(E_1^L, W_1^L, T_1^L)$.

In chapters 5 and 6 we considered $\arg \max_{E_1^L, W_1^L} P(E_1^L, W_1^L)$. In this chapter we need to focus on $\arg \max_{E_1^L, W_1^L, T_1^L} P(E_1^L, W_1^L, T_1^L)$. In equation 7.1, we break down $P(E_1^L, W_1^L, T_1^L)$ step by step using a similar derivation to that previously used in equation 5.5.

$$\begin{aligned}
P(E_1^L, W_1^L, T_1^L) &= P(e_1).P(E_2^L, W_1^L, T_1^L|e_1) \\
&= P(e_1).P(w_1, t_1|e_1).P(E_2^L, W_2^L, T_2^L|e_1, w_1, t_1) \\
&= P(e_1).P(w_1, t_1|e_1).P(e_2|e_1, w_1, t_1).P(E_3^L, W_2^L, T_2^L|e_1, w_1, t_1, e_2) \\
&= P(e_1).P(w_1, t_1|e_1).P(e_2|e_1, w_1, t_1).P(w_2, t_2|e_1, w_1, t_1) \\
&\quad .P(E_3^L, W_3^L, T_3^L|e_1, w_1, t_1, e_2, w_2, t_2) \\
&= P(e_1).P(w_1, t_1|e_1).P(e_2|e_1, w_1, t_1).P(w_2, t_2|e_1, w_1, t_1) \\
&\quad .P(e_3|E_1^2, W_1^2, T_1^2).P(E_4^L, W_3^L, T_3^L|E_1^3, W_1^2, T_1^2) \\
&= P(e_1).P(w_1, t_1|e_1).P(e_2|e_1, w_1, t_1).P(w_2, t_2|e_1, w_1, t_1) \\
&\quad .P(e_3|E_1^2, W_1^2, T_1^2).P(w_3, t_3|E_1^3, W_1^2, T_1^2).P(E_4^L, W_4^L, T_4^L|E_1^3, W_1^3, T_1^3) \\
&\quad \ldots \\
&= \prod_{i=1}^{L} P(e_i|E_1^{i-1}, W_1^{i-1}, T_1^{i-1}).P(w_i, t_i|E_1^i, W_1^{i-1}, T_1^{i-1}) \qquad (7.1)
\end{aligned}$$

By making the n-gram assumptions that we have been making throughout, we are able to approximate equation 7.1 with equation 7.2

$$\begin{aligned} P(E_1^L, W_1^L, T_1^L) &\simeq \prod_{i=1}^{L} P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}).P(w_i, t_i|E_{i-1}^{i}, W_{i-2}^{i-1}, T_{i-2}^{i-1}) \\ &= \prod_{i=1}^{L} P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}). \prod_{j=1}^{L} P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1}) \end{aligned} \tag{7.2}$$

The new task has therefore been simplified to solving equation 7.3.

$$\hat{E}_1^L, \hat{W}_1^L, \hat{T}_1^L = \arg \max_{E_1^L, W_1^L, T_1^L} \prod_{i=1}^{L} P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}). \prod_{j=1}^{L} P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1}) \tag{7.3}$$

To solve equation 7.3 it is necessary to estimate both $P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1})$ and $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1})$ from data.

The training data did not contain any POS tags and therefore needed adapting to allow estimates of these probabilities. After adapting the data and training new language models, as described below in section 7.3, it was still not possible to estimate the necessary probabilities directly. Approximations of these probabilities were therefore necessary.

A number of different approximations were available. We discuss the details of these possibilities later in the experiment reported in section 7.5. Essentially, however, we chose to take the average of two approximations to estimate $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1})$ as per equation 7.4, and chose to approximate $P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1})$ by making independence assumptions from two derivations of this probability as per equation 7.5.

$$P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}) \simeq \frac{1}{2} P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{2} P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1}) \tag{7.4}$$

$$\begin{aligned} P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1}) &= P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1}).P(t_j|E_{j-1}^{j}, W_{j-2}^{j}, T_{j-2}^{j-1}) \\ &= P(t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1}).P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j}) \\ &\simeq P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1}).P(t_j|E_{j-1}^{j}, T_{j-2}^{j-1}) \end{aligned} \tag{7.5}$$

The final approximation of equation 7.1 is given in equation 7.6; and the task becomes that of finding the sequences $E_1^L, W_1^L, T_1^L$ that maximise $\prod_{i=1}^{l} (\frac{1}{2}P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{2}P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})). \prod_{j=1}^{L} P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1}). \prod_{k=1}^{L} P(t_k|E_{k-1}^{k}, T_{k-2}^{k-1})$.

$$P(E_1^L, W_1^L, T_1^L) \simeq \prod_{i=1}^{l} \frac{1}{2}P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{2}P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1}) \\ . \prod_{j=1}^{L} P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1}). \prod_{k=1}^{L} P(t_k|E_{k-1}^{k}, T_{k-2}^{k-1}) \quad (7.6)$$

## 7.3 Data preparation

Having established which probabilities are required, it is necessary to estimate them. There is no problem in estimating $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1})$ or $P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1})$, since these were the probabilities that we have been estimating throughout. The problem is in estimating $P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$ and $P(t_k|E_{k-1}^{k}, T_{k-2}^{k-1})$.

There is considerable similarity between these two pairs of probabilities in that $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1})$ is similar in structure to $P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$, and $P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1})$ is similar in structure to $P(t_k|E_{k-1}^{k}, T_{k-2}^{k-1})$. It is clearly possible to estimate the unknown probabilities by creating an additional finite state machine (FSM), identical in structure to the old one, where POS tags are used instead of words. In order for such a second FSM to be created, training data will be required. This is addressed in section 7.3.1. In order to use the second FSM after training the models it will be necessary to have appropriate testing data. This is dealt with in section 7.3.2.

### 7.3.1 Training data

The original FSM was created by building eight separate language models corresponding respectively to the eight separate sets of training data - one for each named entity and one for the not-an-entity data. The training data consisted of strings such as "!SENT_START this is <org/> it's <dat/> i'm <peo/> reporting from <pla/> !SENT_END" in the not-an-entity data, "<s> a. b. c. news </s>" in the organisation data, "<s> tuesday the fifth of january </s>" in the date data, "<s> ted kopel </s>"

in the person data, and "<s> new york city </s>" in the location data[1].

To produce the necessary language models for the estimation of probabilities related to POS tags, it was similarly necessary to produce eight language models, trained respectively from POS data. As explained in chapter 2, POS taggers have accuracies close to human precision. It was therefore possible to automatically tag each respective set of training data with POS tags. Then, by removing the original words from the sets of tagged training data, the desired sets of data could be generated. The corresponding new data was in the format "STA $t_2$ $t_3$ <org/> $t_8$ <dat/> $t_{14}$ </peo> $t_{17}$ </pla> END" in the not-an-entity data, "<s> $t_4$ $t_5$ $t_6$ $t_7$ </s>" in the organization data, "<s> $t_9$ $t_{10}$ $t_{11}$ $t_{12}$ $t_{13}$ </s>" in the date data, "<s> $t_{15}$ $t_{16}$ </s>" in the person data, and "<s> $t_{18}$ $t_{19}$ </s>" in the location data.

In order to tag the training data, we used TnT (Brants 2000) described in chapter 2, trained on a modified version of the Suzanne corpus (Sampson 1995). To make it comparable in format to the input training data the Suzanne corpus needed to be modified prior to training the TnT tagger. That is, the Suzanne corpus was modified to make it lower case, punctuation was removed, sentence boundaries were replaced with !SENT_START and !SENT_END etc. We chose TnT due to it's availability and suitability for retraining on a modified corpus (speed and flexibility).

Having produced new POS training data, it was then possible to train new language models on it. These language models were able to predict the required $P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$ and $P(t_k|E_{k-1}^{k}, T_{k-2}^{k-1})$.

## 7.3.2 Testing data

Having generated the language models which are capable of estimating $P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$ and $P(t_k|E_{k-1}^{k}, T_{k-2}^{k-1})$, it was necessary to know the POS tags for the testing data. The Cambridge University lattices did not contain POS tags and without this information

[1]The data contained pseudo-words, namely 3-character abbreviations of the named entities, plus additional tags to mark start and end of phrases. The purpose of the pseudo-words is to allow, for example: the language models trained on the date data to predict $P(the| < s >, tuesday)$, the language models trained on organisation data to predict $P(< /s > |c., news)$, and the language model trained on the not-an-entity data to predict $P(< pla/ > |reporting, from)$.

the new machine was useless. It was therefore necessary to POS tag the lattices.

There were two issues that needed addressing when it came to POS tagging the lattices. The first issue was that of finding a source of POS tagged data suitable for training a POS tagger such that it could be used to label word lattices. The second issue was how to assign POS tags to a lattice.

The most logical material to be used was clearly the training material that was used to train the eight language models for the new FSM. This is because the training material was suitably tagged with the same tag set as was required, and was also the same type of data (broadcast news) - even though it was a manual transcript rather than the error-prone data of the lattices. We address the issue of formatting this training data suitably for training the POS tagger after dealing with the question of how to assign POS tags to a word lattice.

The task of assigning POS tags to a word lattice is non trivial. One possibility would be to generate all possible paths through the lattice, generate an M-best list of potential paths where M is the greatest possible for the given lattice, and then POS tag the paths. This method would allow all words to be POS-tagged as accurately as possible, and a new lattice could be generated from the POS-tagged paths. Although this method would generate the best sequence of POS tags for any given path, it would not guarantee that each word in the current lattice would receive only one POS tag. The only way for this method to generate a correct POS-tagged lattice would therefore be to expand the lattice so that only unique sequences of word-POS pairs exist. Two examples of this are given in figure 7.2 to illustrate how a new lattice may need to be expanded.

The problem with this method of POS tagging the lattice is one of efficiency. Even relatively small lattices would generate vast numbers of potential paths; that is, M is enormous. For large lattices, M would be so large that the sentences would not fit on disks, and the computing time for tagging would run into years even for a fast POS tagger like TnT. The new lattices would also be significantly larger than the old lattices. Selecting a smaller M than the largest possible would not guarantee POS tags for all words in the lattice.

The alternative method that we adopted was to consider m-grams from within the

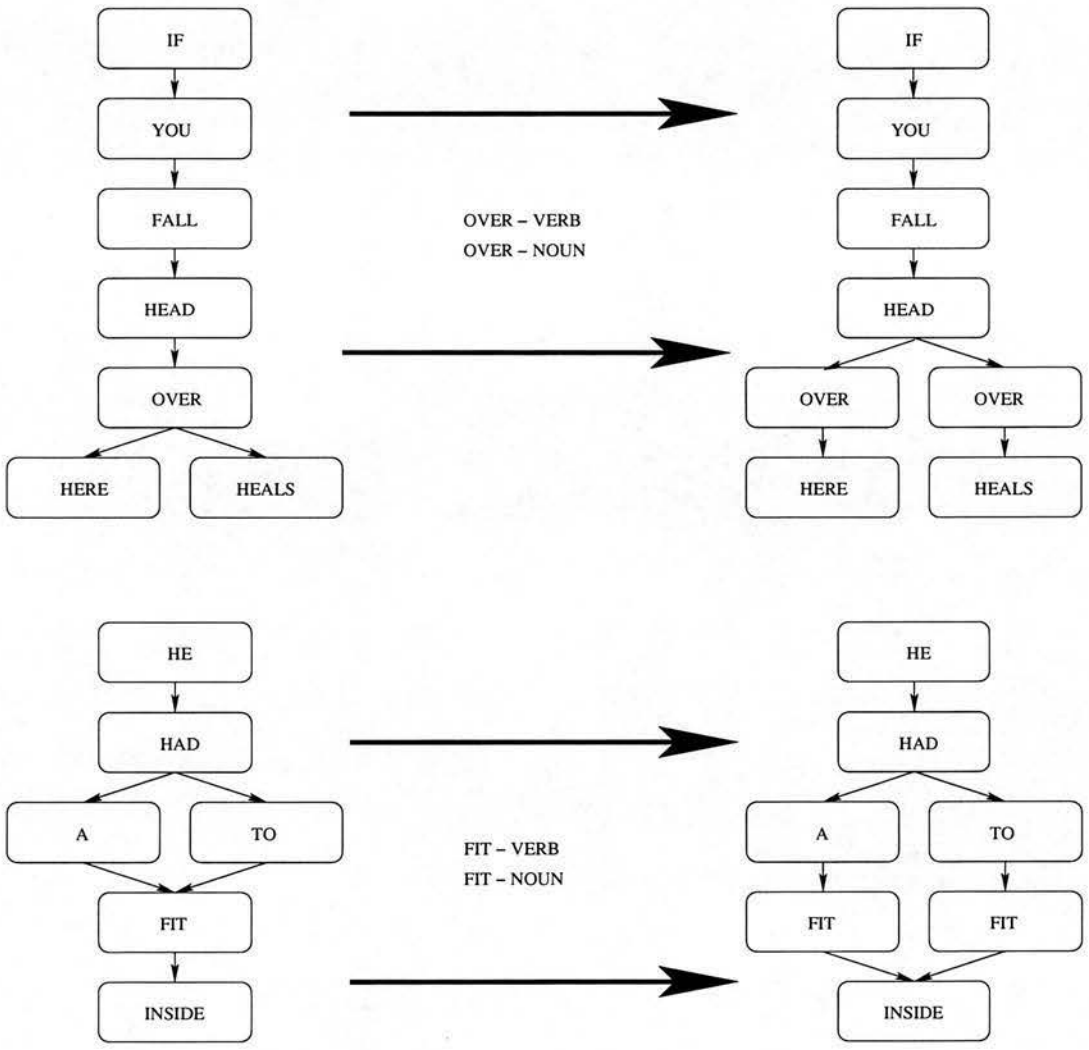


Figure 7.2: *Two examples of lattices which should be expanded when POS is added. The words 'over' and 'fit' have different POS tags (noun or verb) according to the context in which they are found.*

Step 1

The original lattice. The difference between this and the earlier similar lattice is that this lattice is a trigram lattice. All trigram paths to any node are unique! Note: necessity of exactly 2 !NULL.

```
N=14 L=15
J=0  S=0  E=1  W=!SENT_START
J=1  S=1  E=2  W=TO
J=2  S=2  E=3  W=WRECK
J=3  S=2  E=5  W=RECOGNISE
J=4  S=3  E=4  W=A
J=5  S=4  E=6  W=NICE
J=6  S=5  E=7  W=SPEECH
J=7  S=6  E=8  W=SPEECH
J=8  S=6  E=9  W=BEACH
J=9  S=7  E=10 W=!SENT_END
J=10 S=8  E=10 W=!SENT_END
J=11 S=9  E=11 W=!SENT_END
J=12 S=10 E=12 W=!NULL
J=13 S=11 E=12 W=!NULL
```

Step 2

The N (14) 3–grams are all enumerated.

```
!SENT_START

!SENT_START TO
!SENT_START TO WRECK
!SENT_START TO RECOGNISE
TO WRECK A
WRECK A NICE
TO RECOGNISE SPEECH
A NICE SPEECH
A NICE BEACH

RECOGNISE SPEECH !SENT_END
NICE SPEECH !SENT_END
NICE BEACH !SENT_END
SPEECH !SENT_END !NULL
BEACH !SENT_END !NULL
```

Step 3

POS tag the n–gram paths. Note: There is no need to POS tag paths ending with "!" words.

Step 4

Add the POS tags associated with the last word of each n–gram in step 3 to the original lattice.

Figure 7.3: *An illustration of the method for generating the POS tags of any lattice.*

lattices. Since the lattices are n-gram based, by selecting m=n we can be certain that for every node in the lattice all paths of length n-1 leading to it are the same. We can therefore view the lattice as N n-grams[2] (where N is the number of nodes in the lattice - see section 3.2.1).

Whilst N is still large for large lattices, many of these n-grams are duplicates. By sorting the n-grams alphabetically we are able to remove duplicates and therefore POS tags only need to be assigned to unique n-grams. In each instance we are only interested in the POS tag associated with the final word of the n-gram. Having obtained a POS tag for each final word of the n-gram, a POS tag has been found for each node of the lattice, and the lattice can therefore have the respective POS tags added as features. This is illustrated in figure 7.3.

[2]This is not strictly correct because the first node of any lattice is a unigram, and the children of the first node are bigrams, etc. Once enough children have been reached the remainder of the lattice can be composed of n-grams.

Using these methods we are able to generate first the training data, and then the test data for the experiment. Using the training data we are able to train a set of language models for predicting the probabilities of POS given the history, and for predicting the probabilities of entities given POS histories.

The Cambridge University word lattices were trigram based (n=3). We therefore selected m=n and produced N 5-grams (the trigram plus a start and end word) per lattice for both the training and the test data for TnT. TnT is a trigram-based part-of-speech tagger, and was therefore suitable for the task. The only problem with this method was that TnT by default uses the probability of singletons[3] to estimate the probability of unknown words. Since the training data had been modified into trigram form, each word from the training data occurred three times (once as the first word of a trigram, once as the second word of a trigram and once as the final word of the trigram) and so there were no singletons. This problem was solved by using TnT's sparse data mode which simply replaced zero frequencies by a constant.

## 7.4 The topology

As already indicated, it is possible to build two machines, one for words and one for POS. These machines would require very little adaptation and it would then be possible to find the best path through the two machines. Essentially, it would simply be a matter of having two machines operating in parallel. Since both are finite state, the combined machine would also be finite state. It is possible to visualise as per figure 7.4.

Having established that a single FSM offers a solution to the problem, we opted for using a single machine rather than a combined one. We now show one way in which it is possible to do this. Again we use the standard topology (figure 7.5) and simply reconsider what needs to happen on the transition paths and what needs to happen within the states. Previously (section 5.4.4), transitions were associated with $P(e_i|..)$, required no input and were where markup was generated; whilst states were associated with $P(w_i|..)$, required words as input, and generated words as output. In the old model, an entity specific language model was stored within each state for the calculation of the

[3]Words which occur only once in the training material.

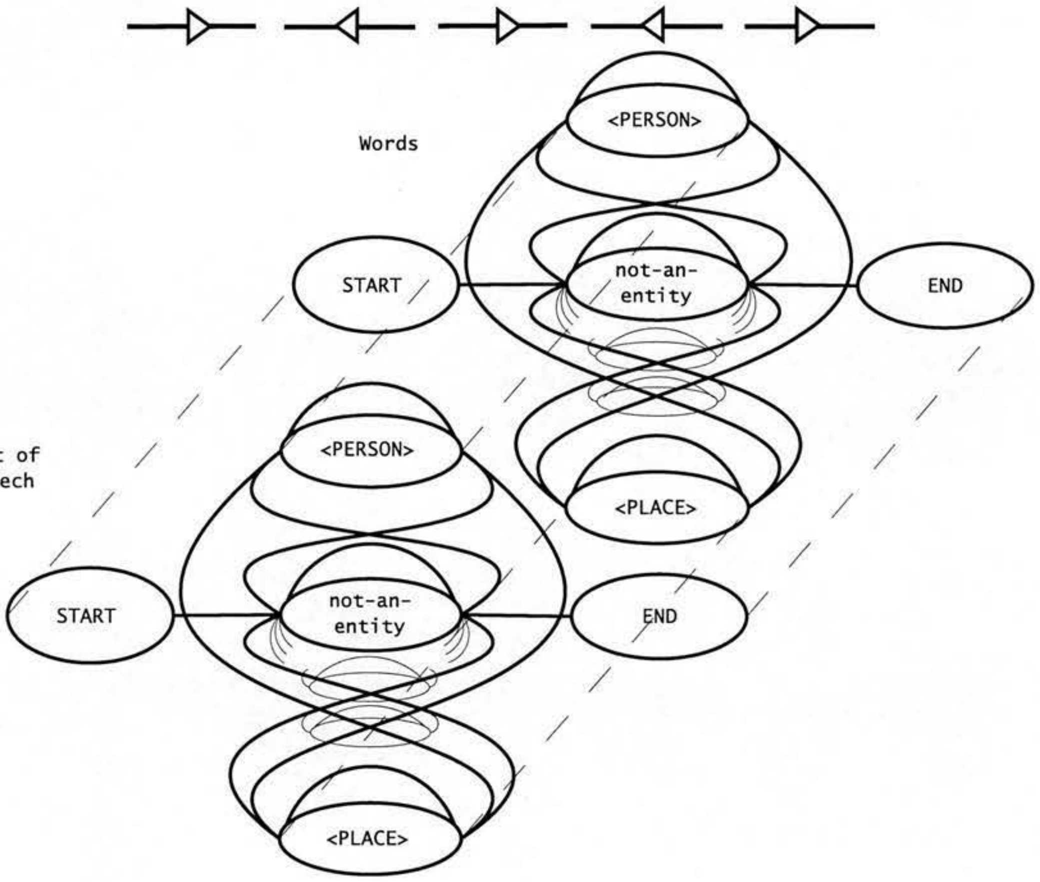


Figure 7.4: *An illustration of a method for generating the named entity-POS-word sequence from a lattice.*

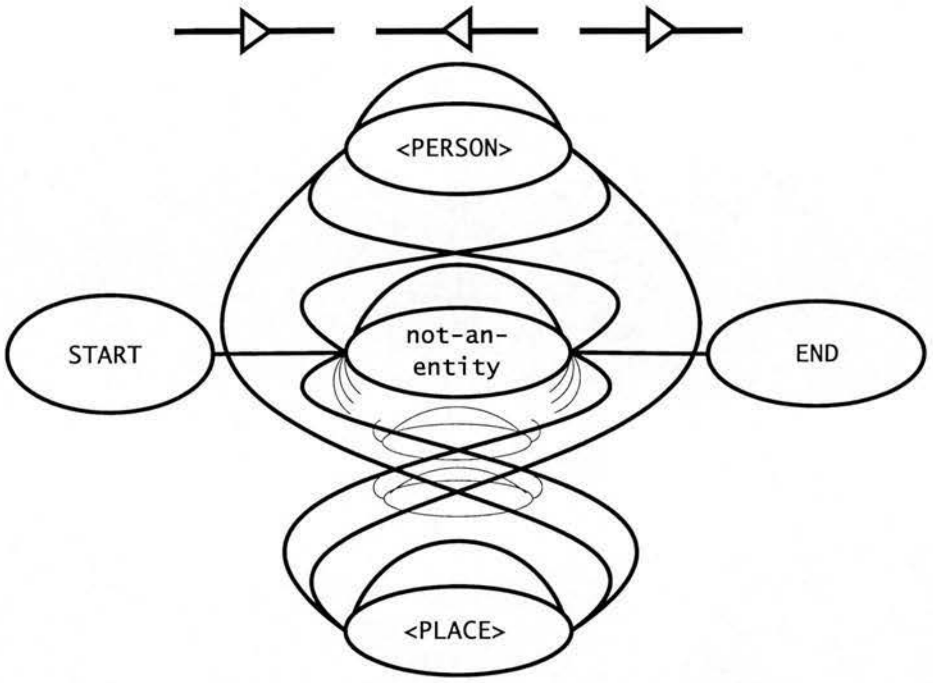


Figure 7.5: *Simplified Topology of the model.*

necessary probabilities.

In the new topology, there are slightly different probabilities to deal with; namely $P(e_i|E_1^{i-1}, W_1^{i-1}, T_1^{i-1})$ and $P(w_i, t_i|E_1^i, W_1^{i-1}, T_1^{i-1})$. Comparison with the previous topology simply makes transitions correspond to $P(e_i|E_1^{i-1}, W_1^{i-1}, T_1^{i-1})$, whereas states correspond to $P(w_i, t_i|E_1^i, W_1^{i-1}, T_1^{i-1})$. Having established this relationship, it is possible to see that transitions correspond to $\frac{1}{2}P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{2}P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$ (from equation 7.5), whilst states correspond to $P(w_j|E_{j-1}^j, W_{j-2}^{j-1}).P(t_j|E_{j-1}^j, T_{j-2}^{j-1})$ (from equation 7.4). Thus, instead of a single probability being evaluated at all transitions, each probability now needs to be calculated from two distinct probabilities. This is also true within states, where the probability needs to be calculated from two others.

The new transitions are still associated with $P(e_i|..)$, however, this probability is calculated by averaging two probabilities (namely $\frac{1}{2}P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{2}P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$); still require no input and still generate markup.

The new states are still associated with $P(w_i|..)$, however, this probability is calculated by a product of two probabilities (namely $P(w_j|E_{j-1}^j, W_{j-2}^{j-1}).P(t_j|E_{j-1}^j, T_{j-2}^{j-1})$); still require words as input and still generate words as output.

In the new model, each state is required to store two language models, the original entity specific language model stored by the state (used to predict probabilities involving words) and the new entity specific model (used to predict probabilities involving entities).

Returning to the accurate diagram of the topology (figure 7.6) used since section 5.4, which allows multi-word named entities, we are able to see how the phrase
!SENT_START the man from <ENAMEX TYPE='LOCATION'>del monte</ENAMEX> he say no !SENT_END !NULL

may be generated from the lattice

| N=12 | L=11 | | | |
|---|---|---|---|---|
| J=0 | S=0 | E=1 | W=!SENT_START | P=STA |
| J=1 | S=1 | E=2 | W=the | P=$t_2$ |
| J=2 | S=2 | E=3 | W=man | P=$t_3$ |
| J=3 | S=3 | E=4 | W=from | P=$t_4$ |
| J=4 | S=4 | E=5 | W=del | P=$t_5$ |
| J=5 | S=5 | E=6 | W=monte | P=$t_6$ |
| J=6 | S=6 | E=7 | W=he | P=$t_7$ |
| J=7 | S=7 | E=8 | W=say | P=$t_8$ |
| J=8 | S=8 | E=9 | W=no | P=$t_9$ |
| J=9 | S=9 | E=10 | W=!SENT_END | P=END |
| J=10 | S=10 | E=11 | W=!NULL | |

The path to follow in figure 7.6 is clearly the sequence of arrows FEGEGEG-CABADGEGEGEGEH. This path corresponds to the sequence of probabilities and the generation of the sequence of text shown in table 7.1. The text generated is the desired text. The POS tag sequence is also generated but this is discarded as it not required[4].

## 7.5 The experiment

In this chapter we have shown that it is mathematically possible to predict the named entity sequence from a word lattice using POS data. We have found a means of calcu-

[4]If the named entity extraction system was part of a larger system, such as an information extraction system, it is likely that the POS tags would not be discarded at this time. For the purpose of this thesis they may be discarded.

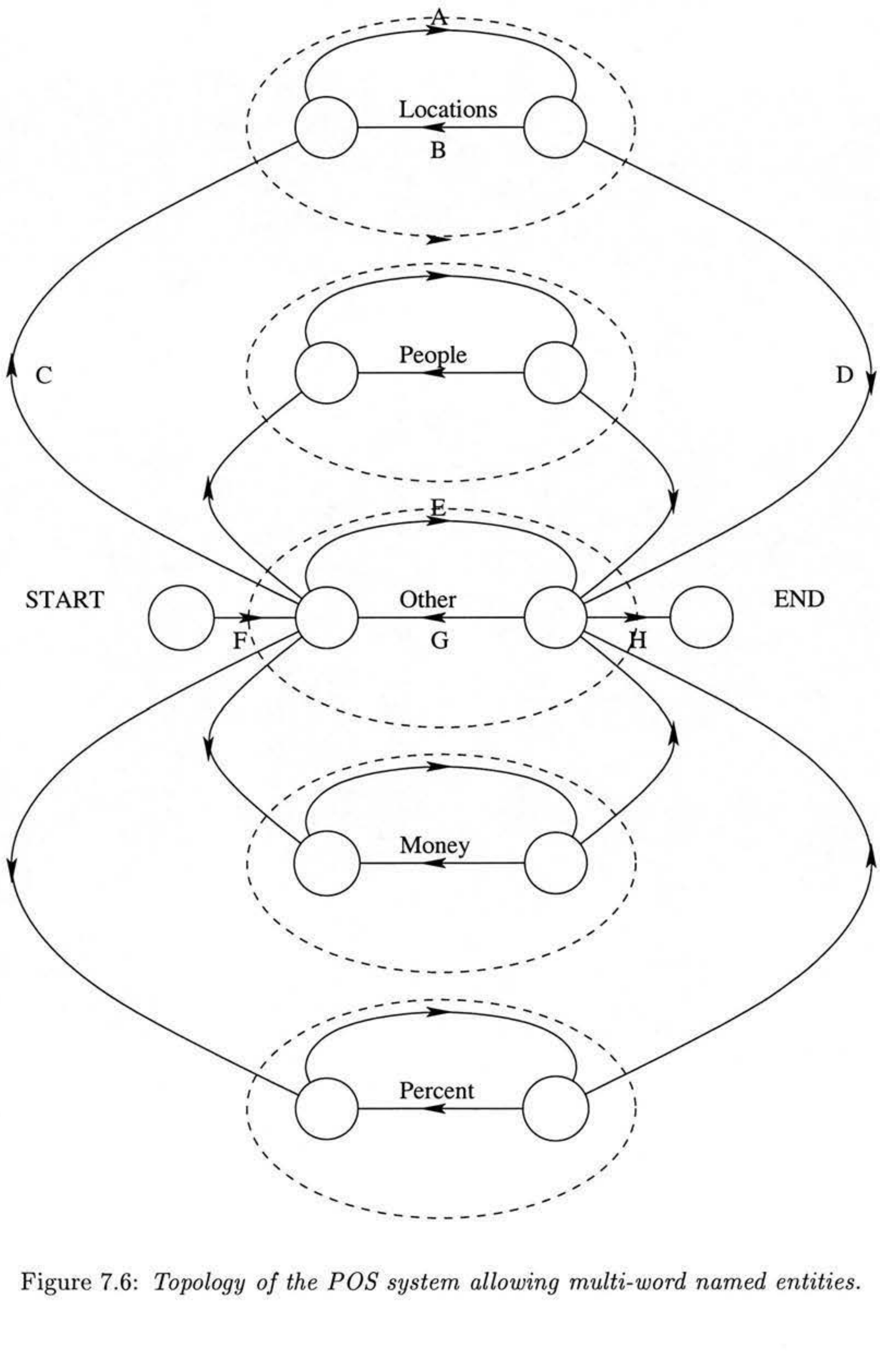


Figure 7.6: *Topology of the POS system allowing multi-word named entities.*

| Arrow | Probability | Text(POS) |
|---|---|---|
| F | 1 | !SENT_START (STA) |
| E | $P(the \| not-an-entity, !SENT\_START)$. $P(t_2 \| not-an-entity, STA)$ | the ($t_2$) |
| G | 1 | |
| E | $P(man \| not-an-entity, !SENT\_START, the)$. $P(t_3 \| not-an-entity, STA, t_2)$ | man ($t_3$) |
| G | 1 | |
| E | $P(from \| not-an-entity, the, man)$. $P(t_4 \| not-an-entity, t_2, t_3)$ | from ($t_4$) |
| G | 1 | |
| C | $\frac{1}{2}P(< PLA/ > \|not-an-entity, man, from)+$ $\frac{1}{2}P(< PLA/ > \|not-an-entity, t_3, t_4)$ | <ENAMEX TYPE ='LOCATION'> |
| A | $P(del \| location, < s >).P(t_5 \| location, < s >)$ | del ($t_5$) |
| B | $\frac{1}{2}(1 - P(< /s > \|location, < s >, del)+$ $\frac{1}{2}(1 - P(< /s > \|location, < s >, t_5)$ | |
| A | $P(monte \| location, < s >, del)$. $P(t_6 \| location, < s >, del)$ | monte ($t_6$) |
| D | $\frac{1}{2}P(< /s > \|location, del, monte)+$ $\frac{1}{2}P(< /s > \|location, t_5, t_6)$ | </ENAMEX> |
| G | 1 | |
| E | $P(he \| location, not-an-entity, from, < PLA/ >)$. $P(t_7 \| location, not-an-entity, t_4, < PLA/ >)$ | he ($t_7$) |
| G | 1 | |
| E | $P(say \| not-an-entity, < PLA/ >, he)$. $P(t_8 \| not-an-entity, < PLA/ >, t_7)$ | say ($t_8$) |
| G | 1 | |
| E | $P(no \| not-an-entity, he, say)$. $P(t-9 \| not-an-entity, t_7, t_8)$ | no ($t_9$) |
| H | $P(!SENT\_END \| not-an-entity, say, no)$. $P(END \| not-an-entity, t_8, t_9)$ | !SENT_END (END) |

Table 7.1: *The probabilities and output relating to figure 7.6.*

lating the relevant probabilities and shown that the overall model structure is similar to the structure in previous chapters (without using the POS data). We have successfully labelled the word lattices with the respective POS tags for all the nodes within the lattices. All that remains is to test whether or not the new system works more effectively than the old system.

In section 7.2, equations 7.4 and 7.5 were used to approximate the desired probabilities. The experiment we conduct uses these, together with other valid approximations, to find possible word-POS-entity sequences. Having found the sequences we then discard the POS tags (as already mentioned at the end of section 7.4), since the task at hand is to find the best sequence of words and entities, and not to find the POS sequence. F-score calculation is not affected by POS, and the scoring software would be unable to deal with it.

Speech recognisers perform calculations based on the weighted sums of log probabilitities and likelihoods. The model that has been used in chapters 5 and 6 has similarly used weights as discussed in chapter 5. The best transcripts were the result of the following weightings: 28 for the language model log probabilities within the lattices, 2 for the acoustic model log probabilities within the lattices, 14 for the $P(e_i|..)$ from the named entity model and 14 for the $P(w_i|..)$. The experiment we now conduct is to decide how the probabilities relating to POS should be weighted. We use as our figure for comparison an F-score of 74.09 - which is the figure we obtain when we do not use POS data and use the original system detailed in the previous chapters.

We would not expect an equal weighting for POS probabilities and word probabilities to render good results, since we would expect that POS probabilities are far worse predictors for cases where the words are common and where we have accurate predictors. Indeed, when equal weightings were used, experimental results showed an F-score of 67.21, a relative drop of almost 7 points of F-score - which supported our hypothesis.

Our next experiment involved weighting the combined probabilities in favour of probabilities based on words. Instead of treating $P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1})$ as the product $P(w_j|E_{j-1}^{j}, W_{j-2}^{j-1}).P(t_j|E_{j-1}^{j}, T_{j-2}^{j-1})$, we rather treated it as a weighted product[5]. Similarly, instead of treating $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1})$ as $\frac{1}{2}P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1})$ +

[5]Weights are applied in the log domain.

$\frac{1}{2}P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$, which we had done in the previous experiment, we weighted this sum as in equation 7.7 or equivalently in equation 7.8.

$$P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}) \simeq \frac{weight - 1}{weight} P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{weight} P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1}) \quad (7.7)$$

$$P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}) \simeq (1 - \frac{1}{weight}) P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}) + \frac{1}{weight} P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1}) \quad (7.8)$$

We considered a number of possible weightings which gave F-scores in the range [72.83, 72.96]. Results were improving but the F-score was still lower than the original F-scores without using POS.

Finally, we conducted an experiment to find out what would happen if we considered $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1}) \simeq P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1})$, which is equivalent to having an infinite weight for this probability, but allowing $P(w_j, t_j|E_{j-1}^{j}, W_{j-2}^{j-1}, T_{j-2}^{j-1})$ still to be treated as a weighted probability. Again we compared the weightings of 7 and 14. This time the results were respectively 74.37 and 74.80. Both results showing an improvement in F-score over the original result of 74.09.

### 7.5.1 Conclusions

We have shown that it is possible to gain improvement in F-score by using POS information within word lattices. We have demonstrated that it is critically important which approximations of probabilities are used and that sometimes the most obvious do not produce the best results. Our experiment has shown that $P(e_i|E_{i-2}^{i-1}, T_{i-2}^{i-1})$ should not be used as an approximation for $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1})$.

There are, of course, other approximations for these probabilities which have not been considered. Potentially, the estimation of $P(e_i|E_{i-2}^{i-1}, W_{i-2}^{i-1}, T_{i-2}^{i-1})$ is one of the places where using a CART tree might prove effective since this would allow the use of more information in the calculation.

In all likelihood, there is further room for improvement by using more accurate POS tagging. Such a POS tagger could be trained on a larger corpus than the Suzanne corpus, or on a more appropriate corpus - possibly the Christine corpus designed by G Sampson of the University of Sussex, which extends the Suzanne corpus to the domain of spoken English (Sampson 2000). One could also expect further improvement by finding a way of POS tagging the lattice nodes based on more context than this method has allowed.

# Chapter 8

# Conclusions and future work

## 8.1 Conclusions and contributions

This thesis has focused on named entity extraction from speech. Prior to this thesis all work on information extraction from speech had been based on transcriptions of speech; some of these these transcriptions have been manual and some the result of automatic speech recognition (ASR). The primary contribution of this thesis has been in moving from speech transcriptions to word lattices. Whereas others have used transcripts, we have progressed towards information extraction from the original speech signal by using word lattices. We have taken a fairly standard statistical model for named entity extraction from transcripts and shown that, not only is it possible to make the transition to a model using word lattices, with only minor adaptations to the overall topology of a model, but that, in doing so, it is possible to improve the resulting F-score.

When evaluating the impact of named entity extraction direct from the word lattices, the resulting transcript, containing named entities, has a higher word error rate (WER) than when the named entity extraction was based on a corresponding ASR transcript; that is, the named entity extraction component has had a negative impact on WER. This result shows an improvement in F-score working perpendicular to the general trend between WER and F-score (Horlock & King 2003*b*).

We also presented an alternative approach to finding the joint probability of words and entities. We showed that it was possible to estimate the joint probability of events without making as many independence assumptions as were previously being made. We were able to demonstrate how this new approach was still in keeping with the standard model topology, and how it was possible to predict the required probabilities using the old language models. This new method, which reduced independence assumptions, yielded an absolute improvement of over 3% in F-score.

In chapter 6 we introduced the idea of discriminatively training each of the language models involved in the standard model. The idea was to improve the results of the system without altering the overall system at all, but simply by improving each of the language models corresponding to the named entities. We described a method which allowed us to bias each of the language models against the data on which each of the other language models were trained. The new language models were trained by iteratively adjusting the probabilities which were thought to be incorrect (a superset of the probabilities that were incorrect). Small improvements in the overall F-scores were noted as a result of the discriminative language model training (Horlock & King 2003*a*).

Finally, in chapter 7 we added part of speech (POS) data to our model. We found a method for POS tagging the word lattices, and a method for finding the named entities based on the POS tags within the word lattices. We showed how it was possible to adapt the standard model topology that we had already been using to incorporate POS. Having made the adaptation to the standard model, the model remained similar in structure to its original format, but the resulting named entity tagged transcripts from the word lattices produced higher F-scores.

We have introduced a method for extracting named entities from word lattices. We have shown how this method gives a higher F-score than the same method applied to ASR transcripts of the speech data. Furthermore, we have shown how these results can be improved further by a variety of methods, including adjusting the estimates of the required probabilities, adjusting the method of training the language models that generate these probabilities, and using additional language models based on extra information (namely POS) to estimate probabilities.

A summary of all experimental results from the thesis is shown in table 8.1 and relevant bar charts plotted in figures 8.1 and 8.2.

| Experiments using the manual transcript | | |
|---|---|---|
| Experiment | Description | F-Score |
| SPRACH-R | Sheffield and Cambridge rule-based system | 69 |
| SPRACH-S | Sheffield and Cambridge statistical system | 80 |
| Baseline | The original model | 79.82 |
| Pre-Stats Rules & Model | Using lexical lookup & the model | 80.78 |
| Full Rules & Model | Lexical lookup, model & splitting entities | 83.46 |
| Redefining $P(A \cap B)$ | Statistical model after re-estimating $P(A \cap B)$ | 83.21 |
| Discriminative Retraining | Statistical model after discriminative retraining | 84.01 |
| Experiments using the speech lattices | | |
| Experiment | Description | F-Score |
| SPRACH-R | Sheffield and Cambridge rule-based system | 59 |
| SPRACH-S | Sheffield and Cambridge statistical system | 68 |
| First Experiment | Statistical model after re-estimating $P(A \cap B)$ | 74.34 |
| 1-Best Experiment | Statistical model on 1-Best transcript | 74.04 |
| Lattice-Best Experiment | Statistical model on Lattice-Best Experiment | 78.87 |
| Discriminative Retraining | Statistical model after discriminative retraining | 74.59 |
| Using POS information | Statistical model using additional POS information | 74.80 |

Table 8.1: *A table summarizing results throughout the thesis.*

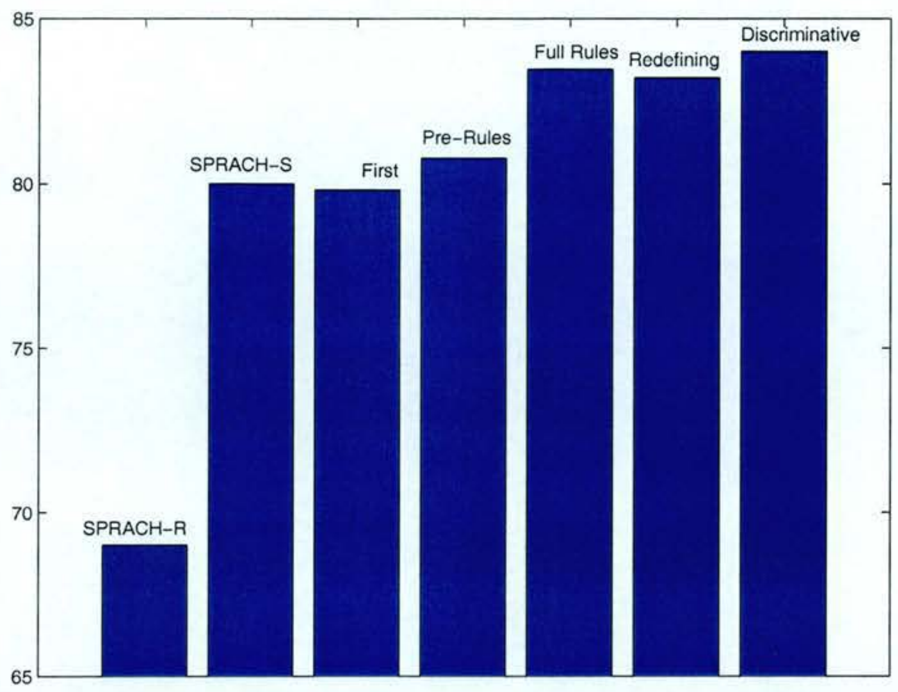


Figure 8.1: *Bar chart summarizing results for manual transcripts.*

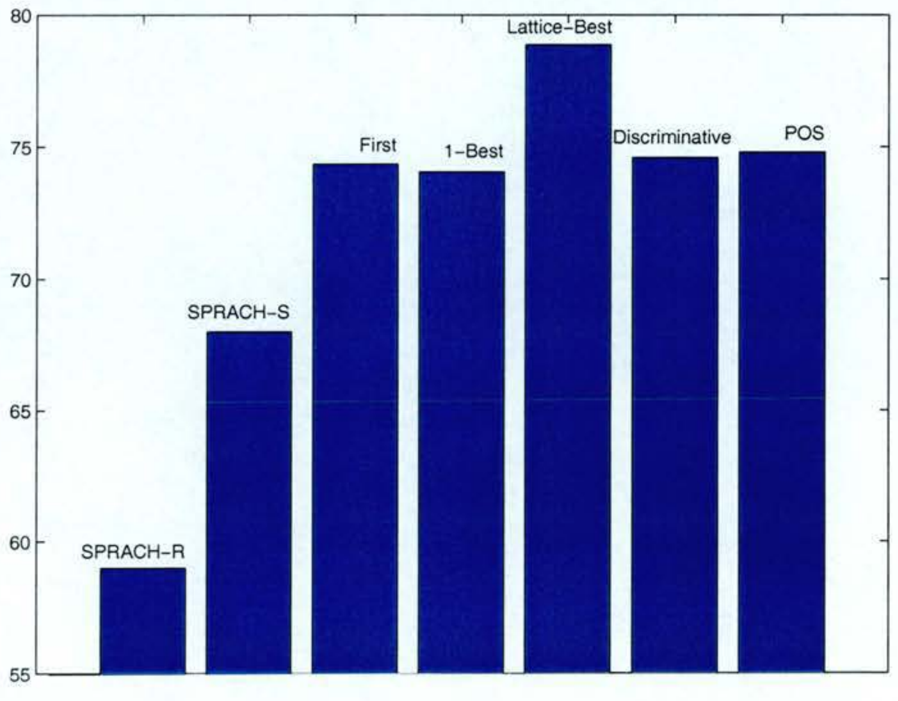


Figure 8.2: *Bar chart summarizing results for speech lattices.*

## 8.2 Future Work

### 8.2.1 Simplification of the model

Throughout this thesis a fairly standard model topology for named entity extraction has been adopted. We have, in general, made use of eight separate language models within this topology. In chapter 6 of the thesis we have focused in some depth on discriminatively adjusting probabilities within the separate language models. In chapter 7 we used pairs of language models, rather than single language models, but the topology remained constant.

An interesting simplification for future work would be to condense the language models into a single language model trained on a single pass of the training data. It would be possible simply to use an off-the-shelf POS tagger, where the tags that are used for training and testing are the named entity classes. This method would not differentiate between multiword named entities and sequential named entities. It would, however, be interesting to see how advantageous the more complex model proved over a simple model - if at all.

### 8.2.2 Extension of the model

#### Class-based smoothing

In (Palmer et al. 2000) it was shown that by using class-based smoothing it was possible to improve the F-measure obtained on a similar named entity extraction task. It is at least possible that the improvement would also apply to the model described within this thesis.

There are a number of classes which potentially lend themselves to the task of named entity recognition. The most obvious of these classes would be the class of numbers (one, two, etc) and the subsidiary classes of fractions (half, quarter, etc) and ordinals (first, fourth, etc). The classes might need to be split up into subclasses e.g. number_less_than_10, etc.

Some other classes which could be made the subject of experiment include: the months of the year (january, february, etc), various time zones (est, gmt, etc), famous people (winston churchill, martin luther king, etc), known Christian names (james, peter, etc), known surnames (brown, davies, etc).

One final class that could be considered would be that of POS. Although we have already shown improvement by using POS, if a general method for smoothing were found, a comparison between the approaches could be made.

We would expect there to be advantages from smoothing; in particular, we would hope smoothing would help generalise named entities. For example, we could find general dates even if the training data was broadcast on an uneventful date such as 17th February and the training data therefore contained many references to this date, but very few to other dates. In this case, the current method would always classify 17th February as a date, althought it may well not correctly mark 18th March as a date.

There are, however, also potential disadvantages with such classes. For example, certain dates occur with much greater frequency than others (contrast 4th July and 11th September with 29th February, etc). It may be necessary to obtain more accurate distribution statistics of the contents of the classes. This should not be difficult, however, as it simply requires large corpora which already exist; i.e. it does not require the corpora to contain named entity mark up. Finding how to combine the distribution with the classes would be a matter for research. Another potential difficulty is using classes which contain ambiguous words such as 'march'. If 'march' is classified as a month, even with an extra distribution factor as suggested in the previous paragraph, problems may occur when trying to clarify its meaning.

**Adding CART**

In chapter 5 we gave a brief description of some preliminary studies using classification and regression trees (CART). We found that CART did not give us any substantial improvement in results to those obtained without the added computational overhead involved with CART - both at train time and run time.

There are several possible reasons for the lack of improvement. These include: (i) CART may not help in named entity recognition, (ii) we might have been using the wrong features for CART, (iii) we could have been weighting CART incorrectly against n-gram language model probabilities.

CART were discarded at an early stage because it was clear that a large improvement in overall results was required and CART were not providing it. A substantial improvement has been made since the early stages, with results well in excess of seventy percent on word lattices, and it is possible that CART could now be used to improve results further.

One of the features which we were using in CART was, 'how was this word classified last time it was classified?' Although this field was a strong and accurate predictor of the next occurrence, a problem arose if the previous classification was incorrect (which was particularly likely for the first classification where there was no previous classification to enable the judgement). Consequently, if the previous classification was a strong predictor of the current word, and the first occurrence was incorrectly classified, there would be adverse knock-on effects.

We believe that there is still potential for improving the results using CART to predict the current entity and that 'how was this word classified last time it was classified?' is a good predictor of the current entity. Finding how to use this best is a matter for future research.

**Hybrid system - adding rules**

The 1998 Message Understanding Conference (Chinchor 1997) showed the benefits of having a hybrid system consisting of both rules and statistics to find named entities within text, the hybrid system winning the competition.

Early in the thesis we investigated the possibility of using rules both as precursors and as postscripts to the statistical model used for named entity extraction. Specifically we used lexicons to mark up certain words as certain entities - as a precursor to the statistical system. We also used rules to change multiple entities which had been grouped into a single entity by the early system - which was unable to determine the difference between $< PLACE > EDINBURGH < /PLACE >< PLACE > SCOTLAND < /PLACE >$ and $< PLACE > EDINBURGH\ SCOTLAND < /PLACE >$. We showed how these extra rules improved overall F-measure of the then system.

With the introduction of the use of word lattices, precursor rules were removed from the system because the markup of word lattices seemed impractical. Fig 8.3 illustrates

```
J=48  S=12  E=13  W=christopher  a=−715.32  l=−8.145
J=50  S=13  E=15  W=brown  a=−418.17  l=−2.988
J=54  S=15  E=18  W=with  a=−647.29  l=−7.167
J=55  S=15  E=19  W=went  a=−634.12  l=−6.890
J=58  S=18  E=24  W=robin  a=−897.09  l=−9.123
J=60  S=19  E=24  W=robbing  a=−912.34  l=−9.345
J=65  S=24  E=27  W=banks  a=−642.89  l=−7.645
```

Figure 8.3: *Problems with marking up word lattices, especially the word banks.*

this. Not only is it difficult to make clear that 'Brown' and 'Christopher' are part of a single named entity (potentially an avoidable problem), but some clever presentation would be required to indicate the markup of 'banks' if the chosen path through the lattice should indicate the markup of 'banks'. It would be possible to expand the lattice to make a larger lattice with such conflicts resolved. This task is, however, non-trivial.

The post-processing part of the rules was also removed from the pipeline once the statistical model was capable of distinction between repeated entities and multiword entities, since this was the sole purpose of the post-processing rules.

It is generally accepted that a few simple rules go a long way in named entity recognition, although the acceptance of the use of these rules varies. Statistically speaking, provided these rules are to be dealt with a priori, further statistical analysis is still legitimate.

One of the places we envisage a gain from rules is in cases where the task definition (Chinchor et al. 1998) is not obvious; i.e. cases where people who have not read the task definition carefully may well disagree. An example of this is shown in figure 8.4 where the correct markup may not be obvious. This situation, although not typical, is not uncommon in speech data. It is imperative that the system classifies these non-obvious cases correctly, since any alternative markup to the stated correct markup (the markup in figure 8.4) will obviously result in a reduction of F-score, even though the system may have made the same judgement that a human would have made.

If the system attempts to mark up the text in figure 8.4, where the trigram language model attempted to evaluate the probabilities associated with the first 'john', it will first consider the probability of a Person entity given the history (which will clearly be very

SPEECH: i went to hamilton with john john gillan
CORRECT: i went to <PLACE>hamilton</PLACE>
with john <PERSON>john gillan</PERSON>

Figure 8.4: *A sentence together with the correct markup of the sentence according to the task definition (Chinchor et al. 1998).*

high as that is what was intended) and then consider the probability of the word 'john', given that it is the start of a name (which similarly will be high). To expect a statistical named entity extractor to come to a decision to classify the first 'john' other than as a person, irrespective of what the task definition states, is fundamentally wrong, since the entity is predicted by these two probabilities - both of which are high.

This is one instance where it may be expected that the use of rules may aid named entity recognition. There are three ways that this could be attempted. The first and simplest would be to add a post-processing rule which looks for repeated names and changes the markup. A more advanced way would be to look for repeated names within the training data, to adjust the training data markup to reflect the more logical output (in our example "<PERSON>john john gillan</PERSON>"), and then add a post-processing rule to adjust the output from the statistical component to what the task definition requires. The third alternative, would be to find a way of spotting the repeated name situation prior to named entity tagging and adjust the input correspondingly. In our case this would involve adjusting the lattices to remove speech repairs, as was attempted in (Spilker et al. 1999).

There are almost certainly other occasions where the actual task definition may not be intuitive. In these cases using rules to 'correct' the output could be a very useful process. There still remains the possibility of using rules to improve named entity extraction even when using word lattices. Such rules would not be trivial and are not covered in this thesis.

### Discriminative training based on unlabelled data

We have shown that it is possible to improve the language models, and consequently the named entity recognition, by discriminative training. We have seen a small improvement when using the training data, and a larger improvement when using a held-out set.

An experiment for future consideration would be to see if techniques which have been developed for learning from unlabelled data could be applied to improve the language models. An example of a method that could be attempted relies on the knowledge of optimum weights for our system. The first stage would be to use the current system weighted optimally to mark up some unlabelled data. Then, by incorrectly weighting the system in a rerun over the same data, a separate set of marked up data would be generated. The first set of labelled data would be more reliable than the second set of labelled data. By comparison of the two data sets it would be possible to detect which probabilities, when adjusted, would give improvement in the incorrectly weighted system. By improving the incorrectly weighted system, we would effectively be increasing the stability of the system; that is, making it more robust against changes in weights. In other words, a model with good estimates of language model probabilities would be robust to small perturbations in the weights (Ostendorf 1998). If there is a direct relationship between stability and correctness, which would need to be proved emperically, it might result in the system with the new language models being more effective when correctly weighted (as well as when incorrectly weighted).

**What is the difference between an aardvark and a zywiec?**

One question that may be worth considering is whether there is more than one type of unknown word. To the reader the word 'aaron' almost certainly refers to a person, whereas the word 'zygote' clearly doesn't refer to any entity that we are now investigating. It would be interesting to investigate whether there are any clues beyond the immediate context to tell us that some unknown words are People named entities but that other unknowns are not. In particular, it might be advantageous to consider the frequency of the individual unknown word within the immediate context. The reason being that if, within a certain context, the unknown word 'floccinaucinihilipilification' appears, it is very unlikely to recur within close proximity; whereas, if an unknown name such as 'horlock' appears, it may very likely reappear shortly afterwards. It would be interesting to investigate whether this phenomenon is true or not, and, if it is true, can the information be utilised to aid named entity extraction from speech.

### 8.2.3 Application of the model

There is potential to use the model defined in this thesis as part of a larger system. Named entity extraction from word lattices may well prove to be a fundamental component in information extraction from speech/information retrieval from speech. In this thesis it has been treated purely as a task in its own right and consequently can only be compared with other methods of named entity extraction from speech or, more particularly, named entity extraction from speech transcripts.

In the future, the application of the model could be an essential part of research. It would be necessary to investigate how named entity extraction from word lattices fit within the larger tasks of information extraction from speech and information retrieval from speech.

#### Integration with speech recognition

It would be useful to integrate the named entity component into the speech recognition end of the process. Although for the purpose of this thesis, the use of Cambridge University word lattices was sufficient, in practical applications it would be necessary to generate new word lattices, which requires a speech recognizer. Three experiments for integration with a speech recognizer are suggested:

- An experiment to determine how broad the lattices need to be. The broader the lattice, the lower the lattice error rate, and potentially the higher the F-score of the named entity component. However the broader the lattice, the slower the process. There is clearly a lattice size such that a broader lattice cannot improve the named entity recognition. An experiment to determine the optimum would be required.

- An experiment to see the effects of a known vocabulary for the speech recognizer on the named entity extraction task. In our experiments the vocabulary of the speech recognizer used to generate the lattices was unknown, as discussed in section 5.6. It would be interesting to investigate whether knowledge of the vocabulary would enable a more closely related vocabulary for the named entity component - for example, the most common 10,000 words from the recognizer's vocabulary.

- An equally interesting, and possibly more productive experiment, would be to change the vocabulary of the speech recognizer to be consistent with the vocabulary of the named entity recognizer. This might be more productive, because the vocabulary of the named entity recognizer is far more constrained than that of the speech recognizer and therefore words in the vocabulary of the named entity recognizer could be predicted accurately in the speech recognizer. The reverse, however, is not likely to be true.

#### Integration with information extraction/information retrieval

At the other end of the spectrum, it is important to integrate the named entity component into an information extraction/information retrieval process.

Named entity extraction is a small subsection of information extraction. Many information extraction systems (eg (Karkaletsis et al. 2003) and (Sekine & Nobata 2003)) use named entity extraction as a preliminary stage of the process. Similarly, information retrieval techniques often use named entities, since important sections of documents tend to be those containing named entities. Named entities are therefore often appropriate for queries to information retrieval systems.

Integration with either system would show the experimental value of improved named entity recognition by the methods outlined in this thesis.

# Appendix A

# The extra confusion caused by lattices

In the following example all probabilities are fictional and for simplicity log probabilities are used rather than real probabilities.

Assume the following subset of a speech lattice.

```
...
J=927 S=365 E=845 W=RECORDS a=-3113.81 l=-10.605
J=937 S=365 E=855 W=RECORD a=-3345.31 l=-10.723
J=2489 S=845 E=1245 W=OF a=-466.25 l=-2.988
J=4915 S=855 E=2083 W=FOR a=-793.03 l=-4.642
J=6028 S=1245 E=2309 W=THAT a=-647.29 l=-5.563
J=7541 S=2083 E=2834 W=THE a=-825.86 l=-1.406
J=7554 S=2309 E=2834 W=THE a=-710.44 l=-4.795
...
```

To find the best path through this lattice (assuming equal unit weighting of the acoustic and language model probabilities) we simply calculate all paths through this lattice and find the most likely.

... RECORDS OF THAT THE ... log probability = -4961.74

... RECORD FOR THE ... log probability = -4980.97

Thus in this instance the selection would be:

... RECORDS OF THAT THE ...

To add the named entity recognition component we require the following probabilities:

| | |
|---|---|
| P(<OTHER> \|...) = -2 | P(<PERSON> \|...) = -4 |
| P(<ORGANIZATION> \|...) = -5 | P(<MONEY> \|...) = -3.55 |
| P(<PERCENT> \|...) = -8 | P(<LOCATION\|...) = -3.5 |
| P(<TIME> \|...) = -6.58 | P(<DATE> \|...) = -9 |

to find the probability of the entity of the first word ($P(e_1|..)$ and $P(e_2|..)$);

| | |
|---|---|
| P(RECORDS\| <OTHER>,...) = -1.8 | P(RECORDS\| <PERSON>,...) = -4 |
| P(RECORDS\| <ORGANIZATION>,...) = -5 | P(RECORDS\| <MONEY>,...) = -3.41 |
| P(RECORDS\| <PERCENT>,...) = -12 | P(RECORDS\| <LOCATION,...) = -2.5 |
| P(RECORDS\| <TIME>,...) = -7.95 | P(RECORDS\| <DATE>,...) = -16 |

to find the probability of 'RECORDS' given an entity ($P(w_1|..)$);

| | |
|---|---|
| P(RECORD\| <OTHER>,...) = -2 | P(RECORD\| <PERSON>,...) = -3.2 |
| P(RECORD\| <ORGANIZATION>,...) = -5.1 | P(RECORD\| <MONEY>,...) = -3.55 |
| P(RECORD\| <PERCENT>,...) = -13 | P(RECORD\| <LOCATION,...) = -3.01 |
| P(RECORD\| <TIME>,...) = -6.58 | P(RECORD\| <DATE>,...) = -14.7 |

to find the probability of 'RECORD' given an entity ($P(w_2|..)$);

P(<OTHER> |RECORDS,<OTHER>,...) = -1.8

P(<OTHER> |RECORDS,<PERSON>,...) = -4

P(<OTHER> |RECORDS,<ORGANIZATION>,...) = -5

P(<OTHER> |RECORDS,<MONEY>,...) = -3.41

P(<OTHER> |RECORDS,<PERCENT>,...) = -12

P(<OTHER> |RECORDS,<LOCATION,...) = -2.5

P(<OTHER> |RECORDS,<TIME>,...) = -7.95

P(<OTHER> |RECORDS,<DATE>,...) = -16

P(<PERSON> |RECORDS,<OTHER>,...) = -1.8

P(<PERSON> |RECORDS,<PERSON>,...) = -4

P(<PERSON> |RECORDS,<ORGANIZATION>,...) = -5

P(<PERSON> |RECORDS,<MONEY>,...) = -3.41

P(<PERSON> |RECORDS,<PERCENT>,...) = -12

P(<PERSON> |RECORDS,<LOCATION,...) = -2.5

P(<PERSON> |RECORDS,<TIME>,...) = -7.95

P(<PERSON> |RECORDS,<DATE>,...) = -16

P(<ORGANIZATION> |RECORDS,<OTHER>,...) = -1.3

P(<ORGANIZATION> |RECORDS,<PERSON>,...) = -4.3

P(<ORGANIZATION> |RECORDS,<ORGANIZATION>,...) = -4.99

P(<ORGANIZATION> |RECORDS,<MONEY>,...) = -6.41

P(<ORGANIZATION> |RECORDS,<PERCENT>,...) = -12.75

P(<ORGANIZATION> |RECORDS,<LOCATION,...) = -2

P(<ORGANIZATION> |RECORDS,<TIME>,...) = -7.67

P(<ORGANIZATION> |RECORDS,<DATE>,...) = -14.3

etc

to find the probability of the entity after 'RECORDS' ($P(e_3|..)$);

P(<OTHER> |RECORD,<OTHER>,...) = -2

P(<OTHER> |RECORD,<PERSON>,...) = -3.2

P(<OTHER> |RECORD,<ORGANIZATION>,...) = -5.1

P(<OTHER> |RECORD,<MONEY>,...) = -3.35

P(<OTHER> |RECORD,<PERCENT>,...) = -11.72

P(<OTHER> |RECORD,<LOCATION,...) = -3.01

P(<OTHER> |RECORD,<TIME>,...) = -6.58

P(<OTHER> |RECORD,<DATE>,...) = -18.71

P(<PERSON> |RECORD,<OTHER>,...) = -2

P(<PERSON> |RECORD,<PERSON>,...) = -3.7

P(<PERSON> |RECORD,<ORGANIZATION>,...) = -5.1

P(<PERSON> |RECORD,<MONEY>,...) = -3.53

P(<PERSON> |RECORD,<PERCENT>,...) = -13

P(<PERSON> |RECORD,<LOCATION,...) = -3.01

P(<PERSON> |RECORD,<TIME>,...) = -4.58

P(<PERSON> |RECORD,<DATE>,...) = -4.7

P(<ORGANIZATION> |RECORD,<OTHER>,...) = -2

P(<ORGANIZATION> |RECORD,<PERSON>,...) = -4.12

P(<ORGANIZATION> |RECORD,<ORGANIZATION>,...) = -5.1

P(<ORGANIZATION> |RECORD,<MONEY>,...) = -3.75

P(<ORGANIZATION> |RECORD,<PERCENT>,...) = -13

P(<ORGANIZATION> |RECORD,<LOCATION,...) = -3.01

P(<ORGANIZATION> |RECORD,<TIME>,...) = -6.59

P(<ORGANIZATION> |RECORD,<DATE>,...) = -14.7

etc

to find the probability of the entity after 'RECORD' ($P(e_4|..)$).

and so on.

These probabilities are not immediately available to us in this format, however, they are calculable by the method described in section 5.4 of the thesis.

We thus build up all of our possible transcriptions.

... RECORDS log probability = -3730.81 (<OTHER>)

... RECORDS log probability = -3732.36 (<MONEY>)

etc

Finally when we have a completed list, from which we select the best named entity marked up path through the lattice.

# Appendix B

# Dynamic Alignment of Lattices with Text

In this appendix we detail a step by step explanation of dynamic alignment of lattices with text as per chapter 3. We consider the lattice in table B.1 and the text in table B.2.

The first stage is to create the m x n grid (where m = L-1 from the lattice and n is the length of the sentence. Along the x-axis we label the words from the text B.2 and along the y-axis the words from the lattice B.1 - the lattice words are associated with the node at which they end (in our example both "RECOGNISE" and "NICE" end at the same node). The bottom left hand corner is marked as containing zero errors. We thus have the grid in figure B.1.

Working left to right, and bottom to top we take the token in each grid space and propogate it to all possible locations (generally three). If the new location is either empty or has a worse count of errors the new token is held in the new location, otherwise it is discarded.

The first token that we propogate then is the only one available. There are three possible ways that we can move the first token as per figure B.2.

- We can move it vertically, corresponding to reading one word in the lattice but no words from the text; ie an insertion error.

| | | | | | |
|---|---|---|---|---|---|
| N=10 | L=11 | | | | |
| J=0 | S=0 | E=1 | W=!SENT_START | a=-129.05 | l=0.00 |
| J=1 | S=1 | E=2 | W=TO | a=-3036.62 | l=-5.622 |
| J=2 | S=2 | E=3 | W=WRECK | a=-3046.72 | l=-9.542 |
| J=3 | S=2 | E=5 | W=RECOGNISE | a=-4032.03 | l=-15.219 |
| J=4 | S=3 | E=4 | W=A | a=-636.12 | l=-6.408 |
| J=5 | S=4 | E=5 | W=NICE | a=-1298.77 | l=-4.815 |
| J=6 | S=5 | E=6 | W=SPEECH | a=-1083.14 | l=-5.312 |
| J=7 | S=5 | E=7 | W=BEACH | a=-1071.32 | l=-5.567 |
| J=8 | S=6 | E=8 | W=!SENT_END | a=-1424.97 | l=-1.092 |
| J=9 | S=7 | E=8 | W=!SENT_END | a=-1424.97 | l=-1.854 |
| J=10 | S=8 | E=9 | W=!NULL | a=0.0 | l=0.0 |

Table B.1: *An example speech word lattice.*

| |
|---|
| !SENT_START TO WRECK THE NICE BEACH !SENT_END |

Table B.2: *An example text for comparison with the lattice.*

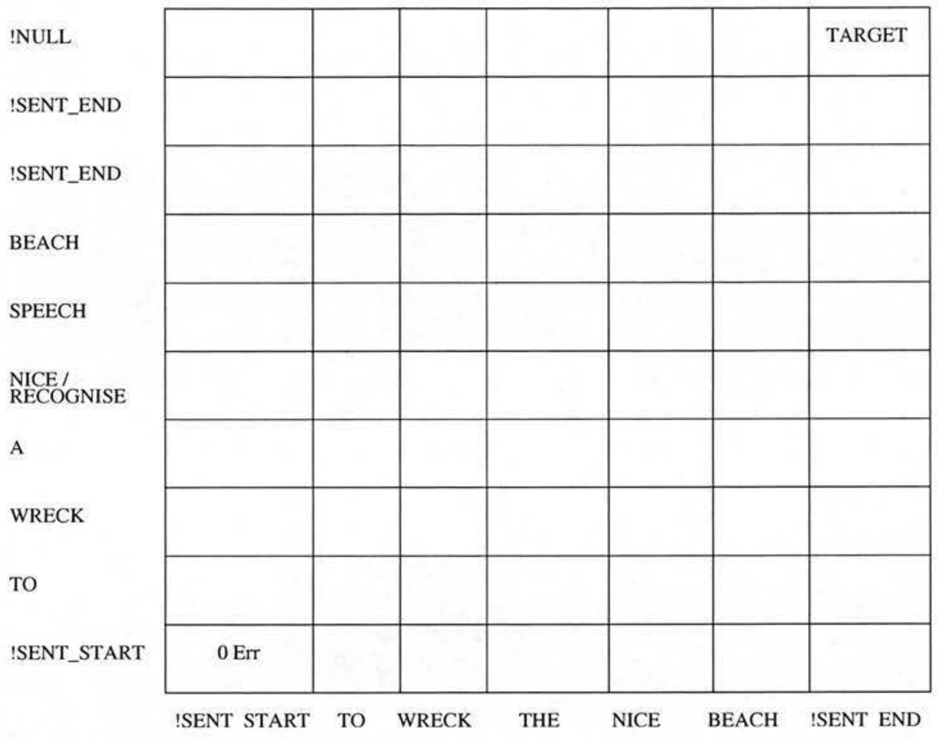


Figure B.1: *Blank grid before any propogation.*

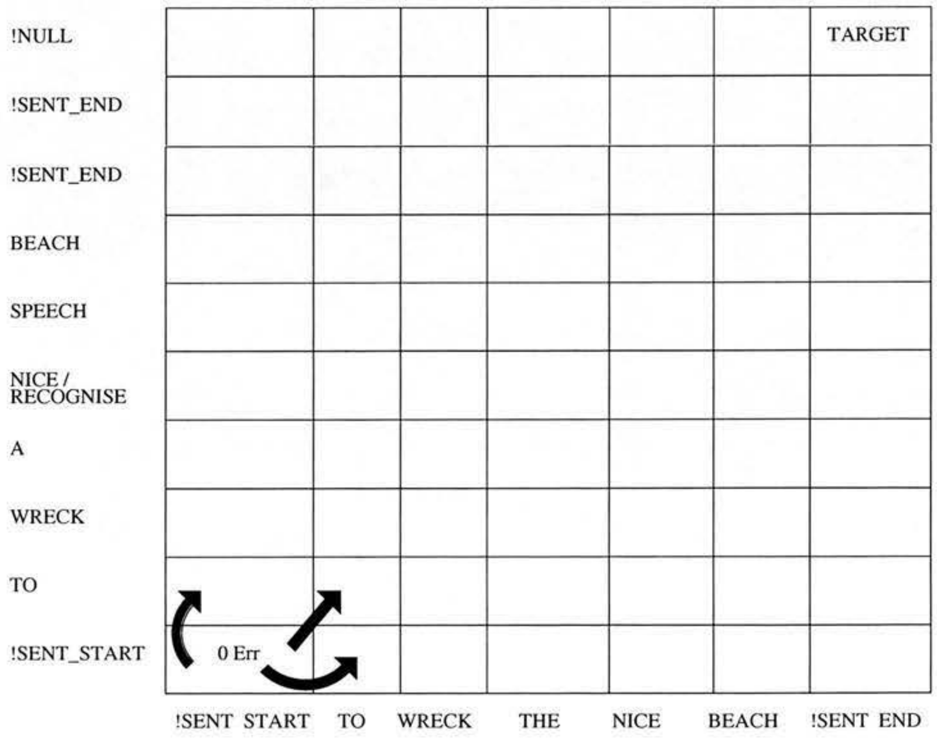


Figure B.2: *Blank grid showing propogation from the first token.*

- We can move it horizontally, corresponding to reading one word in the text but no words in the lattice; ie a deletion error.
- We can move it diagonally. In this instance, because the word "TO" in the lattice matches the word "TO" in the text, no errors occur. If "TO" had not occurred in both lattice and text a substitution error would have occurred here.

We now move to the right and propogate the token there.

- We can move it vertically, corresponding to reading one word in the lattice but no words from the text; ie an insertion error. In this instance there is already a token in place. The current token has zero errors, the new token has 1 deletion error plus the new insertion error. Therefore the new token is destroyed and the old token kept. This token now has 0 errors.
- We can move it horizontally, corresponding to reading one word in the text but no words in the lattice; ie a deletion error. This token now has 2 deletion errors.
- We can move it diagonally. In this instance as the word "TO" in the lattice matches the word "WRECK" in the text, 1 substitution error occurs. This token now has 1 deletion and 1 substitution error.

This process continues until the final square of the bottom row has been propogated. There is no need to propogate further to the right as we have already reached the worst possible senario - no words have matched.

We now move up one level and again commence on the left hand side of the grid. This time there are five possible choices as per figure B.3, which shows the current state of all tokens. The additional possibilities are caused by the division of the lattice at node 2.

- We can move it vertically with the word "WRECK", corresponding to reading one word in the lattice but no words from the textt; ie an insertion error. In this instance there are no tokens in place. The new token has 1 insertion error, plus the new insertion error leaving 2 insertion errors.

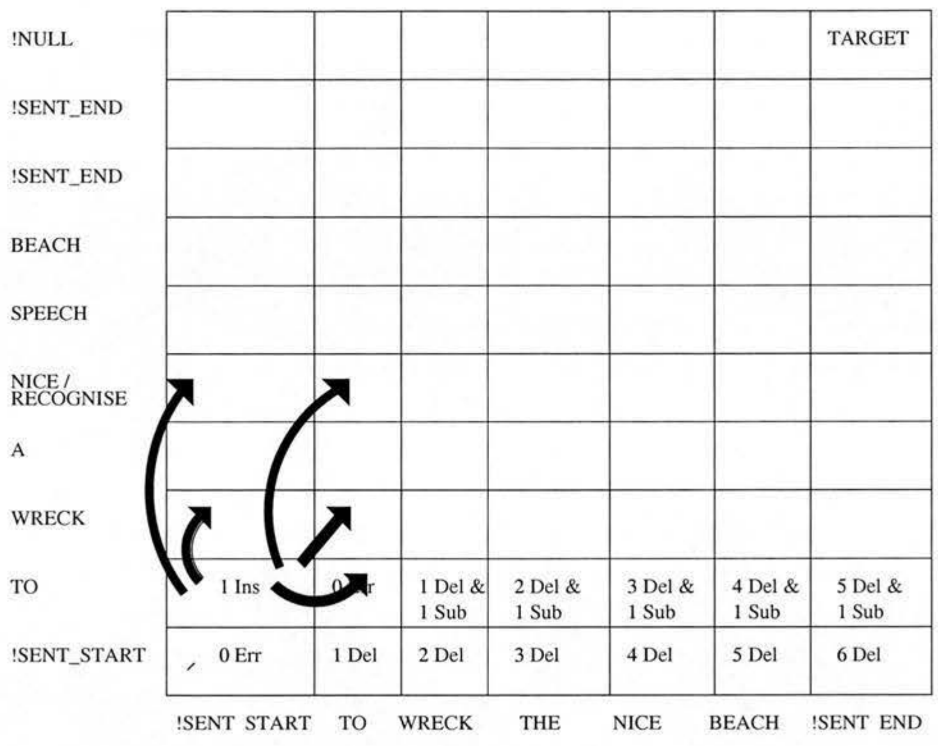


Figure B.3: *Grid after propogation of the first row.*

- We can move it vertically with the word "RECOGNISE", corresponding to reading one word in the lattice but no words from the text; ie an insertion error. In this instance there are no tokens in place. The new token has 1 insertion error, plus the new insertion error leaving two insertion errors.

- We can move it horizontally, corresponding to reading one word in the text but no words in the lattice; ie a deletion error. In this instance there is already a token with 0 errors, thus we keep the zero error token.

- We can move it diagonally with the word "WRECK". In this instance because the word "TO" in the text matches the word "WRECK" in the lattice, 1 substitution error occurs. This token now has 1 insertion and 1 substitution error.

- We can move it diagonally with the word "RECOGNISE". In this instance as the word "TO" in the text matches the word "RECOGNISE" in the lattice, 1 substitution error occurs. This token now has 1 insertion and 1 substitution error.

Progressing to the right again there are 5 possibilities.

- We can move it vertically with the word "WRECK", corresponding to reading one word in the lattice but no words from the text; ie an insertion error. In this instance there is a token containing 1 insertion and 1 substitution. The new token has only the new insertion error, thus the old token is replaced with this new token.

- We can move it vertically with the word "RECOGNISE", corresponding to reading one word in the lattice but no words from the text; ie an insertion error. In this instance there is a token containing 1 insertion and 1 substitution. The new token has only the new insertion error, thus the old token is replaced with this new token.

- We can move it horizontally, corresponding to reading one word in the text but no words in the lattice; ie a deletion error. In this instance there is already a token with 1 deletion and 1 substitution error, thus we keep the new token with only 1 deletion error.

- We can move it diagonally with the word "WRECK". In this instance because the word "WRECK" in the text matches the word "WRECK" in the lattice, no errors occur. This token now has zero errors.

- We can move it diagonally with the word "RECOGNISE". In this instance as the word "WRECK" in the text matches the word "RECOGNISE" in the lattice, 1 substitution error occurs. This token now has 1 substitution error.

Progressing one step to the right until the row is completed there are five possibilities each time, until the grid looks like figure B.4.

The process is repeated row by row in figure B.5 through B.10. It is important that all rows are completely evaluated and the process is not stopped as soon as the target is reached. In this instance the target is reached by the row below the row with least errors.

It is then possible to trace back from the target the best path, since each token has

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | | | | | | | |
| !SENT_END | | | | | | | |
| BEACH | | | | | | | |
| SPEECH | | | | | | | |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub | 4 Del & 1 Sub |
| A | | | | | | | |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del & 2 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub | 4 Del & 1 Sub |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.4: *Grid after propogation of the second row.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | | | | | | | |
| !SENT_END | | | | | | | |
| BEACH | | | | | | | |
| SPEECH | | | | | | | |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub | 4 Del & 1 Sub |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.5: *Grid after propogation of the third row.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | | | | | | | |
| !SENT_END | | | | | | | |
| BEACH | | | | | | | |
| SPEECH | | | | | | | |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 1 Sub | 1 Del & 2 Sub | 3 Del & 2 Sub |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.6: *Grid after propogation of the fourth row.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | | | | | | | |
| !SENT_END | | | | | | | |
| BEACH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 1 Sub | 2 Sub & 1 Del |
| SPEECH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 2 Sub | 2 Sub & 1 Del |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 1 Sub | 1 Sub & 1 Del | 1 Sub & 2 Del |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.7: *Grid after propogation of the fifth row.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | | | | | | | |
| !SENT_END | | | | | | | |
| BEACH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 1 Sub | 2 Sub & 1 Del |
| SPEECH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 2 Sub | 2 Sub & 1 Del |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 1 Sub | 1 Sub & 1 Del | 1 Sub & 2 Del |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.8: *Grid after propogation of the sixth row.*

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | | | | | | | |
| !SENT_END | 4 Ins | 3 Ins | 2 Ins & 1 Sub | 1 Ins & 2 Sub | 1 Sub & 2 Ins | 2 Sub & 1 Ins | 3 Sub |
| BEACH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 1 Sub | 2 Sub & 1 Del |
| SPEECH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 2 Sub | 2 Sub & 1 Del |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 1 Sub | 1 Sub & 1 Del | 1 Sub & 2 Del |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.9: *Grid after propogation of the seventh row.*

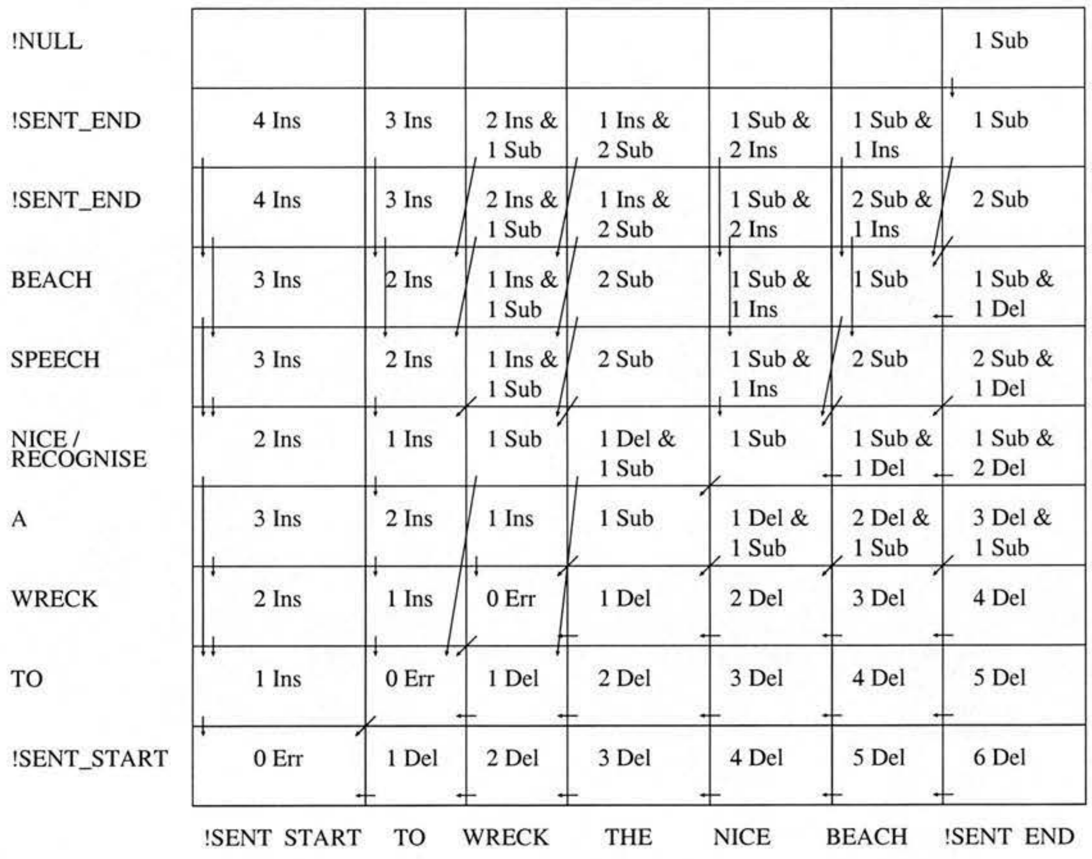
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | TARGET |
| !SENT_END | 4 Ins | 3 Ins | 2 Ins & 1 Sub | 1 Ins & 2 Sub | 1 Sub & 2 Ins | 1 Sub & 1 Ins | 1 Sub |
| !SENT_END | 4 Ins | 3 Ins | 2 Ins & 1 Sub | 1 Ins & 2 Sub | 1 Sub & 2 Ins | 2 Sub & 1 Ins | 2 Sub |
| BEACH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 1 Sub | 2 Sub & 1 Del |
| SPEECH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 2 Sub | 2 Sub & 1 Del |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 1 Sub | 1 Sub & 1 Del | 1 Sub & 2 Del |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.10: *Grid after propogation of the eighth row.*

kept a record of where it was propogated from, as illustrated in figure B.11. The correct solution to our example in reverse order reads:

Target, !SENT_END, BEACH, NICE, A, WRECK, TO, !SENT_START

This sentence is chosen as it contains only 1 substitution error.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| !NULL | | | | | | | 1 Sub |
| !SENT_END | 4 Ins | 3 Ins | 2 Ins & 1 Sub | 1 Ins & 2 Sub | 1 Sub & 2 Ins | 1 Sub & 1 Ins | 1 Sub |
| !SENT_END | 4 Ins | 3 Ins | 2 Ins & 1 Sub | 1 Ins & 2 Sub | 1 Sub & 2 Ins | 2 Sub & 1 Ins | 2 Sub |
| BEACH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 1 Sub | 1 Sub & 1 Del |
| SPEECH | 3 Ins | 2 Ins | 1 Ins & 1 Sub | 2 Sub | 1 Sub & 1 Ins | 2 Sub | 2 Sub & 1 Del |
| NICE / RECOGNISE | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 1 Sub | 1 Sub & 1 Del | 1 Sub & 2 Del |
| A | 3 Ins | 2 Ins | 1 Ins | 1 Sub | 1 Del & 1 Sub | 2 Del & 1 Sub | 3 Del & 1 Sub |
| WRECK | 2 Ins | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del |
| TO | 1 Ins | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del |
| !SENT_START | 0 Err | 1 Del | 2 Del | 3 Del | 4 Del | 5 Del | 6 Del |
| | !SENT START | TO | WRECK | THE | NICE | BEACH | !SENT END |

Figure B.11: *Completed grid showing the origin of tokens - effectively showing best path.*

# Appendix C

# Universal Transcription Format

## C.1 Fragment of Universal Transcription Format training data

```
<utf dtd_version="utf-1.0" audio_filename="a960521.sph" language="english" versi
on="5" version_date="980817" scribe="obert_markoff">
<bn_episode_trans program="ABC_Nightline" air_date="">
<!-- History:  Version 1:  initial release; Version 2: Reformatted to be in spe
c.  -->
<!-- This DTT file was automatically generated by bn_filt.pl Version: 1.12 on 'M
on May 18 08:13:57 EDT 1998' -->
<section type="filler" startTime="0.000" endTime="61.320" id="a960521.1">
<background startTime="0.000" type="Music" level="High">
<background startTime="1.765" type="Music" level="Low">
<turn speaker="Ted_Koppel" spkrtype="male" dialect="native" startTime="1.765" en
dTime="5.074" mode="planned" fidelity="high">
 It's a question that will make a lot of Americans think
</turn>
<turn speaker="a960521_M_US_003" spkrtype="male" dialect="native" startTime="6.8
59" endTime="7.363" mode="spontaneous" fidelity="medium">
 Damn
</turn>
<turn speaker="Ted_Koppel" spkrtype="male" dialect="native" startTime="7.910" en
dTime="13.120" mode="planned" fidelity="medium">
 You think you're white
<time sec="9.711">
 you're not
```

```
<time sec="11.743">
 you're black
</turn>
<turn speaker="Ted_Koppel" spkrtype="male" dialect="native" startTime="15.200" e
ndTime="18.634" mode="planned" fidelity="high">
 It's a question that will make a lot of Americans angry
</turn>
<turn speaker="Ted_Koppel" spkrtype="male" dialect="native" startTime="20.336" e
ndTime="31.267" mode="planned" fidelity="medium">
 In order for you to be black
<time sec="22.599">
 for the rest of your life
<time sec="24.817">
 what would it take to compensate you for that
<time sec="29.179">
 How much do you want
</turn>
<turn speaker="a960521_M_US_003" spkrtype="male" dialect="native" startTime="31.
645" endTime="32.402" mode="spontaneous" fidelity="medium">
 How much do I want
</turn>
<turn speaker="Ted_Koppel" spkrtype="male" dialect="native" startTime="32.402" e
ndTime="36.110" mode="planned" fidelity="medium">
 How much would it take {breath
</turn>
<turn speaker="Ted_Koppel" spkrtype="male" dialect="native" startTime="36.110" e
ndTime="44.019" mode="planned" fidelity="high">
 We continue our series
<time sec="38.116">
 America in black and white
<time sec="40.358">
 Tonight how much is white skin worth
</turn>
```

## C.2 UTF Document Type Definition (DTD)

```
<!SGML "ISO 8879:1986"
--
```

```
File:   @(#)utf.dtd     v2      Aug 17, 1998


Authors: Paul Morgovsky and Milan Young
         Linguistic Data Consortium,
         University of Pennsylvania.

         Henry S. Thompson,
         Language Technology Group
         University of Edinburgh

         Jon Fiscus
         Spoken Natural Language Processing Group
         NIST

Desc: SGML and DTD declaration for the new specifications for the
      Transcription of Spoken Language.

      Numerous changes were made to enable named entity tagging
      and ASR tagging to co-exist.  This dtd is also annotated
      with comments, which when ran through the appropriate PERL
      script, will result is a DTD without active shortrefs.

Revision History:
      - nothing yet

Usage:
        nsgmls utf.dtd filename
--

CHARSET  BASESET  "ISO 646-1983//CHARSET
                   International Reference Version (IRV)//ESC 2/5 4/0"
         DESCSET  0  9 UNUSED   -- NUL,SOH,STX,ETX,ETO,ENQ,ACK,BEL,BS --
                  9  2  9
                  11  2 UNUSED  -- VT,FF --
                  13  1 13
                  14 18 UNUSED  -- SO,SI,DLE,DC1,DC2 --
                  32 95 32
```

```
                 127  1 UNUSED -- del character --
          BASESET  "ISO Registration Number 109//CHARSET
                    ECMA-94 Right Part of Latin-1 Alphabet Nr.3//ESC 2/9 4/3"
          DESCSET  128 32 UNUSED -- no such characters --
                   160 1 UNUSED  -- nbs character --
                   161 95 161    -- 161 through 255 inclusive --

CAPACITY PUBLIC    "ISO 8879:1986//CAPACITY Reference//EN"
SCOPE    DOCUMENT

SYNTAX   SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
                           18 19 20 21 22 23 24 25 26 27 28 29 30 31 127 160
         BASESET  "ISO 646-1983//CHARSET International Reference
                   Version (IRV)//ESC 2/5 4/0"
         DESCSET  0 128 0
         FUNCTION RE                  13
                  RS                  10
                  SPACE               32
                  TAB      SEPCHAR    9
         NAMING   LCNMSTRT ""
                  UCNMSTRT ""
                  LCNMCHAR "_-."
                  UCNMCHAR "_-."
                  NAMECASE GENERAL    YES
                           ENTITY     NO
         DELIM    GENERAL  SGMLREF
                  SHORTREF NONE "*" "+" "^" "%" "@" "," "." "?" "{" "_" "["
                               "&#RE;"
                               "&#RE;&#RS;"
                               "&#RE;&#RS;B"
                               "&#RE;B"
                               "&#RE;B&#RS;"
                               "&#RS;"
                               "&#RS;&#RE;"
                               "&#RS;&#RE;B"
                               "&#RS;B"
                               "&#RS;B&#RE;"
                               "B"
```

```
                          "B&#RE;"
                          "B&#RE;&#RS;"
                          "B&#RS;"
                          "B&#RS;&#RE;"

         NAMES    SGMLREF
         QUANTITY SGMLREF
                  NAMELEN   99999999
                  PILEN     24000
                  TAGLEN    99999999
                  TAGLVL    99999999

FEATURES MINIMIZE DATATAG   NO
                  OMITTAG   YES
                  RANK      YES
                  SHORTTAG  YES
         LINK     SIMPLE    YES 1000
                  IMPLICIT  YES
                  EXPLICIT  YES 1
         OTHER    CONCUR    NO
                  SUBDOC    YES 99999999
                  FORMAL    YES
APPINFO  NONE>

<!-- This dtd has bee augmented with comments, which, after applying the followin
     filter will dissable shortrefs in the DTD.  Thus, text tokens are not parsed
     sgml but is instead left to the application.

perl -pe  'if (/DELETE TO DISABLE SHORTREF/) { ($_ = "\n") } elsif (/BEGIN COMMEN

  -->

<!DOCTYPE utf [

<!-- Quick Substitution Entities -->

<!-- BEGIN COMMENT TO DISABLE SHORTREF -->
<!ENTITY % textTokens             "(separator | pName | mispronounced | misspelling
<!-- END COMMENT TO DISABLE SHORTREF -->

<!-- DELETE TO DISABLE SHORTREF
<!ENTITY % textTokens           "(#PCDATA | contraction | fragment | hyphen | wti
    DELETE TO DISABLE SHORTREF -->

<!ENTITY % ne_bound         "( b_enamex | e_enamex | b_timex | e_timex | b_numex
<!ENTITY % asr_bound        "( b_foreign | e_foreign | b_unclear | e_unclear | b_



<!ENTITY NONSPEECH              "<nonSpeech>" >
<!ENTITY ACOUSTICNOISE          "<acousticnoise>" >
<!ENTITY SEP                    "<separator>" >
<!ENTITY PNAME                  "<pName>" >
<!ENTITY MISPRONOUNCED          "<mispronounced>" >
<!ENTITY MISSPELLING            "<misspelling>" >
<!ENTITY ACRONYM                "<acronym>" >
<!ENTITY IDIOSYNCRATIC          "<idiosyncratic>" >
<!ENTITY NONLEXEME              "<nonlexeme>" >
<!ENTITY PERIOD                 "<period>" >
<!ENTITY QMARK                  "<qmark>" >
<!ENTITY COMMA                  "<comma>" >
<!ENTITY IGNORE                 "">

<!-- Document Grammar Specifications -->
<!-- Structural definition -->
<!ELEMENT utf           - -     ( bn_episode_trans | conversation_trans ) >
<!ELEMENT bn_episode_trans
                        - -     (section | recording_change | background)+ >
<!ELEMENT section       - -     (turn | background)* >

<!ELEMENT conversation_trans            - -     (turn | background)* >

<!ELEMENT recording_change - O  EMPTY >
<!ELEMENT turn          - -     ( %textTokens; | time | background | %ne_bound; |
<!ELEMENT separator     - O     EMPTY >
```

```
<!-- Floating elements -->
<!ELEMENT background    - O      EMPTY >
<!ELEMENT time            - O    EMPTY >
<!ELEMENT wtime           - O    EMPTY >

<!-- Bouunding tags made explicitly -->
<!ELEMENT b_foreign       - O    EMPTY >
<!ELEMENT b_unclear       - O    EMPTY >
<!ELEMENT b_overlap       - O    EMPTY >
<!ELEMENT b_noscore       - O    EMPTY >
<!ELEMENT b_aside         - O    EMPTY >
<!ELEMENT e_foreign       - O    EMPTY >
<!ELEMENT e_unclear       - O    EMPTY >
<!ELEMENT e_overlap       - O    EMPTY >
<!ELEMENT e_noscore       - O    EMPTY >
<!ELEMENT e_aside         - O    EMPTY >

<!ELEMENT b_enamex        - O    EMPTY >
<!ELEMENT b_timex         - O    EMPTY >
<!ELEMENT b_numex         - O    EMPTY >
<!ELEMENT e_enamex        - O    EMPTY >
<!ELEMENT e_timex         - O    EMPTY >
<!ELEMENT e_numex         - O    EMPTY >

<!-- Applied word tags -->
<!ELEMENT fragment      - O      EMPTY >
<!ELEMENT contraction   - O      EMPTY >

<!-- Shortref elements -->
<!ELEMENT pName         - O      EMPTY >
<!ELEMENT mispronounced - O      EMPTY >
<!ELEMENT misspelling   - O      EMPTY >
<!ELEMENT acronym       - O      EMPTY >
<!ELEMENT idiosyncratic - O      EMPTY >
<!ELEMENT nonlexeme     - O      EMPTY >
<!ELEMENT nonSpeech     - O      EMPTY >
<!ELEMENT acousticnoise - O      EMPTY >
<!ELEMENT period        - O      EMPTY >
```

```
<!ELEMENT qmark        - O     EMPTY >
<!ELEMENT comma        - O     EMPTY >
<!ELEMENT hyphen       - O     EMPTY >

<!-- Attributes of the Tags -->
<!ATTLIST utf     dtd_version      (utf-1.0) #REQUIRED
                  audio_filename   CDATA #REQUIRED
                  language         CDATA #REQUIRED
                  scribe           CDATA #IMPLIED
                  version          NUMBER #IMPLIED
                  version_date     CDATA #IMPLIED>

<!ATTLIST bn_episode_trans
                      program      CDATA #REQUIRED
                      air_date     CDATA #IMPLIED >

<!ATTLIST conversation_trans
                      recording_date       CDATA #IMPLIED >

<!ATTLIST section     type        (report|filler|nontrans) #REQUIRED
                      startTime   CDATA #REQUIRED
                      endTime     CDATA #REQUIRED
                      id          CDATA #IMPLIED
                      topic       CDATA #IMPLIED >

<!ATTLIST recording_change show CDATA #REQUIRED
                      date        CDATA #REQUIRED
                      sec         CDATA #REQUIRED >

<!ATTLIST turn        speaker     CDATA #REQUIRED
                      spkrtype (male|female|child|unknown) #REQUIRED
                      dialect     CDATA #IMPLIED
                      startTime   CDATA #REQUIRED
                      endTime     CDATA #REQUIRED
                      mode        (planned|spontaneous) #IMPLIED
                      channel     CDATA #IMPLIED
                      fidelity    (low|medium|high) #IMPLIED >
```

```
<!ATTLIST b_noscore            startTime         CDATA #REQUIRED
                             endTime      CDATA #REQUIRED
                             reason       CDATA CDATA >

<!ATTLIST b_foreign    language   CDATA #REQUIRED >

<!ATTLIST contraction e_form     CDATA #REQUIRED >

<!ATTLIST b_overlap    startTime        CDATA #IMPLIED
                     endTime    CDATA #IMPLIED >

<!ATTLIST time         sec        CDATA #REQUIRED >

<!ATTLIST wtime        startTime  CDATA #REQUIRED
                     endTime    CDATA #REQUIRED
                     clust      CDATA #IMPLIED
                     conf       CDATA #IMPLIED >

<!ATTLIST background         startTime   CDATA #REQUIRED
                             type        (music|speech|other) #REQUIRED
                             level       (off|low|high) #REQUIRED >

<!ATTLIST b_enamex   type       CDATA  #REQUIRED
                     status     (opt)  #IMPLIED
                     alt        CDATA  #IMPLIED >

<!ATTLIST b_timex    type       CDATA  #REQUIRED
                     status     (opt)  #IMPLIED
                     alt        CDATA  #IMPLIED >

<!ATTLIST b_numex    type       CDATA  #REQUIRED
                     status     (opt)  #IMPLIED
                     alt        CDATA  #IMPLIED >


<!-- Short Refference Mappings -->

<!-- BEGIN COMMENT TO DISABLE SHORTREF -->
```

```
<!SHORTREF TURN         '.'     PERIOD
                        '?'     QMARK
                        ','     COMMA
                        '+'     MISPRONOUNCED
                        '@'     MISSPELLING
                        '_'     ACRONYM
                        '^'     PNAME
                        '*'     IDIOSYNCRATIC
                        '%'     NONLEXEME
                        '{'     NONSPEECH
                        '['     ACOUSTICNOISE
                        '&#RS;B&#RE;'   IGNORE
                        '&#RS;&#RE;'    IGNORE
                        '&#RE;&#RS;'    SEP
                        '&#RE;&#RS;B'   SEP
                        '&#RE;'         SEP
                        '&#RE;B&#RS;'   SEP
                        '&#RE;B'        SEP
                        '&#RS;&#RE;B'   SEP
                        '&#RS;'         SEP
                        '&#RS;B'        SEP
                        'B&#RE;&#RS;'   SEP
                        'B&#RE;'        SEP
                        'B&#RS;&#RE;'   SEP
                        'B&#RS;'        SEP
                        'B'             SEP   >

<!USEMAP TURN turn >

<!-- END COMMENT TO DISABLE SHORTREF -->

]>
```

# Bibliography

Appelt, D. & Martin, D. (1999), Named entity extraction from speech: Approach and results using the textpro system, *in* 'DARPA Broadcast News Workshop', Morgan Kaufmann, pp. 51–54.

Babych, B. & Hartley, A. (2003), Improving Machine Translation Quality with Automatic Named Entity Recognition, *in* 'EAMT Workshop at EACL'.

Bikel, D., Schwartz, R. & Weischedel, R. (1999), 'An algorithm that learns what's in a name', *Machine Learning* **34**(1), 211–231.

Borthwick, A. (1999), A Maximum Entropy Approach to Named Entity Recognition, PhD thesis, University of New York.

Brants, T. (2000), TNT: A Statistical Part-of-Speech Tagger, *in* '6th Applied Natural Language Processing Conference', Seatle, pp. 224–231.

Brill, E. (1994), Some Advances in Rule-Based Part of Speech Tagging, *in* 'Twelfth National Conference on Artificial Intelligence', pp. 722–727.

Brill, E. (1995), Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging, *in* 'Computational Linguistics', Vol. 21, MIT Press, pp. 543–565.

Brown, P., Pietra, V., deSouza, P., Lai, J. & R, M. (1992), 'Class-based n-gram models of natural language', *Computational Linguistics* **18**(4), 467–479.

Buckland, M. & Gey, F. (1994), 'The relationship between recall and precision', *Journal of the American Society for Information Science* **45**(1), 12–19.

Cheung, W., Pang, K., Lyu, M., Ng, K. & King, I. (2000), Chinese optical character recognition for information extraction from video images, *in* 'The 2000 International Conference on Imaging Science, Systems and Technology (CISST'2000)', Vol. 1, CSREA Press, pp. 269–275.

Chinchor, N. (1997), MUC-7 Named Entity Task Definition , *in* 'Proceedings of the 7th Message Understanding Conference'.

Chinchor, N., Brown, E., Ferro, L. & Robinson, P. (1999), 1999 Named Entity Recognition Task Definition, Mitre and SAIC.

Chinchor, N., Brown, E. & Robinson, P. (1998), 'The hub-4 named entity task definition (version 4.8)'.

Cleverdon, C. (1972), 'On the inverse relationship of recall and precision', *Journal of Documentation* **28**, 195–201.

Collier, N., Park, H., Ogata, N., Takeishi, Y., Nobata, C. & Ohta, T. (1999), The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers, *in* 'EACL'99', pp. 271–272.

Curran, J. & Clark, S. (2003), Investigating GIS and Smoothing Maximum Entropy Taggers, *in* 'Seventh conference on natural language learning CONNL', Edmonton, Canada, pp. 164–167.

Dzeroski, S., Erjavec, T. & Zavrel, J. (2000), Morphosyntactic Tagging of Slovene:Evaluating Taggers and Tagsets, *in* 'Second International Conference on Language Resources and Evaluation (LREC 2000)', Athens.

Farmakiotou, D., Karkaletsis, V., Koutsias, J., Sigletos, G., Spyropoulos, C. & Stamatopoulos, P. (2000), Rule-Based Named Entity Recognition for Greek Financial Texts, *in* 'Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000)', Athens, pp. 75–78.

Farmakiotou, D., Karkaletsis, V., Samaritakis, G., Petasis, G. & Spyropoulos, C. (2002), Named Entity Recognition from Greek Web Pages, *in* 'Companion Volume of 2nd Hellenic Conference on AI (SETN-02)', pp. 91–202.

Fisher, B. (1998), 'Manual for ner99ns scoring pipeline', http://www.nist.gov.

Gotoh, Y. & Renals, S. (1999), Statistical annotation of named entities in spoken audio, *in* 'Workshop: Accessing Information in Spoken Audio', Cambridge.

Gotoh, Y., Renals, S. & Williams, G. (1999), Named entity tagged language models, *in* 'ICASSP-99', Phoenix.

Grishman, R. (1995), 'Tipster phase ii architecture design document (tinman architecture) version 1.52', ftp://www.cs.nyu.edu/pub/nlp/tipster/152.ps.

Grover, C., Matheson, C., Mikheev, A. & Moens, M. (2000), LT TTT - A Flexible Tokenisation Tool, *in* 'Second International Conference on Language Resources and Evaluation'.
*http://www.ltg.ed.ac.uk/software/ttt/index.html

Grover, C., McDonald, S., NicGearailt, D., Karkaletsis, V., Farmakiotou, D., Samaritakis, G., Petasis, G., Pazienza, T., Vindigni, M., Vichot, F. & Wolinski, F. (2002), Multilingual XML-Based Named Entity Recognition for E-Retail Domains, *in* '3rd International Conference on language Resources and Evaluation (LREC 2002)', Las Palmas, Canary Islands.

Harman, D. (1993), Overview of the First Text REtrieval Conference (TREC-1), *in* 'NIST Special Publication 500-207: The First Text REtrieval Conference (TREC-1)', Department of Commerce, National Institute of Standards and Technology.

Heeman, P. & Allen, J. (1997), Incorporating pos tagging into language modelling, *in* 'Eurospeech', Rhodes, Greece, pp. 2767–2770.

Henderson, J., Salzberg, S. & Fasman, K. (1997), 'Finding genes in DNA with a hidden markov model', *Journal of Computational Biology* **4**(2), 127–142.
*citeseer.nj.nec.com/article/henderson97finding.html

Hobbs, J., Appelt, D., Bear, J., Israel, D., Kameyama, M., Stickel, M. & Tyson, M. (1996), FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text, *in* 'Finite State Language Processing', MIT Press, pp. 383–406.

Horlock, J. & King, S. (2003*a*), Discriminative Methods for Improving Named Entity Extraction on Speech Data, *in* 'Eurospeech', Vol. 4, Geneva, pp. 2765–2768.

Horlock, J. & King, S. (2003*b*), Named Entity Extraction from Word Lattices, *in* 'Eurospeech', Vol. 2, Geneva, pp. 1265–1268.

Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunningham, H. & Wilks, Y. (1998), Description of the LaSIE-II System as used for MUC-7, *in* '7th Message Understanding Conference (MUC-7)'.

Iyer, R. & Ostendorf, M. (1997), Transforming out-of-domain estimates to improve in-domain language models, *in* 'Eurospeech', Vol. 4, pp. 1975–1978.

Karkaletsis, V., Spyropoulos, C., Souflis, D., Hachey, B., Pazienza, M., Vindigni, M., Cartier, E. & Coch, J. (2003), Demonstration of the CROSSMARC System, *in* 'Human Language Technology Conference (NAACL/HLT 2003', Edmonton, Canada.

Kim, J. (2001), Named Entity Recognition from Speech and Its Use in the Generation of Enhanced Speech Recognition Output, PhD thesis, University of Cambridge.

Kim, J. & Woodland, P. (2000), A rule-based named entity recognition system for speech input, *in* 'International Conference on Spoken Language (ICSLP-2000)'.

King, S., Frankel, J. & Richmond, K. (2003), 'www.cstr.ed.ac.uk/projects/espresso'.

Krupka, G. & Hausman, K. (1998), Isoquest: Description of the NETOWL$^{TM}$Extractor System as used in MUC-7, *in* 'Seventh Message Understanding Conference (MUC7)', Fairfax, Virginia.

Kubala, F., Schwartz, R., Stone, R. & Weishedel, R. (1998), Named entity extraction from speech, *in* 'Broadcast News Transcription and Understanding Workshop'.

Lin, D. (1998), Using Collocation Statistics in Information Extraction, *in* 'Seventh Message Understanding Conference (MUC7)', Fairfax, Virginia.

Malthus, T. (1798), 'An essay on the principle of population'.

McCarthy, M. (1999), 'Is there a basic spoken vocabulary: Technology and common sense', *SELL* **1**(2).

Mihalcea, R. & Moldovan, D. (2001), 'Document indexing using named entities'. *citeseer.nj.nec.com/526318.html

Mikheev, A., Moens, M. & Grover, C. (1998), Description of the LTG system used for MUC-7, *in* 'Seventh Message Understanding Conference (MUC7)', Fairfax, Virginia.
*citeseer.nj.nec.com/mikheev98description.html

Miller, D., Schwartz, R., Weischedel, R. & Stone, R. (1999), Named entity extraction from broadcast news, *in* 'DARPA Broadcast News Workshop', BBN, Morgan Kaufmann, pp. 37–40.

*National Association of Broadcasters Information Resource Center* (2000), http://www.nab.org/irc/Virtual/faqs.asp#World-Totals.

Nobata, C., Sekine, S., Isahara, H. & Grishman, R. (2002), Summerization System Integrated with Named Entity Tagging, *in* '3rd International Conference on language Resources and Evaluation (LREC 2002)', Las Palmas, Canary Islands.

Ostendorf, M. (1998), 'Training statistical language models with out of domain data', http://ssli.ee.washington.edu/ssli/projects/LM.html.

Pallett, D. (1997), Overview of the 1997 DARPA Speech recognition workshop, *in* 'DARPA Speech Recognition Workshop'.

Palmer, D., Burger, J. & Ostendorf, M. (1999), Information extraction from broadcast news speech data, *in* 'DARPA Broadcast News Workshop', Morgan Kaufmann, pp. 41–46.

Palmer, D., Ostendorf, M. & Burger, J. (2000), 'Robust information extraction from automatically generated speech transcriptions', *Speech Communication* .

Poibeau, T., Acoulon, A., Avaux, C., Beroff-Bnat, L., Cadeau, A., Calberg, M., Delale, A., De Temmerman, L., Guenet, A., Huis, D., Jamalpour, M., Krul, A., Marcus, A., Picoli, F. & Plancq, C. (2003), The multilingual named entity recognition framework, *in* 'EACL'03'.
*citeseer.nj.nec.com/584057.html

Przybocki, M., Fiscus, J., Garofolo, J. & Pallett, D. (1999), 1998 HUB-4 Information Extraction Evaluation, *in* 'DARPA Broadcast News Workshop', Morgan Kaufmann, pp. 13–18.

Renals, S., Gotoh, Y., Gaizauskas, R. & Stevenson, M. (1999), Baseline ie-ne experiments using the sprach/lasie system, *in* 'DARPA Broadcast News Workshop', Morgan Kaufmann, pp. 47–50.

Robinson, P., Brown, E., Burger, J., Chinchor, N., Douthat, A., Ferro, L. & Hirschman, L. (1999), Overview: Information Extraction from Broadcast News, *in* 'DARPA Broadcast News Workshop', Morgan Kaufmann, pp. 27–30.

Rosenfeld, R. (1994), 'The cmu-cambridge statistical language modeling toolkit v2'. http://svr-www.eng.cam.ac.uk/~prc14/toolkit_documentation.html ftp from ftp://ftp.cs.cmu.edu/project/fgdata/CMU_SLM.

Sampson, G. (1995), *English for the Computer*, Clarendon Press.
*http://www.grsampson.net/BEFC.html

Sampson, G. (2000), 'Christine corpus'.
*http://www.grsampson.net

Sekine, S. & Eriguchi, Y. (2000), Japanese Named Entity Evaluation Analysis of Results, *in* '15th International Conference on Computational Linguistics', pp. 1106–1110.

Sekine, S. & Nobata, C. (2003), Sentence Extraction with Information Extraction Technique, *in* 'Document Understanding Conferences', Edmonton, Canada.

Spilker, J., Weber, H. & Görz, G. (1999), Incorporating pos tagging into language modelling, *in* 'Eurospeech', pp. 2031–2034.

Srihari, R. & Li, W. (1999), Information Extraction Supported Question Answering, *in* 'Text REtrieval Conference (TREC-8)'.

Sun, J., J, G., Zhang, L., Zhou, M. & Huang, C. (2000), Extractubg the names of genes and gene products with a Hidden markov Model, *in* '18th International Conference on Computational Linguistics (COLING00)', Saarbrucken, Germany, pp. 201–207.

Sun, J., J, G., Zhang, L., Zhou, M. & Huang, C. (2002), Chinese Named Entity Identification Using Class-based, *in* '19th International Conference on Computational Linguistics (COLING02)', Taipei, Taiwan, pp. 967–973.

Taylor, P., Caley, R., Black, A. & King, S. (1999), 'Edinburgh speech tools library - system documentation', www.cstr.ed.ac.uk/projects/speech_tools.

Tufis, D., Dienes, P., Oravecz, C. & Varadi, T. (2000), Principled hidden tagset design for tiered tagging of hungarian, *in* '3rd International Conference of Language Resources and Evaluation (LREC2000)', Athens.

Tulic, M. (2002), 'Names'.
*http://www.anindexer.com/about/name/nameindex.html

van Eynde, F., Zavrel, J. & Daelemans, W. (2000), Part of Speech Tagging and Memmatisation for the Spoken Dutch Corpus, *in* 'Second International Conference on Language Resources and Evaluation (LREC 2000)', Athens.

Viterbi, A. (1967), 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', *IEEE Transactions on Information Theory* **13**, 260–269.

Wactlar (1999), New directions in video information extraction and summarization, *in* '10th DELOS Workshop', Santorini, Greece.

Wakao, T., Gaizauskas, R. & Wilks, Y. (1996), Evaluation of an Algorithm for the Recognition and Classification of Proper Names, *in* '16th International Conference on Computational Liguistics (COLING96)', Copenhagen, pp. 418–423.

Young, S., Russell, N. & Thornton, J. (1989), 'Token passing : a simple conceptual model for connected speech recognition systems', Technical Report TR38.