AraNLP: A Java-based Library for the Processing of Arabic Text

Maha Althobaiti, Udo Kruschwitz, Massimo Poesio
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
{mjaltha, udo, poesio}@essex.ac.uk

Abstract

We present a free, Java-based library named "AraNLP" that covers various Arabic text preprocessing tools. Although a good number of tools for processing Arabic text already exist, integration and compatibility problems continually occur. AraNLP is an attempt to gather most of the vital Arabic text preprocessing tools into one library that can be accessed easily by integrating or accurately adapting existing tools and by developing new ones when required. The library includes a sentence detector, tokenizer, light stemmer, root stemmer, part-of-speech tagger (POS-tagger), word segmenter, normalizer, and a punctuation and diacritic remover.

Keywords: Arabic Natural Language Processing, Java, Tools

1. Introduction

Languages with extensive morphological features require critical preprocessing in preparation for being subjected to Natural Language Processing (NLP) techniques. Arabic, a highly derivational and inflectional language, is characterised by a complex set of morphological features including gender, number, person, case, state, mood, and voice. In addition, Arabic has a set of clitics, which attach to the stem after affixes such as conjunctions, prepositions, future marks, definite articles, and pronouns.

The rich morphology of Arabic makes preprocessing the text an essential step for any Arabic NLP application. Moreover, word- and sentence-level preprocessing might differ from one NLP application to another within the same NLP task. For example, it has been shown that stemming is particularly effective for morphologically complex languages (Pirkola, 2001). Larkey et al. (2002), however, investigated the effectiveness of various degrees of stemming for both mono- and cross-language Arabic Information Retrieval (IR). Thus, many light stemmers were built so that each one of them removes a different set of attachable clitics and suffixes from the Arabic word. Their study found that a light stemmer, which removes stop words, definite articles, and the conjunction clitic (*wa*, و, 'and')[1] from the beginning of words, and a small number of suffixes from the end of words, is more effective for cross-language retrieval than a root stemmer, which searches for the root of each word. Habash and Sadat (2006) investigated the effect of different word-level preprocessing decisions for Arabic on Statistical Machine Translation (SMT) quality. They found that splitting off proclitics alone is most effective for large amounts of training data, while English-like tokenization is best when working with small amounts of training data.

A good number of tools are available for preparing Arabic text and developing Arabic NLP systems. These tools, in comparison with other languages, are still limited in coverage (Shaalan and Raza, 2009). In addition, integration and compatibility problems might occur with some tools. Thus, Arabic NLP researchers find themselves either modifying the existing processing tools to suit their needs, or building their own pipeline that consists of essential text preparation tools arranged in a particular order, depending on the application's requirements. Therefore, providing a library equipped with all or most of the tools essential for the processing of Arabic text (e.g., tokenization, sentence detection, word segmenter, stemming, POS-tagging), will make it possible to move Arabic NLP forward and to facilitate the reuse of already existing preprocessing algorithmic resources.

In this paper we present AraNLP, a Java-based toolkit for the processing of Arabic text. It supports the most important preprocessing steps, such as diacritic and punctuation removal, tokenization, sentence segmentation, part-of-speech tagging, root stemming, light stemming, and word segmentation. These tools are usually required to prepare the text for more advanced NLP tasks. Figure 1 illustrates a typical processing pipeline applying the tools provided by AraNLP. The goal of AraNLP is to gather most of the vital Arabic text preprocessing tools into one library that can be accessed easily. We incorporated missing tools and included existing algorithmic resources. AraNLP has already been used by Althobaiti et al. (2013) to prepare the Arabic text for their experiment and it successfully preprocessed the corpus. The library is available free online[2].

The remainder of this paper is structured as follows: Section 2 includes background information on Arabic and its morphological characteristics. Section 3 provides a detailed explanation of the processing tools included in the AraNLP library. Finally, the conclusion features our future plans.

2. The Characteristics of Arabic

2.1. Arabic Morphology

Arabic morphology is studied more than any other aspect of the language, because it is an essential and complex issue that concerns everyone developing Arabic NLP. In gen-

[1] Throughout the entire paper, Arabic words are represented as follows: (*Qalam transliteration*, Arabic word, 'English translation').

[2] https://sites.google.com/site/mahajalthobaiti/resources

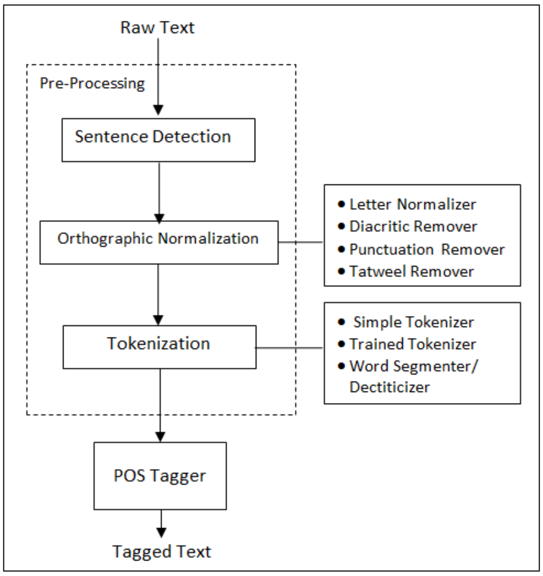


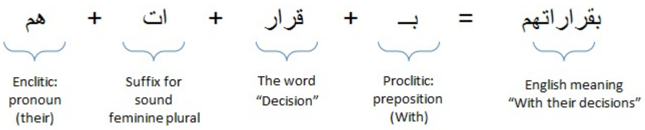
Figure 1: Typical processing pipeline of AraNLP

eral, morphology means the study of word structure or form (Ritchey, 1998). Arabic's rich morphology interacts with both orthography and syntax. The primary concept in morphology is the morpheme, the smallest unit of meaning in a language. For example, the word (*daarswn*, دارسون, 'scholars') consists of two morphemes: (*daars*, دارس, 'scholar') and (*wn*, ون, 'suffix for masculine plural'). There are three essential types of morphemes that concatenate to form words in Arabic:

- Stem: The term "stem" has two slightly different meanings. First, a stem can be the core part of a word that expresses the basic meaning and cannot be further divided into smaller morphemes (Payne, 2006). For example, the stem of the Arabic word (*musaafirwn*, مسافرون, 'travellers') is (*safar*, سفر, 'travel'). This usage has been followed by different studies in order to build stemmers for Arabic, like Khoja's stemmer (Khoja and Garside, 1999). Second, stem can refer to the part of the word that is common in all of its inflected forms (Kroeger, 2005). According to the second definition, the stem of the Arabic word (*musaafirwn*, مسافرون, 'travellers') is (*musaafir*, مسافر, 'traveller').

- Affixes: Affixes attach to the stem. There are three types of affixes: (a) prefixes that attach before the stem, (b) suffixes that attach after the stem, and (c) circumfixes that enclose the stem.

- Clitics: Clitics are morphemes that attach to the stem after affixes. A clitic has the syntactic characteristics of a word but is based phonologically on another word (Loos et al., 2004). There are several possible Arabic clitics that are distinguishable based on their position in the word. Proclitics come at the beginning of words. They usually represent conjunctions such as (*wa*, و, 'and'), (*bi*, ب, 'with'), (*fa*, ف, 'then'), (*ka*, ك, 'as'), and (*li*, ل, 'for'). Enclitics come at the end of words. They usually represent pronouns such as (*kumaa*, كما, 'your'), (*humaa*, هما, 'their'), and (*ye*, ي, 'my'). The following illustration shows how affixes and clitics attach to words:


هم
+
ات
+
قرار
+
بـ
=
بقراراتهم
Enclitic: pronoun (their)
Suffix for sound feminine plural
The word "Decision"
Proclitic: preposition (With)
English meaning "With their decisions"

2.1.1. Arabic Derivational and Inflectional Features

Arabic is a highly derivational and inflected language. These properties ease the process of expanding the Arabic vocabulary by using only the roots and the morphological patterns. There are approximately 10,000 independent roots and 85% of Arabic words are derived from triliteral roots (de Roeck and Al-Fares, 2000). Thus, new Arabic words are generated by applying derivational patterns to root forms. The Arabic language's derivational and inflected nature makes it possible to arrange Arabic words according to the roots from which they are derived.

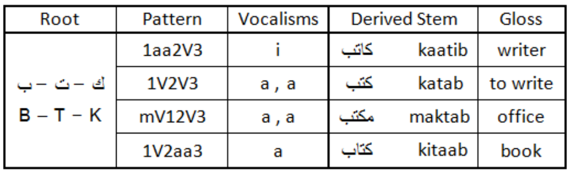
Derivation is the process by which new words are created. Three types of morphemes are required to create a word templatic stem: roots, patterns, and vocalisms. The root morpheme consists of three or four radicals[3], and in rare cases up to five. A root expresses a meaning that is shared amongst all of its derivations. For example, the root morpheme (د - ر - س, 'd-r-s') 'studying-relating' has many derivations, which share the same meaning, such as (*daras*, درس, 'to study'), (*daaris*, دارس, 'student'), (*diraasah*, دراسة, 'studying'), (*tadrys*, تدريس, 'teaching'), and (*madaars*, مدارس, 'schools'). The vocalism morpheme determines the short vowels to be used within patterns. There are three short vowels in Arabic: fatha (ـَ, 'a'), damma (ـُ, 'u'), and kasra (ـِ, 'i'). The pattern morpheme is a template in which vocalisms and root radicals are included. In the following examples a pattern is represented using a string of letters and numbers to mark where root radicals and vocalisms are inserted (Habash, 2010). For example, the pattern 1V2V3 indicates that there are three root radicals and two vocalisms in the same order as it is in the pattern. A pattern can also contain additional consonants or long vowels. For example,'tV1aa2V3' contains a constant 't' and a long vowel 'aa'. Table 1 provides some examples of stem construction.

The derivation process can be summarised with the following equation:

$TemplaticStem = Root + Pattern + vocalisms.$

In inflectional morphology, the lexical category and the core meaning of the word remains unchanged, but the extensions are always variable depending on a set of feasible features. In Arabic, four inflectional features are restricted to verbs. The following list illustrates the features applied to verbs and their values.

[3] Radicals is the term used when talking about root to mean consonants making up the root



Root	Pattern	Vocalisms	Derived Stem		Gloss
ك – ت – ب B – T – K	1aa2V3	i	كاتب	kaatib	writer
	1V2V3	a , a	كتب	katab	to write
	mV12V3	a , a	مكتب	maktab	office
	1V2aa3	a	كتاب	kitaab	book

Table 1: Examples of word stems derived from their roots

- Aspect: perfective, imperfective, imperative
- Mood: indicative, subjunctive, jussive
- Person: 1st person, 2nd person, 3rd person
- Voice: active, passive

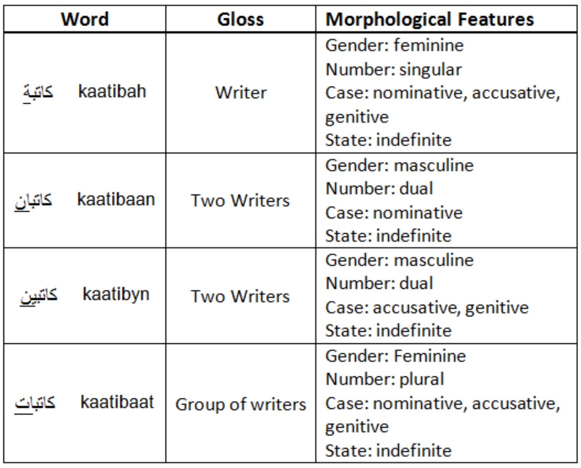
The inflectional features, which are applied only to Arabic nouns and their possible values are as follows:

- Case: nominative, accusative, genitive
- State: definite, indefinite

Moreover, the morphological features for verbs and nouns/adjectives are as follows:

- Gender: feminine, masculine
- Number: singular, dual, plural

We took one word stem (*kaatib*, كاتب, 'writer') derived from a root as explained in Table 1. Taking into account that this word is a noun, it can be inflected for gender, number, case, and state. Table 2 shows some examples of words inflected from the stem (*kaatib*, كاتب, 'writer').



Word		Gloss	Morphological Features
كاتبة	kaatibah	Writer	Gender: feminine Number: singular Case: nominative, accusative, genitive State: indefinite
كاتبان	kaatibaan	Two Writers	Gender: masculine Number: dual Case: nominative State: indefinite
كاتبين	kaatibyn	Two Writers	Gender: masculine Number: dual Case: accusative, genitive State: indefinite
كاتبات	kaatibaat	Group of writers	Gender: Feminine Number: plural Case: nominative, accusative, genitive State: indefinite

Table 2: Examples of words inflected from the stem

3. The Modules

3.1. Sentence Boundary Detection

Sentence boundary detection is the process of isolating independent sentences. Finding the correct sentence boundaries is more important for some NLP task than others, and more critical for some languages and colloquial dialects than others, as well. This is due to the ambiguity of punctuation marks, and the misuse of these marks in some cases (Grefenstette and Tapanainen, 1994). Many Arabic NLP studies rely on known Arabic sentence separators (, ; : . ?) to segment raw text into sentences, and even depend on syntactic analysis to resolve the ambiguity of punctuation, as in the study of Ouersighni (2001). However, we found no study on processing Modern Standard Arabic (MSA) that provides evaluation results for the sentence detectors they used.

Depending purely on a few rules and one's intuition that some punctuation marks, more often than not, are used to delimit sentences is not an optimal solution, especially for NLP tasks to which sentence detection is crucial. Therefore, we decided to build a maximum entropy model[4] for identifying sentence boundaries in raw Arabic text. The corpus on which the model has been trained consists of 1,838 sentences collected from 59 Arabic Wikipedia documents of various genres. According to the machine learning package, the input format of the training data should be one sentence per line. Contextual features of the potential sentence boundary, including the tokens preceding and following the token that contains the end-of-sentence character and the spaces that delimited the tokens, are used[5]. The trained model performed well on a testing corpus made up of 871 sentences, with 0.97 precision and recall reaching nearly 0.98.

3.2. Tokenization

The standard pre-processing step for many NLP tasks is tokenization, which divides a string of written language into its component tokens. For less complex languages, tokenization usually involves splitting punctuation, and some affixes off of the words. On the other hand, morphologically rich languages, like Arabic, require a more extensive tokenization process to separate different types of clitics and particles from the word. This complex tokenization is usually called word segmentation (Green and DeNero, 2012). The word segmenter provided by AraNLP is discussed in detail in Section 3.4.

More relevant to our current topic is simple tokenization, which only splits off punctuation and non-alphanumeric characters from words. For example, the word (*faransaa.*, فرنسا., 'France.') is separated into two tokens 'فرنسا' and '.'.

Although this type of tokenization may seem simple and require no disambiguation, there are some NLP tasks for which it may be unsuitable, like Named Entity Recognition (NER). Occasionally, punctuation and numbers appear in the names of entities such as product names and numbers (e.g., 'Olympus SP-820UZ digital camera'). The names of these types of NEs are translated into Arabic with the same numbers and punctuation. In addition, specific domains (e.g., University domain) introduce new types of entities (e.g., course code and room numbers) that contain punctuation and other symbols that should be considered single tokens (Althobaiti et al., 2012). Thus, it is necessary to take into account the careful separation of non-alphanumeric

[4] The Maxent machine learning package, available as part of the OpenNLP project was used to train both Sentence Detector and Tokenizer

[5] http://svn.apache.org/repos/asf/opennlp/trunk/opennlp-tools/src

characters from the words. To address this issue, we built a model that detects token boundaries using MaxEnt machine learning. The training corpus we used consists of around 52,000 tokens from the Arabic Wikipedia collection. The training algorithm uses contextual features such as the two characters preceding and following the position where a space might be added, the tokens preceding and following that position without crossing sentence boundaries, and class information about the preceding and following two characters (e.g., letters, numbers, end-of-sentence characters). A testing corpus with 21,000 tokens was used to evaluate the trained tokenizer, which achieved a 0.97 precision and recall score.

3.3. Stemming

Stemming is the process of reducing derived or inflected words to their stems or original roots. Stemming is considered a key step in many IR applications and is a common component in almost any text mining system. In fact, stemming is essential for Arabic IR (Croft et al., 2010). Research has shown that Arabic stemming is challenging because of its highly inflectional and derivational nature (Aljlayl and Frieder, 2002; Larkey et al., 2007). In addition, language independent stemmers, which focus on suffix removal, have proved useless to the Arabic language (Larkey et al., 2002). The work on stemming can be divided into two main types, according to the aims of the stemming process. Some work tries to reduce the words to their original roots (root-extraction stemmers), while other work aims to extract and remove affixes (light stemmers). Each type of stemmer has its own significance. In other words, a stemmer that performs well with certain applications may perform poorly with others (Aljlayl and Frieder, 2002; El-Beltagy and Rafea, 2011).

AraNLP supports the two types of stemmers in order to encompass all potential needs. We implemented several versions of light stemming akin to those suggested by Larkey et al. (2002). They tried to remove strings that appeared as affixes more often than they appeared at the beginning or end of Arabic words without affixes. Their light stemming versions have been thoroughly tested and proved efficient for NLP tasks such as NER (Abdul-Hamid and Darwish, 2010).

As for the root stemmer, we incorporated the algorithm provided by Khoja and Garside (1999). They relied on morphological analysis to develop their stemmer by first removing layers of prefixes and suffixes, and then checking a set of roots and patterns to specify whether the remainder was a known root with a known pattern. This stemmer proved sufficient, as it helps to improve the performance of some systems (Larkey and Connell, 2006). The Khoja Stemmer has been modified, so that it can be used easily within our AraNLP library. We also exempted 'stop words' from stemming instead of removing them, as in the Khoja Stemmer. We used the same list of 168 Arabic stop words provided by Khoja and Garside.

3.4. Word Segmentation & POS Tagging

As already mentioned, Arabic is a highly morphological language with a considerable number of bound clitics and affixes such as conjunctions, particles, prepositions, and pronouns. Segmenting bound clitics and affixes reduces data sparsity and simplifies analyzing the text syntactically. These benefits are extremely useful for some NLP tasks such as POS tagging. A large number of possible segmentation levels[6] can be applied to Arabic text, according to the types of clitics to be split. For example, a shallow segmenter may only separate conjunctions and prepositions from the word. More complex segmentation may break up the word into its stem and different affixes (e.g., conjunctions, interrogative clitics, definite articles, future verbal particle). Examples of available Arabic NLP tools that perform clitic tokenization are (MADA+TOKEN)[7], and AMIRA[8].

Our library links up to the Stanford Arabic word segmenter and POS tagger. The segmenter produces the three Penn Arabic Treebank (PATB) clitic segmentations: conjunctions, prepositions, and pronouns. The main advantage of this word segmenter is that it processes raw text quickly in comparison to other word segmenters, as its implementation is based on a sequence classifier (Conditional Random Fields). The Stanford POS tagger is based on a maximum-entropy technique. We noticed that the Arabic POS tagger quality increased when the text is segmented in order to separate bound clitics from words. In AraNLP, you can use the POS tagger directly, as word segmentation is carried out automatically before POS tagging.

3.5. Arabic Normalization

A large number of NLP tasks require the text be free of punctuation or diacritics, if not both. Therefore, we implemented a simple tool to remove punctuation and diacritics. It removes all three forms of diacritics, as suggested by Diab et al. (2007): Vowel Diacritics, Nunation Diacritics and the Shadda. The tool removes the following default punctuation: commas (,), semi-colons (;), colons (:), exclamation points (!), question marks (?), hyphens (-), En dashes (–), apostrophes ('), points of ellipsis (...), Arabic commas (،), Arabic semi-colons (؛), and Arabic question marks (؟).

For many NLP applications, another issue that should be addressed in raw Arabic text is inconsistent variations. For example, different forms of alif (أ، آ، ا، إ) might be written interchangeably; another example is alif maqsurah and regular dotted Yaa' (ى، ي) which are usually used interchangeably at the final position of the word. The same is true for taa marbutah and haa (ة، ه). These misspelling errors in Arabic affect 11% of all words in the Penn Arabic Treebank (PATB) (Habash, 2010). AraNLP provides a different level of orthographic normalization that can be carried out on Arabic text to reduce noise and data sparsity. This includes normalization of the hamzated alif to a bare alif (alif without hamzah), normalization of taa marbutah to haa, normalization of the dotless yaa (alif maksura)

[6] It is also called segmentations/decliticization schemes (Habash, 2010).

[7] https://flintbox.com/public/project/8348

[8] https://www.flintbox.com/public/project/8335/

to yaa, and the removal of tatweel (stretching character). AraNLP enables the user to customize the level of normalization according to the application's need. In addition, the punctuation can easily be added or deleted from the list of punctuation marks.

4. Conclusion

AraNLP provides developers with the essential modules for processing raw Arabic text, which can be used together or individually. The modules can be rearranged to specially design pipelines according to the particular requirements of any NLP task. The output of each module can easily be used as input for the next one. The library itself is platform-independent and uses Unicode to represent Arabic characters inside Java classes in order to avoid compatibility problems with any integrated development environment.

In future, we plan to develop and incorporate more essential tools for NLP applications in order to produce a more comprehensive Java-based library for Arabic NLP.

5. Acknowledgements

We would like to thank three anonymous reviewers for their valuable comments and constructive suggestions, which helped us to revise the paper.

6. References

Abdul-Hamid, A. and Darwish, K. (2010). Simplified feature set for Arabic Named Entity Recognition. In *Proceedings of the 2010 Named Entities Workshop*, pages 110–115. Association for Computational Linguistics.

Aljlayl, M. and Frieder, O. (2002). On Arabic search: improving the retrieval effectiveness via a light stemming approach. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 340–347. ACM.

Althobaiti, M., Kruschwitz, U., and Poesio, M. (2012). Identifying Named Entities on a University Intranet. In *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, pages 94–99. IEEE.

Althobaiti, M., Kruschwitz, U., and Poesio, M. (2013). A Semi-supervised Learning Approach to Arabic Named Entity Recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 32–40, Hissar, Bulgaria, September. INCOMA Ltd. Shoumen, BULGARIA.

Croft, W., Metzler, D., and Strohman, T. (2010). *Search Engines: Information Retrieval in Practice*. Alternative Etext Formats. ADDISON WESLEY Publishing Company Incorporated.

de Roeck, A. and Al-Fares, W. (2000). A morphologically sensitive clustering algorithm for identifying Arabic roots. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 199–206. Association for Computational Linguistics.

Diab, M., Ghoneim, M., and Habash, N. (2007). Arabic diacritization in the context of statistical machine translation. In *Proceedings of MT-Summit*.

El-Beltagy, S. and Rafea, A. (2011). An accuracy-enhanced light stemmer for Arabic text. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(2):2.

Green, S. and DeNero, J. (2012). A class-based agreement model for generating accurately inflected translations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 146–155. Association for Computational Linguistics.

Grefenstette, G. and Tapanainen, P. (1994). *What is a Word, what is a Sentence?: Problems of Tokenisation*. Rank Xerox Research Centre.

Habash, N. and Sadat, F. (2006). Arabic preprocessing schemes for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL.*, pages 49–52. ACL.

Habash, N. Y. (2010). Introduction to Arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Khoja, S. and Garside, R. (1999). Stemming Arabic Text. *Lancaster, UK, Computing Department, Lancaster University*. http://zeus.cs.pacificu.edu/shereen/research.htm.

Kroeger, P. (2005). *Analyzing Grammar: An Introduction*. Cambridge University Press.

Larkey, L. S. and Connell, M. E. (2006). Arabic information retrieval at UMass in TREC-10. Technical report, DTIC Document.

Larkey, L. S., Ballesteros, L., and Connell, M. E. (2002). Improving stemming for Arabic Information Retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–282. ACM.

Larkey, L., Ballesteros, L., and Connell, M. (2007). Light Stemming for Arabic Information Retrieval. In Soudi, A., Bosch, A. d., and Neumann, G., editors, *Arabic Computational Morphology*, volume 38 of *Text, Speech and Language Technology*, pages 221–243. Springer Netherlands.

Loos, E., Anderson, S., Day, D., Jordan, P., and Wingate, J. (2004). *Glossary of linguistic terms*, volume 29. SIL International.

Ouersighni, R. (2001). A major offshoot of the DIINAR-MBC project: AraParse, a morphosyntactic analyzer for unvowelled Arabic texts. In *ACL 39th Annual Meeting*, pages 9–16.

Payne, T. E. (2006). *Exploring language structure: a student's guide*. Cambridge University Press.

Pirkola, A. (2001). Morphological typology of languages for IR. volume 57, pages 330–348. MCB UP Ltd.

Ritchey, T. (1998). General morphological analysis. In *16th EURO Conference on Operational Analysis*.

Shaalan, K. and Raza, H. (2009). NERA: Named entity recognition for Arabic. volume 60, pages 1652–1663. Wiley Online Library.