Minimally-supervised Methods for Arabic Named Entity Recognition

Maha Jarallah Althobaiti

A thesis submitted for the degree of Doctor of Philosophy (PhD)

School of Computer Science and Electronic Engineering
University of Essex

February 2016

Declaration

This thesis is the result of my own work, except where explicit reference is made to the work of others, and has not been submitted for another qualification to this or any other university. This thesis does not exceed the word limit for the respective Degree Committee.

Maha Althobaiti

To my loving father and mother!

The greatest source of inspiration, support and motivation, since my childhood

Acknowledgments

Pursuing a PhD degree at the University of Essex has been a wonderful and unforgettable journey in my life. I would like to take this opportunity to thank those who have helped me on this journey, especially those who have guided me along the way.

First of all, I would like to gratefully and sincerely thank my supervisors, Prof. Udo Kruschwitz and Prof. Massimo Poesio, for their guidance and valued advice throughout my doctoral journey. Their trust, knowledge, and research skills have been an inspiration for me and will deeply influence my career and future life. It has been a privilege and a pleasure to undertake the PhD under their excellent supervision. My gratitude and special thanks is extended to my examiners Prof. Paolo Rosso from University of Valencia and Dr. Luca Citi from University of Essex for their interesting discussion and helpful suggestions.

A word of gratitude is also extended to my fellow PhD students and all members of the Language and Computation (LAC) Group at University of Essex for making my experience here a pleasant one and for all the interesting academic discussions we have had. I especially thank Deirdre Lungley for her advice and prompt answers to all of my questions.

I would also like to thank University of Essex for providing a comfortable environment and friendly atmosphere for students. Studying at the University of Essex, particularly the School of Computer Science and Electronic Engineering, was the most rewarding experience I have ever had.

A very special gratitude is reserved for my beloved parents and caring siblings. I admit that any success in my life would not have been achieved without their love, care, continuous encouragement, and support.

Abstract

Named Entity Recognition (NER) has attracted much attention over the past twenty years, as a main task of Information Extraction. The current dominant techniques for addressing NER are supervised methods that can achieve high performance, but require new manually annotated data for every new domain and/or genre change. Our work focuses on approaches that make it possible to tackle new domains with minimal human intervention to identify Named Entities (NEs) in Arabic text. Specifically, we investigate two minimally-supervised methods: semi-supervised learning and distant learning. Our semi-supervised algorithm for identifying NEs does not require annotated training data or gazetteers. It only requires, for each NE type, a seed list of a few instances to initiate the learning process. Novel aspects of our algorithm include (i) a new way to produce and generalise the extraction patterns (ii) a new filtering criterion to remove noisy patterns (iii) a comparison of two ranking measures for determining the most reliable candidate NEs. Next, we present our methodology to exploit Wikipedia structure to automatically develop an Arabic NE annotated corpus. A novel mechanism is introduced, based on the high coverage of Wikipedia, in order to address two challenges particular to tagging NEs in Arabic text: rich morphology and the absence of capitalisation. Neither technique has yet achieved performance levels comparable to those of supervised methods. Semi-supervised algorithms tend to have high precision but comparatively low recall, whereas distant learning tends to achieve higher recall but lower precision. Therefore, we present a novel approach to Arabic NER using a combination of semi-supervised and distant learning techniques. We used a variety of classifier combination schemes, including the Bayesian Classifier Combination (BCC) procedure, recently proposed for sentiment analysis. According to our results, the BCC model leads to an increase in performance of 8 percentage points over the best minimally-supervised classifier.

Contents

List of Figures

List of Tables

List of Acronyms

NLP	Natural Language Processing
NER	Named Entity Recognition
NE	Named Entity
IE	Information Extraction
IR	Information Retrieval
QA	Question Answering
ACE	Automatic Content Extraction
CoNLL	Conference on Computational Natural Language Learning
MUC	Message Understanding Conference
TREC	Text Retrieval Conference
SL	Supervised Learning
SSL	Semi-supervised Learning
DL	Distant Learning
BCC	Bayesian Classifier Combination
IBCC	Independent Bayesian Classifier Combination
CA	Classical Arabic
MSA	Modern Standard Arabic
NP	Noun Phrase
SVMs	Support Vector Machines
CRFs	Conditional Random Fields
HMM	Hidden Markov Model
ME	Maximum Entropy
PMI	Pointwise Mutual Information

Part I

Introduction

Chapter 1

Introduction

1.1 Motivation

Named Entity Recognition (NER) is the Information Extraction (IE) task concerned with identifying and classifying Named Entities (NEs) found in texts. A Named Entity (NE) is simply anything that can be referred to with a proper name [Jurafsky and Martin, 2009]. For example, a generic news-oriented Arabic named-entity recogniser might find the organisation شركة أبل "Apple Inc." and the locations كوبرتينو "Cupertino" and كاليفورنيا "California" in the text: يقع المقر الرئيسي لشركة أبل في كوبرتينو كاليفورنيا, "Apple Inc. is headquartered in Cupertino California".

NER is one of the most popular tasks of IE that has been utilised in several Natural Language Processing (NLP) applications. For example, in reference to Information Retrieval (IR), which aims to identify and retrieve relevant documents from a set of data according to an input query, Guo *et al.* [2009] reported that about 71% of the queries in search engines contain NEs. Therefore, improving the retrieval of documents for queries containing NEs would increase the performance of the whole IR system. Furthermore, extracting NEs benefits the Question Answering (QA) systems, which aim to take questions as input and return concise and precise answers. In fact, 80% of the 200 questions that comprised the Question Answering track competition in Text REtrieval Conference (TREC-8) asked for NEs [Srihari and Li, 1999] (e.g., who (Person), where (Location), when (Time)). NER can be used to recognise the NEs within the question, which will help later in identifying relevant documents and passages. It can also be used to find the precise answer in relevant

passages, if the question asks for an NE [Abouenour *et al.*, 2012; Hammo *et al.*, 2002]. In addition, NER has served as an experimental sandbox for a number of techniques [Bikel *et al.*, 1997; Liao and Veeramachaneni, 2009; McCallum and Li, 2003].

Most research on NER has been carried out in English [Nadeau and Sekine, 2007], but a significant amount of research has also been conducted in Arabic [Shaalan, 2014]. Many studies revolving around Arabic NER are based on hand-crafted rules [Elsebai *et al.*, 2009; Shaalan and Raza, 2007]. The most recent studies use supervised machine learning to automatically produce sequence labelling algorithms starting from a collection of annotated training examples [AbdelRahman *et al.*, 2010; Benajiba and Rosso, 2008; Darwish, 2013]. Formerly, there was a lack of freely available resources for Arabic NER. The situation has changed, however, so that there are currently a number of Arabic NE annotated corpora [Alotaibi and Lee, 2014; Benajiba *et al.*, 2007; Mohit *et al.*, 2012]. Nonetheless, changing the domain or expanding the set of NE classes[1] still always requires domain-specific experts and new annotated data. That is, even state-of-the-art NER systems do not necessarily perform well on other domains [Darwish, 2013; Poibeau and Kosseim, 2001].

Early work in NER in the 1990s was aimed primarily at extraction from newspaper articles [Grishman and Sundheim, 1996], but since about 1998, there has been increasing interest in entity identification from several types of text styles, such as biomedical text, telephone conversation transcripts, weblogs, and social networks [Krallinger *et al.*, 2013; Ritter *et al.*, 2011; Settles, 2004]. Therefore, dependence on manually annotated data is insufficient for the variety of domains and texts produced by fast-changing technologies. It is also known that manually annotated data is expensive and cumbersome to create. As a result, these data often lack coverage. We have noticed that most of the current annotated corpora for Arabic NER can only support supervised learning with a small number of classes [Benajiba *et al.*, 2007; Mohit *et al.*, 2012]. For example, Benajiba *et al.* [2007] built an Arabic NER system in order to detect and classify NEs into one of four classes: Person, Location, Organisation, and Miscellaneous. Another study by Shaalan and Raza [2009] presented an attempt to extract the 10 most important categories of NEs in Arabic

[1]The terms 'type', 'class', and 'category' are used interchangeably in this thesis.

script. Their system, however, used hand-crafted rules and dictionaries, which could not cover all names and were specific to the domain. Thus, we aim to move the state-of-the-art in NER forward by reducing the burden of supervision in training and by developing minimally-supervised algorithms for NER, particularly for Arabic NER.

1.2 Research Objectives

The main objectives of this research are summarised as follows:

- Develop learning algorithms for Arabic NER that can be adapted to different domains with minimal human intervention; specifically, test and compare minimally-supervised techniques.
- Compare the performance of minimally-supervised methods to those of supervised learning-based NER systems.
- Exploit semi-structured knowledge on the web to compensate for the lack of distinguishing orthographic features in Arabic.
- Study the possibility of combining multiple minimally-supervised techniques to improve the overall quality of the output.

1.3 Contributions

The research described in this thesis accomplished several goals and made many contributions to the field of NER for the Arabic language. These can be summarised as follows:

- We introduced a capitalisation threshold for Arabic, a language that does not have capitalised letters, using the high coverage of Wikipedia.
- We proposed a mechanism to handle the rich morphology in Arabic, and eliminate the need to perform any deep morphological analysis by exploiting Wikipedia features such as anchor texts and redirects.

- We developed a method to automatically create NE annotated corpus from Wikipedia for the standard NE classes. Our Wikipedia-derived Corpus (WDC) is available for free online to the community of research[2].

- We presented in-depth experiments with semi-supervised learning and a pattern-based approach to Arabic NER across many NE classes.

- We presented a novel approach to Arabic NER using a combination of semi-supervised learning and distant supervision.

- We used the Independent Bayesian Classifier Combination (IBCC) scheme for NER and compared it to traditional voting methods.

- We introduced the classifier combination restriction as a means of controlling how and when the predictions of individual classifiers are combined.

- We built AraNLP, a Java-library that provides essential computational preprocessing tools for the Arabic Language. This library is explained in a published paper and available online to the community of research[3].

1.4 Thesis Structure

The structure of the rest of this thesis is as follows. Chapter 2 provides some information about Arabic and its history. It discusses the Arabic script used to write Modern Standard Arabic (MSA) and its elements. The chapter also discusses the aspects of Arabic that are most relevant to NLP in general and NER in particular such as derivational and inflectional features and complex morphology. Chapter 3 offers the necessary background on NER research, covering the main aspects of NER such as NE types, annotation schemes, and evaluation measures. The chapter then describes the techniques that have been used by NER systems and the new approaches that are currently being studied in the research community to overcome the disadvantages of traditional methods. Chapter 4 covers Arabic

[2]The WDC data set is available at `https://sites.google.com/site/mahajalthobaiti`
[3]AraNLP is available at `https://sites.google.com/site/mahajalthobaiti`

NER. The chapter first discusses the challenges. Then, it lists the available Arabic linguistic resources according to their genres and the NE types they encompass. Finally, the chapter describes the basic computational tasks that are necessary for Arabic NER systems and the approaches that have been adopted by Arabic NER systems. Chapter 5 presents our semi-supervised algorithm for identifying NEs in Arabic text. The algorithm adopts the bootstrapping technique that only needs a set of seeds in order to initiate the learning process. The chapter explains the algorithm in detail and presents extensive experiments with their results. Chapter 6 describes our methodology to automatically generate an NE annotated corpus from Wikipedia using minimal time and human intervention. The chapter also illustrates the mechanism we introduced to exploit the high coverage of Wikipedia in order to address two challenges particular to tagging NEs in Arabic text: rich morphology and the absence of capitalisation. The evaluation results of the created corpus is also discussed in the chapter. Chapter 7 presents a novel approach to Arabic NER, which involves combining semi-supervised and distant learning techniques using a variety of classifier combination schemes, including the Bayesian Classifier Combination (BCC) procedure. The chapter also introduces the classifier combination restriction as a means of controlling how and when the predictions of multiple classifiers are combined. It also shows the effect of this restriction on the performance of the combination methods. Chapter 8 conveys our conclusions and highlights the main findings of this research. It also explains our vision for possible future work.

List of Publications Related to This Work

The work in this thesis has been published in the following peer-reviewed journal and proceedings:

- Althobaiti, M., Kruschwitz, U., and Poesio, M. (2015). Combining Minimally-supervised Methods for Arabic Named Entity Recognition. Transactions of the Association for Computational Linguistics (TACL), 3, 243-255.

- Althobaiti, M., Kruschwitz, U., and Poesio, M. (2014). AraNLP: a Java-based Library for the Processing of Arabic Text. In Proceedings of the Ninth International Con-

ference on Language Resources and Evaluation (LREC), Reykjavik, Iceland, pages 4134-4138.

- Althobaiti, M., Kruschwitz, U., and Poesio, M. (2014). Automatic Creation of Arabic Named Entity Annotated Corpus Using Wikipedia. In Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, pages 106-115.

- 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), Hissar, Bulgaria, pages 32-40.

Other Publications

- Lungley, D., Trevisan, M., Nguyen, V., Poesio, M. and Althobaiti, M. (2013). GALATEAS D2W: A Multi-lingual Disambiguation to Wikipedia Web Service. In Exploration, Navigation and Retrieval of Information in Cultural Heritage Workshop (ENRICH) at SIGIR, Dublin, Ireland.

- Althobaiti, M., Kruschwitz, U., and Poesio, M. (2012). Identifying Named Entities on a University Intranet. In Computer Science and Electronic Engineering Conference (CEEC), Colchester, UK, pages 94-99.

Part II

Background and Related Work

Chapter 2

Characteristics of Arabic

To possess a comprehensive background of Arabic NER, one should have a thorough knowledge of three essential areas: (a) the Arabic language, (b) the definition of NE task, the current challenges, and the latest research and (c) the research of Arabic NER. These three areas of interrelated works are presented in this chapter and the next two chapters (i.e., Chapter 3 and Chapter 4) respectively.

This chapter gives a general overview of the Arabic language, emphasising the aspects that concern the topic of this thesis. First, Section 2.1 presents the history and variants of Arabic. Second, elements and symbols of the Arabic script are explained in Section 2.2. Next, the structure and rich morphological features of Arabic words are illustrated in detail in Section 2.3. Finally, Section 2.4 explains the structure of Arabic sentences.

2.1 Introduction

Arabic belongs to the Semitic family of languages that spread throughout North Africa and Middle East. The living language members of the Semitic family include Arabic, Aramaic (including Syriac), Hebrew, and Amharic. The Semitic language family has a long and distinguished literary history and has played important roles in the cultural landscape of the Middle East for more than 4,000 years [Hetzron, 2013; Testen, 2015].

Today, Arabic is the largest living language member of the Semitic group in terms of the number of speakers. It is spoken by more than 400 million people in the Arab world, and used by more than 1.5 billion Muslims worldwide [UNESCO, 2012, 2015].

Arabic includes three main forms: Classical Arabic (CA), Modern Standard Arabic (MSA), and Colloquial Arabic [Elgibali, 2005; Habash, 2010; Korayem *et al.*, 2012]. Classical Arabic (CA) is essentially the form of the language found in the Quran[1]. After the spread of Islam, CA becomes the prominent language of religious worship and sermons [Watson, 2007]. Most Arabic religious texts and many historical Arabic manuscripts are written in CA.

Modern Standard Arabic (MSA) is the standard language in use today. It is the official language of all Arab League countries and the only form of Arabic taught in schools at all stages. It is used as the primary language of writing, most formal speech, and education. The Syntax, morphology, and phonology of MSA is based on CA. The main differences between MSA an CA are (a) the orthography of conventional written Arabic, and (b) the vocabulary, as MSA is lexically much more modern than CA in order to include modern technical terms [Al-Johar, 1999; Farber *et al.*, 2008].

Colloquial Arabic refers to the many informal regional spoken dialects used for everyday communication. It is not taught in schools due to its irregularity. Unlike the widespread use of MSA across all Arab countries, colloquial Arabic varies dramatically based on many factors, but is primarily dependent on geography and social class. So, colloquial Arabic differs not only from one Arab country to another, but also across regions within the same country. Although colloquial Arabic is rarely written and mostly spoken, written colloquial Arabic can be mainly found in social media communication, as well as a certain amount of literature such as plays and poetry [Shaalan, 2014].

2.2 Elements of the Arabic Script

The Arabic script is the second most widely used alphabetic writing system in the world after Latin script [Testen, 2015]. It is written from right to left and there is no distinction between uppercase and lowercase letters. The basic Arabic alphabet has 28 letters and 3 different types of diacritical marks. In both printed and handwritten Arabic, most letters within a word are connected in a cursive style.

[1]The holy book of Islam.

Arabic script was first used to type Arabic texts, most notably the Quran. The spread of Islam made the Arabic script became the writing system for other languages, such as Persian, Urdu, Sindhi, Malay, and Kurdish. Adaptations of the Arabic script for other languages resulted in the addition of some letters as well as the redefinition of some letters' phonetic values [Kaye, 1996]. Since the focus of this thesis is on the Arabic language, specifically MSA in NLP, this section discusses the Arabic script used in MSA.

The basic 28 letters of the Arabic alphabet corresponds to Arabic's 28 consonantal sounds. Most of these basic letters take different shapes depending on whether they are at the beginning, middle or end of a word, so they may show four distinct forms (initial, medial, final or isolated). Only six letters (و، ز، ر، ذ، د، ا) do not have a distinct medial form and have to be written with their final form without being connected to the succeeding letter. Their initial forms are the same as their isolated forms. Table 2.1 shows the 28 basic letters used in MSA.

Name	Letter	Contextual Forms			Name	Letter	Contextual Forms		
	Isolated	Initial	Medial	Final		Isolated	Initial	Medial	Final
alif	ا	-	-	ـا	Daad	ض	ضـ	ـضـ	ـض
baa'	ب	بـ	ـبـ	ـب	Taa'	ط	طـ	ـطـ	ـط
taa'	ت	تـ	ـتـ	ـت	Zaa'	ظ	ظـ	ـظـ	ـظ
thaa'	ث	ثـ	ـثـ	ـث	`ayn	ع	عـ	ـعـ	ـع
jym	ج	جـ	ـجـ	ـج	ghayn	غ	غـ	ـغـ	ـغ
Haa'	ح	حـ	ـحـ	ـح	faa'	ف	فـ	ـفـ	ـف
khaa'	خ	خـ	ـخـ	ـخ	qaaf	ق	قـ	ـقـ	ـق
daal	د	-	-	ـد	kaaf	ك	كـ	ـكـ	ـك
dhaal	ذ	-	-	ـذ	laam	ل	لـ	ـلـ	ـل
raa'	ر	-	-	ـر	mym	م	مـ	ـمـ	ـم
zayn	ز	-	-	ـز	nuwn	ن	نـ	ـنـ	ـن
syn	س	سـ	ـسـ	ـس	haa'	ه	هـ	ـهـ	ـه
shyn	ش	شـ	ـشـ	ـش	waaw	و	-	-	ـو
Saad	ص	صـ	ـصـ	ـص	yaa'	ى	يـ	ـيـ	ـى

Table 2.1: Arabic basic letters.

Diacritical marks are of great importance in Arabic, as they are used to help children and those learning Arabic to pronounce the words correctly. They are usually optional;

written Arabic words can be diacritised, partially diacritised or entirely undiacritised. But when it comes to religious texts, children's educational texts, and some classical poetry, diacritical marks are written in full [Elkateb *et al.*, 2006; Habash, 2010; Ryding, 2005]. In some cases, diacritical marks are crucial to help readers in reducing the ambiguity of certain words. For example, the meaning of the word سُلَّم *sullam* "Ladder"[2] can be changed totally to mean "peace" by only changing the diacritical marks: سِلم *silm*.

There are three types of diacritical marks: *vowel*, *nunation*, and *shaddah*. Vowel diacritical marks represent three short vowels in Arabic: *fatHah* ـَ */a/*, *dammah* ـُ */u/*, and *kasrah* ـِ */i/*. The *sukuwn* ـْ diacritical mark means there is no vowel. The nunation diacritical marks can only occur in indefinite words. They are pronounced as a short vowel followed by */n/* sound which is unwritten. The shaddah diacritical mark ـّ is used to represent a double consonant. It can be combined with vowel or nunation diacritical marks as illustrated in Table 2.2.

Types of Diacritics	**Types**	**Examples & Pronunciation**
Vowel	fatHah /a/	مَ /ma/
	dammah /u/	مُ /mu/
	kasrah /i/	مِ /mi/
Nunation	fatHat-an /an/	مً /man/
	dammat-un /un/	مٌ /mun/
	kasrat-in /in/	مٍ /min/
Shaddah Consonant Doubling Diacritic	shaddah	مّ /mm/
	shaddah with vowel	مَّ /mma/, مِّ /mmi/, مُّ /mmu/
	shaddah with nunation	مًّ /mman/, مٍّ /mmin/, مٌّ /mmun/
No vowel	sukun	مْ /m/

Table 2.2: Arabic diacritical marks.

There are additional symbols that can be treated as separate letters and/or special combinations of letters and additional diacritical marks. One example of such a symbol is

[2]Throughout this thesis, Arabic words are supplemented with transliterations and translations in the following order: Arabic-word *Qalam-transliteration* "English-translation".

hamzah, which can be considered a separate letter (ء) or a diacritical mark with other letter forms (ء، ا، إ، آ، ؤ، ئ) [Habash *et al.*, 2007; Soudi *et al.*, 2007]; see Table 2.3.

Furthermore, the *taa' marbuwTah* ة, is a morphological marker for a feminine ending like in مكتبة *maktabah* "'Library'. It is actually a hybrid letter merging the form of the letters *haa'* ه and *taa'* ت. Also the *alif maqSuwrah* ى, which looks like a dotless *yaa'* ي, is a morphological marker used to mark feminine endings and underlying word roots. It is a merge of the forms of the letters *alif* ا and *yaa'* ي [Habash, 2010].

The standard computer encodings of Arabic, such as CP1256, ISO-8859, and Unicode, consider the *hamzah* letter marks to be separate letters (not diacritical marks). They also consider *taa' marbuwTah* and *alif maqSuwrah* to be separate letters [Habash, 2010].

Hamzah Types	**Shapes**	**Examples**		
hamzah-on-the-line	ء	الماء	almaa'	water
alif with hamzah above	أ	أخبار	'akhbaar	news
alif with hamzah below	إ	إياك	'iyaaka	thee
alif with maddah above	آ	آمال	~aamaal	hopes
waaw with hamzah above	ؤ	مؤرخ	mu`rrkh	historian
yaa' with hamzah above	ئ	سائل	saa'il	questioner

Table 2.3: The hamzah letter forms.

2.3 Arabic Morphology

In linguistics, morphology is defined as the study of internal word formation. Arabic is characterised by complex, productive morphology, with a root-and-pattern word-formation mechanism. The morphology of Arabic is the most studied aspect of Arabic when working on Arabic NLP, because of its important interactions with orthography and syntax [Habash, 2010]. In addition, the morphological analyser is an important component of Arabic NLP systems, which must be aware of the word structure whatever the goal of the systems. In what follows, a brief description of Arabic morphology is given with examples for clarification.

2.3.1 Word Structure

The primary concept in morphology is the morpheme, the smallest unit of meaning in a language. For example, the word باحثون *baaHithwn* "researchers" consists of two morphemes: باحث *baaHith* "researcher" and ون *wn* "suffix for masculine plural". There are three essential types of morphemes - stems, affixes, and clitics - that concatenate to form words in Arabic in the following order:

"$Arabic\ Word = Proclitic(s) + Prefix(es) + Stem + Suffix(es) + Enclitic(s)$".

Detailed information about each type is provided below.

- 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 word مسافرون *musaafirwn* "travellers" is سفر *safar* "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 مسافرون *musaafirwn* "travellers" is مسافر *musaafir* "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. The prefixes and suffixes, attached to the stems, are predictable and limited to a set of features like gender, number, person, aspect, and so on [Farber *et al.*, 2008]. For example, the prefix ن *na* is added to the beginning of imperative verbs in order to indicate the 'first person plural'. More examples of affixes are shown in Table **??**.

- Clitics: Clitics are morphemes that attach to the stem after affixes (i.e., they are attached to the inflected base words). A clitic has the syntactic characteristics of the word attached to it, 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

(a) A sample of prefixes

Example	Conditions	Within the Word	English Translation
ت	**POS:** Verb **Person:** 3rd **Number:** singular **Gender:** feminine **Aspect:** imperfective **Voice:** active	ت + كتب = تكتب t + katab = taktub	She writes
ي	**POS:** Verb **Person:** 3rd **Number:** singular **Gender:** masculine **Aspect:** imperfective **Voice:** active	ي + كتب = يكتب y + katab = yaktub	He writes

(b) A sample of suffixes

Example	Conditions	Within the Word	English Translation
ون	**POS:** Noun **Case:** nominative **Gender:** masculine **Number:** plural	كاتب + ون = كاتبون Kaatib + wn = kaatibwn	Writers (males)
ين	**POS:** Noun **Case:** accusative, genitive **Gender:** masculine **Number:** plural	كاتب + ين = كاتبون Kaatib + yn = kaatibyn	Writers (males)

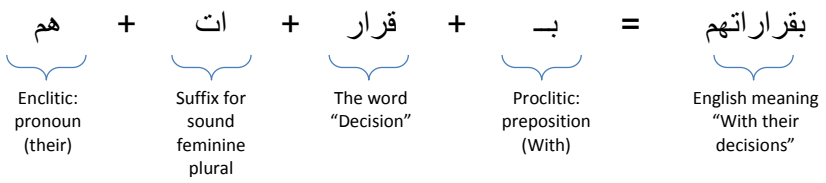
(c) A sample of circumfixes

Example	Conditions	Within the Word	English Translation
ت + ...+ان	**POS:** Verb **Person:** 3rd **Number:** Dual **Gender:** feminine **Aspect:** imperfective **Voice:** active	ت + كتب + ان = تكتبان t + katab + aan = taktubaan	They write (females)
ي + ...+ان	**POS:** Verb **Person:** 3rd **Number:** Dual **Gender:** masculine **Aspect:** imperfective **Voice:** active	ي + كتب + ان = يكتبان y + katab + aan = yaktubaan	They write (males)

Table 2.4: Types of affixes.

(e.g., و *wa*, ف *fa*), prepositions (e.g., بـ *bi*, كـ *ka*, لـ *li*.), future particles (e.g., سـ *sa*), and the definite article ال *Al* "the". Enclitics come at the end of words. They represent pronouns such as كما *kumaa* "your", هما *humaa* "their", and ي *ye* "my".

The following illustration shows how affixes and clitics attach to words:



2.3.2 Derivational and Inflectional Morphology

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 from other words. 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 ـ ر ـ س "d-r-s" "studying-relating" has many derivations, which share the same meaning, such as درس *daras* "to study", دارس *daaris* "student", دراسة *diraasah* "studying", تدريس *tadrys* "teaching", and مدارس *madaars* "schools". The vocalism morpheme determines the short vowels to be used within patterns. There are three short vowels in Arabic: Fathah ـَ */a/*), Dammah ـُ */u/*, and Kasrah ـِ */i/*. The pattern morpheme is a template in which vocalisms and root radicals are included. In the following examples we illustrate a pattern, as in [Habash, 2010], using a string of letters and numbers to mark

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

where root radicals and vocalisms are inserted. 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, the pattern *tV1aa2V3* contains a constant "t" and a long vowel "aa". Table 2.5 provides some examples of stem construction.

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

Table 2.5: Examples of stems derived from their roots.

The derivation process can be summarised with the following equation:

$$Stem = Root + Pattern + vocalisms.$$

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.

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

The inflectional features, that 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

For example, if we take one word stem كاتب *kaatib* "writer", derived from a root as explained in Table 2.5, then it can be inflected for number, gender, case, and state. Table 2.6 shows some examples of words inflected from the stem كاتب *kaatib* "writer".

Word		English Translation	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 2.6: Examples of words inflected from the same stem.

2.4 Arabic Syntax

Syntax is the linguistic area concerned with describing how words are arranged together to compose phrases and sentences. It can also be defined as the study of the formation of sentences and the relationship between their component parts [Aoun *et al.*, 2009; Testen, 2015].

2.4.1 Essential Syntactic Principles

There are two essential syntactic principles that affect the structure of Arabic phrases and sentences: مطابقة *agreement/concord*, and عمل *government* [Ryding, 2005].

Agreement or concord is the compatibility of words in a phrase or clause, so that, each word in a phrase conforms to and reflects the others' features. For example, adjectival modifiers of nouns agree with the gender and case of the noun they modify, that is, a masculine singular noun takes a masculine singular adjective. A verb is masculine singular if it has a masculine singular subject and so forth [Ryding, 2005].

The two terms *agreement* and *concord* are often used interchangeably. The use of these terms synonymously, however, proves inaccurate because the term *agreement* refers to matching between the verb and its subject, whereas the term *concord* is used to refer to matching between nouns and their dependants (adjectives, other nouns, or pronouns). Categories of agreement and concord in Arabic include: inflection for gender, number, and person for verbs and pronouns and inflection for gender, number, definiteness, and case for nouns and adjectives [Blake, 1994].

The second syntactic principle is *government* in which certain words (عوامل *'awaamil* "governing words") cause others to inflect in particular ways - not in agreement with the governing word, but as a result of the effect of the governing word. The typical governing words in Arabic are prepositions, particles, and verbs. For example, when a preposition precedes a noun, it causes the noun to be in the genitive case. Another example is transitive verb which causes a direct object to be in the accusative case.

2.4.2 Sentence Structure

Arabic sentences are generally classified as either *verbal* or *nominal* sentences[4]. A nominal sentence is defined as a sentence which begins with a noun, while a verbal sentence is one that begins with a verb. In each case, the sentence can be simple, compound, or complex.

Generally speaking, a sentence composed of a single clause is a simple sentence. The combination of more than one clause using coordinating conjunctions forms a compound

[4]Nominal sentence is also called copular/equational sentence.

sentence. A coordinating conjunction is a particle that connects two words, phrases or clauses together such as و "and", او/ام "or", فـ "Then/and so,". Each clause in the compound sentence remains independent; therefore, the conjunctions have no effect on the morphology or syntax of the following clause, but build up the sentence contents in an additive way.

The combination of a main clause and one or more subordinate clauses, is called a complex sentence. Subordinate clauses are of three main types: complement clauses, adverbial clauses, and relative clauses. Unlike coordinating conjunctions, many Arabic subordinating conjunctions have a grammatical effect on the structure of the following clause. For example, أنّ *'anna* "that" is followed by a clause whose subject is either a noun in the accusative case or a suffixed pronoun. لكي *likay* "in order to" is followed by a verb in the subjunctive mood [Ryding, 2005]. Table 2.7 shows examples of the simple, compound, and complex sentences.

Simple Sentence	تفتحت الأزهار. The flowers bloomed.
Compound Sentence	جاء الربيع وتفتحت الأزهار. The spring came and the flowers bloomed.
Complex Sentence	يبذل الفريق قصارى جهده لكي يحافظ على موقعه في التصنيف العالمي. The team makes the maximum effort in order to maintain its position in the world rankings.

Table 2.7: Sentence Types.

Nominal Sentence

The structure of the nominal sentence consists of two parts: مبتدا *subject/topic*, and خبر *predicate/complement*. The subject can be a proper noun, definite noun, or pronoun and the predicate can be a proper noun, indefinite noun, adjective, prepositional phrase, adverbial phrase, or a sentence. The adjectival predicate agrees with the subject in gender and number [Habash, 2010].

- $Subject_{pronoun}$ $Predicate_{proper\_noun}$

هو زيد

He Zaid

"He is Zaid"

- $Subject_{proper\_noun}$ $Predicate_{adjective}$

  زيد نائم

  Zaid asleep

  "Zaid is asleep"

- $Subject_{definite\_noun}$ $Predicate_{adjective}$

  الكتاب مفيد

  the-book useful

  "The book is useful."

- $Subject_{definite\_noun}$ $Predicate_{prepositional\_phrase}$

  الضيف في البيت

  the-guest in the-home

  "The guest is in the home."

- $Subject_{definite\_noun}$ $Predicate_{adverbial\_phrase}$

  النافورة امام المبنى

  the-fountain front the-building

  "The fountain is in front of the building."

- $Subject_{definite\_noun}$ $Predicate_{nominal\_sentence}$

  الحديقة سورها جميل

  the-garden fence-its beautiful

  "The garden's fence is beautiful."

- $Subject_{definite\_noun}$ $Predicate_{verbal\_sentence}$

  الطالب يكتب الدرس

  الطالب يكتب-ضمير مستتر الدرس

the-student writes-(3rd person singular masculine hidden pronoun) the-lesson

"The student writes the lesson"

Verbal Sentence

The verbal sentence starts with a verb, which is followed by a subject and one or more objects. The verb agrees with the subject in gender, number, person. The most basic form of the verbal sentence has only a verb with a conjugated or pro-dropped pronominal subject. Like other languages, some verbs in Arabic can have more than one object.

- $Verb\ Subject_{hidden\_pronoun}$

  كتب

  كتب-ضمير مستتر

  wrote-(3rd person singular masculine hidden pronoun)

  "He wrote"

- $Verb\ Subject_{pronoun}$

  كتبنا

  كتب-نا

  Wrote-we

  "We wrote"

- $Verb\ subject_{hidden\_pronoun}\ Object_{prnoun}$

  كتبها

  كتب-ضمير مستتر ها

  wrote-(3rd person singular masculine hidden pronoun) it

  "He wrote it."

- $Verb\ subject_{noun}\ Object_{noun}$

  كتب الكاتب الكتاب

  wrote the-author the-book

  "The author wrote the book."

2.5 Summary

This chapter presented a brief description of Arabic script, its elements, history, and variants. It also illustrated the main characteristics of Arabic that concern many NLP tasks including NER. The derivational and inflectional nature of the language was also covered in detail. Finally, this chapter explained the structure of Arabic sentences with various examples, showing possible part-of-speech categories and the expected order of words in a sentence.

Chapter 3

Named Entity Recognition (NER)

In the previous Chapter, we provided a general overview of Arabic. We discussed the Arabic script used to write Modern Standard Arabic (MSA). We also discussed the aspects of Arabic that are of most concern to Natural Language Processing (NLP) in general and Named Entity Recognition (NER) in particular, such as derivational and inflectional features, and complex morphology.

This chapter illustrates the essential aspects of the NER problem. Section 3.1 presents the standard definition of the NER task. It also offers a brief description of the chronology of NER history, starting from the first emergence of the term *Named Entity* and continuing through the important evaluation campaigns and conferences that gave the NER task its popularity. Section 3.2 describes the NE types and hierarchies that are proposed and defined in many conference shared-tasks and research. The annotation schemes and the evaluation measures for NER are explained in Section 3.3 and Section 3.4 respectively. Section 3.5 demonstrates the techniques that have been used in NER systems and new approaches that researchers currently study to overcome the current challenges of traditional techniques.

3.1 Definition and History

Named Entity Recognition (NER) is one of the most essential aspects of Information Extraction (IE). Both conceptually and practically, NER involves two sub-problems: the detection of the names of entities in text, and the classification of these names into a predefined set of categories of interest, such as the names of People, Organisations, Locations, Products, etc. For example, in the sentences “Mark lives in London. He has worked at University College

London since 2012.", "Mark", "London", and "University College London" are examples of Person, Location and Organisation entities, respectively.

The term *Named Entity* (NE) was first introduced to the Natural Language Processing (NLP) community at the sixth Message Understanding Conference (MUC-6) in 1996. The MUC conferences were the evaluation campaigns that contributed significantly to IE research and provided the benchmark for Named Entity systems. At that time, the NE task basically entailed identifying the names of all the People, Organisations, and Geographic Locations in a text. The task also involved identifying Time, Date, Currency, and Percentage expressions [Grishman and Sundheim, 1996].

Since MUC-6, several campaigns have arisen to evaluate NE in different languages. For example, the Multilingual Entity Task evaluations (MET-1 and MET-2) provided a new opportunity to assess progress on the NE task in Chinese, Japanese, and Spanish [Chinchor, 1998; Merchant *et al.*, 1996]. The NE task was one of the two tasks assessed and organised by the Information Retrieval and Extraction Exercise (IREX) project for Japanese [Sekine and Isahara, 1999]. Also, the HAREM was the first evaluation contest for Portuguese NER [Santos *et al.*, 2006]. The Conferences on Computational Natural Language Learning (CoNLL 2002 and CoNLL 2003) included shared tasks on NER in four languages: English, German, Dutch and Spanish. Arabic was one of the three languages that were investigated in the task of entity detection and recognition in the Automatic Content Extraction Program (ACE) [Doddington *et al.*, 2004].

3.2 Types of Named Entities (NEs)

The three most studied types of named entities in the literature are Person, Location, and Organisation [Sekine *et al.*, 2002; Tjong Kim Sang and De Meulder, 2003]. These types were collectively called "ENAMEX" at the 1996 MUC-6 competition. MUC-6 also introduced the "TIMEX" and "NUMEX" types, which cover Date, Time, Money, and Percent [Grishman and Sundheim, 1996].

Furthermore, the CoNLL conferences added the type Miscellaneous (MISC) to include

all proper names that were not covered by "ENAMEX" types (i.e., Person, Location, and Organisation) [Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003]. Examples of proper names covered by Miscellaneous (MISC) are nationalities (e.g., English, Brazilian, ...etc), and events (e.g., 2016 Summer Olympics, the World Cup 2014, ...etc). The detailed list of CoNLL-2003 NE types can be found in Appendix B.

Five entity types were presented at ACE 2003: Person, Organization, Location, Facility, and Geo-Political entity. The Facility entity referred to permanent man-made structures and buildings. Geo-Political entities represented geographical regions defined by political and/or social groups [Strassel *et al.*, 2003]. This set of entity types was extended in ACE 2004 to include Weapon and Vehicle. Another 40 subtypes were also introduced by ACE 2004. At ACE 2005, three subtypes - Individual, Group, and Indeterminate - were added to the type Person. Table 3.1 illustrates the types and subtypes of ACE 2005.

Entity Types	**Entity Sub-types**
Person(PER)	Group, Indeterminate, Individual
Organisation(ORG)	Commercial, Educational, Entertainment, Government, Media, Medical-Science, Non-Governmental, Religious, Sports
Location(LOC)	Address, Boundary, Celestial, Land-Region-Natural, Region-General, Region-International, Water-Body
Facility(FAC)	Airport, Building-Grounds, Path, Plant, Subarea-Facility
Geo-political entities(GPE)	Continent, County-or-District, GPE-Cluster, Nation, Population-Center, Special, State-or-Province
Weapon (WEA)	Biological, Blunt, Chemical, Exploding, Nuclear, Projectile, Sharp, Shooting, Underspecified
Vehicle (VEH)	Air, Land, Subarea-Vehicle, Underspecified, Water

Table 3.1: ACE 2005 entity types and subtypes.

Many hierarchies of named entity types are presented in the literature. Brunstein [2002] proposes a BBN hierarchy that contains 29 types and 64 subtypes, representing

answer categories for the Question Answering (QA) task. These 29 categories include slightly modified versions of the five ACE types: Person, Location, Organisation, GPE, and Facility. For example, museums fall into the Facility category according to ACE annotation guidelines, but in with Organisation type in BBN categories. The BBN categories also include the MUC types - Time, Date, Money, and Percent - again with some modifications. Sekine and Nobata [2004] try to cover most frequent name types and rigid designators in a newspaper using a named entity hierarchy that includes 200 extended categories and fine-grained subcategries.

The number of the NEs types continues to rise in parallel with increasing studies dedicated to specific domains. So, studies that are interested in bioinformatics usually target entities from that domain such as Protein, DNA, and RNA. In addition, some types of named entities are targeted according to specific needs such as handling Course Code, and Room Number, on University intranet [Althobaiti *et al.*, 2012].

3.3 Annotation Schemes for NEs

According to Leech [1997], *annotation* is the process of adding linguistic information to an electronic textual data of spoken and/or written language in order to denote the phenomena to be studied. Annotating text for the NER task is done at the word level and requires a human linguist to label each word with its corresponding NE type. The most commonly used NE tags are already covered in Section 3.2. This section describes the two main schemes for NE annotation.

3.3.1 Inline annotation

Inline annotation directly places annotations and original text in the same file and has two main formats.

3.3.1.1 Column-based format

In this format, adopted in CoNLL 2002 and CoNLL 2003, the data consists of two columns separated by a single space. Each word is put on a single line and there is an empty line

after each sentence. The first column in each line is the token and the second column is the NE tag. The tag represents the NE type and its position within entity (e.g., the tag *I-PER* is used for tokens inside an NE chunk of type Person). There are various forms of NE chunks representation illustrated as follows:

- **IOB1**: Tokens tagged with O are Outside of NEs and the I tag is used for tokens Inside NEs. The first token inside an NE chunk immediately following another NE chunk of the same type receives a B tag.

Apple	I-ORG
Inc.	I-ORG
is	O
a	O
technology	O
company	O
headquartered	O
in	O
Cupertino	I-LOC
California	B-LOC

- **IOB2**: It is same as IOB1, except that a B tag is used to mark the beginning of an NE chunk.

Apple	B-ORG
Inc.	I-ORG
is	O
a	O
technology	O
company	O
headquartered	O
in	O
Cupertino	B-LOC
California	B-LOC

- **IOE1**: An E tag is used to mark the last token of an NE chunk immediately preceding another NE chunk of the same type. The I is used for tokens Inside an NE chunk. The O tag is for tokens Outside of NEs.

Apple	I-ORG
Inc.	I-ORG
is	O
a	O
technology	O
company	O
headquartered	O
in	O
Cupertino	E-LOC
California	I-LOC

- **IOE2**: IOE2 is same as IOE1, except that an *E* tag is used to mark every token, which exists at the end of the NE chunk.

Apple	I-ORG
Inc.	E-ORG
is	O
a	O
technology	O
company	O
headquartered	O
in	O
Cupertino	E-LOC
California	E-LOC

- **BILOU**: An *L* tag marks the last token of an NE chunk containing more than or equal to two tokens, and *U* marks a unit-token NE chunk (i.e., NE chunk containing a single token). The *B*, and *I*, are used for tokens at the beginning, and inside an NE chunk. The *O* tag is for tokens Outside NEs. BILOU is also known in the literature as BEISO representation, where *S* (single) is used instead of *U*,and *E* (end) is used instead of *L* [Kudo and Matsumoto, 2001; Ratinov and Roth, 2009].

Apple	B-ORG
Inc.	L-ORG
is	O
a	O
technology	O
company	O
headquartered	O
in	O
Cupertino	U-LOC
California	U-LOC

IOB1 was first introduced as a representation of text chunks by Ramshaw and Marcus [1995] and was used for base NP chunking. IOB2, IOE1, and IOE2 are three alternatives to IOB1 first introduced by Sang and Veenstra [1999]. In CoNLL 2002, the chunk tag format IOB2 was adopted for NE task, while CoNLL 2003 used the IOB1 format.

3.3.1.2 SGML format

SGML (Standard Generalized Markup Language) is used to specify tag sets and document markup language. In the MUC evaluation campaigns, the markup was defined in SGML Document Type Descriptions (DTDs) and used to annotate named entities in the text. The NE annotations are enclosed in angled brackets; SGML tag elements represent the NE categories, while subcategories are represented by an SGML tag attribute. The following shows the SGML format for the example sentence:

Apple Inc. is a technology company headquartered in Cupertino California.

<ENAMEX TYPE="ORGANISATION">Apple Inc. </ENAMEX>is a technology company headquartered in <ENAMEX TYPE ="LOC">Cupertino </ENAMEX><ENAMEX TYPE ="LOC">California </ENAMEX>

3.3.2 Standoff annotation

Standoff annotation is used to separate annotations from the primary data they refer to. Primary data can be the original text or other annotation layers [Thompson and McKelvie, 1997]. The ACE program made use of the XML-based stand-off annotation format

since complex representations of IE tasks were required. Figure 3.1 illustrates the ACE annotation format for the aforementioned sentence example.

Apple Inc. is a technology company headquartered in Cupertino California.

```
<entity ID="E1" TYPE="ORG" SUBTYPE="Commercial" CLASS="SPC">
  <entity_mention ID="E1-1" TYPE="NAM" LDCTYPE="NAM">
    <extent>
      <charseq START="1" END="11"> Apple Inc. </charseq>
    </extent>
    <head>
      <charseq START="1" END="11"> Apple Inc. </charseq>
    </head>
  </entity_mention>
  <entity_attributes>
    <name NAME="Apple Inc.">
      <charseq START="1" END="11"> Apple Inc. </charseq>
    </name>
  </entity_attributes>
</entity>
<entity ID="E2" TYPE="GPE" SUBTYPE="Population-Center" CLASS="SPC">
  <entity_mention ID="E2-1" TYPE="NAM" LDCTYPE="NAM" ROLE="LOC">
    <extent>
      <charseq START="53" END="62"> Cupertino </charseq>
    </extent>
    <head>
      <charseq START="53" END="62"> Cupertino </charseq>
    </head>
  </entity_mention>
  <entity_attributes>
    <name NAME="Cupertino">
      <charseq START="53" END="62"> Cupertino </charseq>
    </name>
  </entity_attributes>
</entity>
<entity ID="E3" TYPE="GPE" SUBTYPE="State-or-Province" CLASS="SPC">
  <entity_mention ID="E3-1" TYPE="NAM" LDCTYPE="NAM" ROLE="LOC">
    <extent>
      <charseq START="63" END="73"> California </charseq>
    </extent>
    <head>
      <charseq START="63" END="73"> California </charseq>
    </head>
  </entity_mention>
  <entity_attributes>
    <name NAME=" California">
      <charseq START="63" END="73"> California </charseq>
    </name>
  </entity_attributes>
</entity>
```

Figure 3.1: ACE stand-off annotation sample.

3.4 Evaluation Measures for NER

NER systems are evaluated based on a comparison of their output and that of human experts. Thus, several measures have been defined to evaluate the quality of NER systems against manually annotated data. However, the three evaluation measures most widely used at the conferences and evaluation campaigns of the NE task are: Precision, Recall, and F_β [Chinchor, 1998; Marsh and Perzanowski, 1998; Tjong Kim Sang and De Meulder, 2003]. These measures can be computed as follows:

$$Precision = \frac{\text{Number of detected NEs that are correctly classified by the system}}{\text{Number of NEs that are detected by the system}},$$

$$Recall = \frac{\text{Number of detected NEs that are correctly classified by the system}}{\text{Number of actual NEs in the gold standard corpus}}.$$

$$F_\beta = (\beta^2 + 1)\frac{\text{Precision*Recall}}{\beta^2 * (\text{Precision + Recall})}$$

where β is the the relative weight of precision and recall. It is set to 1 when the precision and recall are the same weight. In this case, the measure is called the traditional F-measure or balanced F-score and can be expressed as follows:

$$F_{\beta=1} = 2 * \frac{\text{Precision*Recall}}{(\text{Precision + Recall})}$$

Next, we illustrate the scoring techniques used in various evaluation campaigns, taking into account different ways of dealing with various errors produced by the NER systems as shown in Table 3.2.

MUC: NER systems are evaluated in MUC conferences for *type* and *span*. *Type* refers to the system's ability to classify the detected NE correctly and *span* indicates the system's ability to predict the full span of the detected NE. So, the system wins two points when

Correct Output	NER System Output	Error
<Organisation> McDonald </Organisation>	<Person> McDonald </Person>	NE Type: wrong NE boundaries: right
<person> John McDonald </person>	<Person> John </Person> McDonald	NE Type: right NE boundaries: wrong
As <person> Oliver </person> gave	<Organisation> As Oliver </Organisation> gave	NE Type: wrong NE boundaries: wrong
<Location> London </Location>	London	Missing
surgeon	<Person> surgeon </Person>	Incorrectly hypothesising

Table 3.2: Types of errors produced by NER systems.

it is able to predict the full span of an NE and classify it correctly. On the other hand, one point is given to the system when it manages to only identify the full boundaries of the detected NE, but fails to assign the correct *type*. This evaluation technique tests all possible errors that the system might produce.

CoNLL: Evaluation in CoNLL does not give partial credit, but requires an exact match of *type* and *span*. So, the scoring process only awards points when the system predicts the full span of an NE and classifies it correctly. This is useful for some systems and applications, but not for generic applications. For example, some applications in the bioinformatics field need to determine whether or not a particular sentence contains a specific gene. In this case, only information about NE existence in the sentence is required, but detecting the full span of an entity name is not [Tsai *et al.*, 2006].

ACE: Evaluation in ACE is complex because the ACE program defines complex tasks (e.g., Mention Detection (MD), Co-reference Resolution), which are extensions of NER. In addition, many subtypes are taken into consideration when detecting entities. The organisers of the ACE program have chosen a measure called *ACE-value* to score the performance

of participants. This measure assigns different weights to different types and different levels of entity mentions (pronominal, nominal, or named entity mentions). Moreover, different penalties are also assigned to different types of errors of the MD and Co-reference systems. ACE evaluation may be the most powerful evaluation scheme currently available. However, it is extremely complex and makes error analysis difficult.

3.5 Approaches to NER

Although the research aiming to identify NEs in running text started in 1991 [Rau, 1991], it has grown quickly and steadily since the first IE competitions dedicated to the task in 1995 [Grishman and Sundheim, 1996]. Since then, many studies participating in different conferences and journals have contributed to the field. Work on English preceded work on other languages. Thus, this section is dedicated to the presentation of recent and important English NER works, which have achieved milestones in the evolution of NER research.

Early studies on NER were mainly based on handcrafted rules. For example, five systems out of eight in the MUC-7 competition were rule-based systems [Nadeau and Sekine, 2007]. Many of the Rule-based systems outperformed other participants in the MUC-6 and MUC-7 competitions [Marsh and Perzanowski, 1998; Sundheim, 1996]. Machine learning techniques, however, have played an important role in moving NER research forward by providing different learning methods, as explained in the following sections.

3.5.1 Supervised Learning

Supervised learning (SL) has been the dominant technique for NER since 1997 [Pantel and Pennacchiotti, 2006]. *Nymble* [Bikel *et al.*, 1997] and *IdentiFinder* [Bikel *et al.*, 1999], developed by BBN using the Hidden Markov Model (HMM), are two of the earliest NER systems that utilised machine learning techniques. Other widely used supervised learning techniques are Maximum Entropy (ME) [Borthwick *et al.*, 1998], Decision Trees [Sekine *et al.*, 1998], Support Vector Machines (SVMs) [Isozaki and Kazawa, 2002], and Conditional Random Fields (CRFs) [McCallum and Li, 2003].

The general approach of supervised learning is to train a model on a large collection of annotated data and to study the features of positive and negative examples. The trained model captures examples that have the same features of annotated data on unseen examples. Therefore, the performance of a supervised system depends on the collection of annotated data. That is, good systems require a large amount of manually annotated training corpora with high coverage [Mihalcea and Chklovski, 2003]. Formally, the sequence of tokens in the unstructured text is denoted as $X = x_1...x_n$. Each x_i has to be classified into one of a set Y of labels. This gives a tag sequence $Y = y_1...y_n$. A labelled example is the pair $< x, y >$ and the set of n training examples is taken from the space $(X \times Y)$. The training set could be written as $< x_1, y_1 >, < x_2, y_2 >, ... < x_n, y_n >$. Each feature can be thought of as a function $f : (x, y, i) \mapsto R$ that takes as its argument the sequence x_i, and returns a real-value capturing features of the ith token and its neighbouring tokens when it is assigned label y [Jones, 2005].

Although a considerable amount of annotated data is available for many languages, including Arabic [Zaghouani, 2014], changing the domain or expanding the set of classes always requires domain-specific experts and new annotated data, both of which demand time and effort. For example, Althobaiti *et al.* [2012] developed a classifier to recognise the three named entities, which are constantly referenced in the University domain: Person, Room Number, and Course Code. A person name is considered a common named entity, while Course Code and Room Number are specific to the University domain, which required building a specific corpus that contained manually annotated course codes and room numbers. Therefore, much of the current research on NER focuses on approaches that require minimal human intervention such as semi-supervised learning [Abney, 2010; Nadeau *et al.*, 2006] and distant learning [Mintz *et al.*, 2009; Nothman *et al.*, 2013].

3.5.2 Semi-supervised Learning

Semi-supervised Learning (SSL) [Abney, 2010] has been used for various NLP tasks, including NER [Chapelle *et al.*, 2006; Collins and Singer, 1999]. *Bootstrapping* is the most common semi-supervised learning technique [Nadeau and Sekine, 2007]. It involves a small

degree of supervision, such as a set of seeds, to initiate the learning process [Nadeau and Sekine, 2007]. An early study that introduced mutual bootstrapping and proved highly influential is [Riloff and Jones, 1999]. They presented an algorithm that begins with a set of seed examples of a particular entity type (e.g., London is entity of type City). Then, all contexts (e.g., "the city of <X >", "hotels in <X >") found around these seeds in a large corpus are compiled, ranked, and used to find new examples.

Pasca *et al.* [2006] use the same bootstrapping technique as Riloff and Jones [1999], but apply the technique to very large corpora (100 million web documents). They demonstrate that starting from a seed of 10 example facts, it is possible to generate one million facts with a precision rate of 88%. They also use distributional similarity to generate synonyms (words with the same semantic class), which allows pattern generalisation. For example, the word *Brazilian*, in the pattern (*<NE >is a Brazilian writer born in <NE >*), can be replaced with similar words (*Japanese, Chinese, Mexican,* and *Portuguese*), resulting in the induction of new patterns such as (*<NE >is a Japanese writer born in <NE >*).

Etzioni *et al.* [2005] propose a system called "KnowItAll" that aims to automate the process of extracting large collections of facts, such as names of cities, movies, or scientists from the web in a domain-independent and scalable manner. The system has the ability to extract information without any manually labelled training instances, starting with a set of classes (such as City, Country,...etc) and a set of generic extraction patterns to generate candidate facts. For example, the generic pattern 'NP1 such as NPList2' indicates that the head of each simple noun phrase (NP) in the list 'NPList2' is a member of the class named in 'NP1'. So for the class City, KnowITAll extracts four candidate cities from the sentence "Tours are provided to cities such as London, Paris, Berlin, and Madrid". In order to automatically test the plausibility of the candidate extracted facts, KnowItAll uses Pointwise Mutual Information (PMI) statistics to measure the dependence between each extracted fact and the automatically generated discriminator phrases associated with the class (such as "X is a city" for the class City). Figure 3.2 shows the flowchart of the main components in KnowItAll. The system depends on three methods to boost the system's recall and extraction rate without sacrificing precision: (1) pattern learning that learns

domain-specific extraction rules and enables additional extractions, (2) sub-class extraction that automatically determines sub-classes in order to improve recall (e.g., 'Chemist' and 'Biologist' are specified as sub-classes of 'Scientist'), and (3) list extraction that locates lists of class instances, learns a wrapper for each list, and extracts list elements using the wrapper [Etzioni *et al.*, 2005]. .

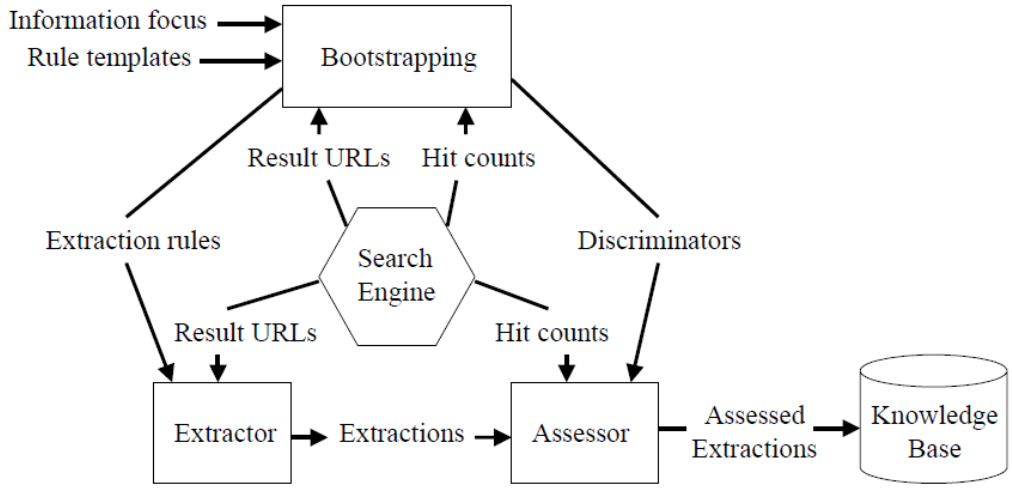


Figure 3.2: Flowchart of KnowItAll system [Etzioni *et al.*, 2005].

Nadeau *et al.* [2006] present a semi-supervised NER system that handles more than the three classical named-entity types (Person, Location, and Organisation) and is made of two modules. The first module is used to automatically create large gazetteers of entities by retrieving pages with a set of seeds and applying a web page wrapper to identify the location of specific types of information within a web page. The second module is used to disambiguate entities based on heuristics. The MUC-7 corpus is used to evaluate the system, which outperforms baseline supervised systems, but it still can not compete with more complex supervised systems.

Liao and Veeramachaneni [2009] propose a simple semi-supervised learning algorithm using Conditional Random Fields (CRF). Their algorithm improves the performance of the NER system by incorporating knowledge from unlabelled text. So, the algorithm starts with a small amount of labelled data (L) and a classifier C_k that is trained on L. A large unlabelled corpus U from the test domain is exploited by automatically and gradually

adding new training data D from U to L, provided that L has two properties: 1) L is accurately labelled, so that the labels assigned by automatic annotation of the selected unlabelled data are correct, and 2) L is not redundant, so that the new data is from regions in the feature space that the original training data set does not adequately cover. Thus, the classifier C_k is expected to get better monotonically as the training data gets updated through k iterations.

Baroni *et al.* [2010] present an algorithm that automatically identifies the most distinctive properties of each concept from naturally occurring text, which requires no supervision and minimal pre-encoded knowledge (POS tagging, lemmatisation of the corpus, and a set of extraction templates defined over POS sequences). So, given a list of concepts and a corpus, the algorithm constructs structured representations of the concepts in two phases. In the first phase, the algorithm uses pattern filtering to identify and score potential properties of the concepts. In the second phase, the algorithm generalises the strings connecting each scored concept-property pair and assigns a type to each of them. The properties are scored based on the number of distinct patterns connecting them to a concept, rather than on the number of co-occurrences in the corpus. This is based on the fact that the variety of patterns connecting a potential property and a concept is a good indicator of the presence of a true semantic link.

3.5.3 Distant Learning

Distant learning (DL) is another popular paradigm that avoids the high cost of supervision. It depends on the use of external knowledge (e.g., encyclopedias such as Wikipedia, unlabelled large corpora, or external semantic repositories) to increase the performance of the classifier, or to automatically create new resources for use in the learning process.

Kazama and Torisawa [2007] used Wikipedia to improve the accuracy of their CRF-based NE recogniser. So, they exploited the first sentence of each Wikipedia article to automatically create lists of category labels, which were used as features in the NE recogniser. In order to create these lists, they retrieved the corresponding Wikipedia entry for each candidate word sequence in the CoNLL 2003 data set. Then, they used a noun phrase

following forms of the verb "to be" to derive a label. For example, their method associated the Wikipedia entry "Apple Inc." to the category label "company" using the noun phrase following the verb "is" in the first sentence of the Wikipedia article "Apple Inc. is an American multinational technology company".

The automatic creation of training data using external knowledge has also been investigated in many studies. An *et al.* [2003] extracted sentences containing listed entities from the web and produced a 1.8 million Korean word data set. Their corpus performed as well as manually annotated training data.

Nothman *et al.* [2008] exploited English Wikipedia to create a massive corpus of named entity annotated text. They transformed Wikipedia's links into named entity annotations by classifying the target articles into standard entity types (i.e, Person, Location, and Organisation). In order to classify Wikipedia articles, they used a semi-supervised bootstrapping approach with heuristics based primarily on two features: category head nouns and definitional opening sentences of articles. The bootstrapping approach starts by extracting the features from each article. Iteratively, confident mappings from feature to NE type are inferred from classified articles. Then, the classifier is again applied to all of Wikipedia. The Wikipedia-derived corpora they automatically constructed using the aforementioned methodology tend to perform better than other cross-corpus train/test pairs in comparison with MUC, CoNLL, and BBN corpora.

Richman and Schone [2008] used a method similar to that presented by Nothman *et al.* [2008] in order to automatically create NE annotated data in six languages: French, Ukrainian, Spanish, Polish, Russian, and Portuguese. Their approach involved classifying English Wikipedia articles and using Wikipedia inter-language links to deduce the classifications in corresponding articles of other languages. With these classifications, they automatically created annotated Wikipedia articles. Then, they used the generated corpus as training sets for a variant of BBN's Identifinder in French, Polish, Portuguese, Russian, Spanish, and Ukrainian. Their NER classifier trained on a Spanish corpus built from Wikipedia articles achieved overall F-measure as high as 84.7% on gold-standard corpora, comparable to an NER classifier trained on up to 40,000 words of a gold-standard newswire

data set.

Rather than using outgoing links of Wikipedia to derive NE annotations as in the study of Richman and Schone [2008], Mika *et al.* [2008] used attribute-value pairs found in infoboxes. For example, Wikipedia article 'Apple Inc.' has an infobox which contains *Products* as an attribute and *Mac, iPad, and iPhone* as values. [attribute - value] –>[Products - Mac, iPad, and iPhone]. For each value in the infobox pairs, the system locates its instances in the Wikipedia article's text, and tags them with the corresponding attribute. They used a CoNLL-trained NER tagger to tag English Wikipedia text, which helped map infobox attributes into NE types. Then, they used this mapping to project NE types onto the labelled instances in the text, which made up the NE training data. The evaluation process revealed that their automatic training data did not exceed the gold-standard classifier on Wikipedia data, but combining their automatic and gold-standard annotations in the training phase outperformed the gold-standard classifier alone.

Nothman *et al.* [2013] developed a method that automatically created massive, multilingual training annotations for named entity recognition by exploiting the text and internal structure of Wikipedia. They first categorised Wikipedia articles into a specific set of named entity types, training and evaluating on 7,200 manually-labelled Wikipedia articles across nine languages: English, German, French, Italian, Polish, Spanish, Dutch, Portuguese, and Russian. Their cross-lingual approach achieved up to 95% accuracy. Then, they transformed Wikipedia's links into named entity annotations according to the NE type of the target articles. This technique produced reasonable annotations, but was not immediately able to compete with existing gold-standard data. They better aligned their automatic annotations to the gold standard corpus by deducing additional links and heuristically tweaking the Wikipedia corpora. Following this approach, millions of words were annotated in the aforementioned nine languages. Their method for automatically deriving corpora from Wikipedia outperformed the methods proposed by Richman and Schone [2008] and Mika *et al.* [2008] when they tested their Wikipedia-trained models on CONLL shared task data and other gold-standard corpora. When evaluated on manually annotated Wikipedia text, their Wikipedia-trained models also significantly outperformed gold-standard newswire-

trained models by 10-12% in terms of F-measure.

3.5.4 Combination of Multiple Classifiers

The combination of multiple learning algorithms (i.e., ensemble learning) is the process by which multiple models, such as classifiers or experts, are combined in order to obtain better predictive performance than could be obtained from any of the constituent models or to reduce the likelihood of an unfortunate selection of poor decisions [Opitz and Maclin, 1999; Polikar, 2007].

We are not aware of any previous work combining minimally supervised methods (e.g., semi-supervised learning, distant learning) for NER tasks in Arabic or any other natural language, but there are many studies that have examined classifier combination schemes to combine various supervised-learning systems. Florian *et al.* [2003] presented the best system at the NER CoNLL 2003 task, with an F-measure value equal to 88.76%. They used a combination of four diverse NE classifiers: the transformation-based learning classifier, a Hidden Markov Model classifier (HMM), a robust risk minimisation classifier based on a regularised winnow method [Zhang *et al.*, 2002], and an ME classifier. The features they used included tokens, POS and chunk tags, affixes, gazetteers, and the output of two other NE classifiers trained on richer data sets. Their methods for combining the results of the four NE classifiers improved the overall performance by 17-21% when compared with the best performing classifier.

Benajiba *et al.* [2008b] examined the impact of using different sets of features for each NE type instead of simultaneously using a single set of features for all NE types. So, they created a classifier for each NE type that was independently based on an optimal feature set, and then combined the different classifiers for a global NER system. An incremental feature selection method was used to select an optimised feature set and to observe the errors. In order to create a classifier for each NE type, they adopted two discriminative approaches: Support Vector Machines (SVMs) and Conditional Random Fields (CRFs). They investigated many sets of features for each NE type in isolation first, then they ranked the features according to the best performance of the two classifiers for each NE

type. In total, 16 features representing the lexical, contextual, morphological, gazetteer, and shallow syntactic features were examined. Finally, the classifications of all individual classifiers (one classifier per NE type) are combined. In case of conflicts, where the same token was annotated as two different NE types, they used a simple heuristic based on the classifier precision for that specific tag, favoring the tag with the highest precision. The evaluation process was conducted using the ACE data sets. They concluded that each feature plays a role in recognising the NE to different degrees and each NE type is sensitive to different features. The best system's overall performance in terms of F-measure was 83.5%, 76.7%, and 81.31% for ACE 2003, ACE 2004, and AC 2005 respectively. The study also reported that it can not be concluded whether CRFs is better than SVMs or vice versa.

Saha and Ekbal [2013] studied classifier combination techniques for various NER models under single and multi-objective optimisation frameworks. They used seven diverse classifiers - naïve Bayes, decision tree, memory based learner, HMM, ME, CRFs, and SVMs - to build a number of voting models based on identified text features that are selected mostly without domain knowledge. The combination methods used were binary and real vote-based ensembles. They reported that the proposed multi-objective optimisation classifier ensemble with real voting outperforms the individual classifiers, and the corresponding single-objective classifier ensemble.

3.6 Summary

In this chapter we presented an overview of the essential aspects of the NER research such as the annotation schemes and the standard evaluation metrics. We also covered the NE classes and hierarchies defined in many IE competitions. We then discussed the different techniques that have been used in NER systems and the challenges they constantly face. In addition, we discussed the direction of the current research of which the main effort is to overcome the disadvantages of traditional techniques and reduce annotation labor.

Chapter 4

Arabic Named Entity Recognition

The previous chapter presented the essential aspects of NER such as the standard definition, annotation schemes, formal evaluation metrics, and NE classes that were proposed in the research and key evaluation campaigns. We also presented the state-of-the-art in NER, the current challenges, the new approaches and the possible directions of future research.

This chapter describes in detail the approaches used in Arabic NER, including the recent research. The characteristics of Arabic that make Arabic NER a challenge are presented with examples in Section 4.1. The types of NEs and the domains covered by the available linguistic resources are explained in Section 4.2. The basic computational tasks that are considered essential preprocessing steps for most Arabic NER in the literature are described in detail with lists of their commonly-used tools in Section 4.3 and Section 4.4. Section 4.5 shows the approaches used in Arabic NER research.

4.1 Challenges for Arabic NER

Arabic poses many challenges to NLP that originate in the language's complex nature (see Chapter 2). The following section summarises the difficulties that may arise during Arabic NLP tasks in general and Arabic NER in particular:

- Capitalisation is not used in Arabic, which makes determining proper nouns in the text difficult. Benajiba *et al.* [2007] assert in their study that the main obstacle to obtaining high performance in Arabic NER is the absence of capital letters. Although exploiting English translation of Arabic words might be useful in some cases [Farber

et al., 2008], it is not a complete alternative to capitalisation because it depends on the translation process.

- Arabic is characterised by a rich morphology with a large set of features (see Chapter 2). This complex morphology causes what is called data sparseness or data insufficiency, which means that the ratio of vocabulary size to the total number of words is very high because multiple affixes attach to the same word. For example, the five rows in Table 4.1 refer to one word but appear to be five different words because of the different clitics attached to the word. As a result, Arabic words and contexts may appear in different forms, further complicating the classification of NEs. To cover all different word forms and contexts, a huge training corpus is required to achieve a reasonable performance [Benajiba and Rosso, 2008].

Arabic Word	**English Translation**
لندن	London
للندن	to London
فلندن	then London
ولندن	and London
بلندن	in London

Table 4.1: Data sparseness in Arabic due to clitics.

- Short vowels in the form of diacritical marks are required in Arabic for pronunciation and disambiguation. However, modern Arabic texts are usually written without diacritical marks. This absence of diacritical marks causes the structural and lexical ambiguity of some words. This is attributable to the fact that a word form in Arabic may have two or more meanings depending on the context in which they appear [Oudah and Shaalan, 2012]. For example, when the diacritical marks are absent, the word لما may refer to the person's name لَما *lamaa* "Lama", or to the word لِما *limaa* "Why", or to the word لَمّا *lammaa* "When". Therefore, complex processing is required to analyse the text syntactically.

- Some Arabic names are ambiguous out of context since most names in Arabic have an

interpretation as a common noun, adjective, or even verb as shown in Table 4.2. This characteristic of the Arabic language along with the absence of capitalisation make the list lookup strategies for detecting Arabic NEs less effective and more complex because ambiguity should be removed not only between different types of NEs but also between NEs and other regular words in the text.

Arabic Name	English Translation	Examples	English Translation	POS Categories
سعيد/ Sayed	happy	الولد سعيد	The boy is happy.	Adjective
		ذهب سعيد إلى الجامعة	Sa`yd (happy) went to the University.	Person Name
أمل/ Amal	hope	الأمل هو الحياة	The hope is the life.	Noun
		زارت أمل صديقتها	Amal (hope) visited her friend.	Person Name
صالح/ Saleh	righteous	هذا رجل صالح	He is a righteous man.	Adjective
		يقرأ صالح الكتاب	Saleh (righteous) is reading a book.	Person Name

Table 4.2: Examples of personal Arabic names that can be nouns or adjectives.

- There are some Arabic letters that are commonly misspelled and appear in many variant forms. For instance, in some Arab countries, the letter ي *yaa'* is often written without dots. This changes the meaning and makes *yaa'* look like the letter ى *alif maqSuwrah* [Habash, 2010]. On the other hand, ى *alif maqSuwrah* might also be written mistakenly as a bare *alif*. For Example, the word على without dots is the preposition "on" while علي with dots is the person's name "Ali".

- The orthography of Modern Standard Arabic (MSA) has been standardised for a long time now [Habash, 2010]. However, some differences persist across Arab writers. For example, there are two common spellings of the proper nouns of some geographic entities ending with an /a/ vowel such as Syria and Africa. Standard *hamzah* spelling rules in Arabic also have some exceptions. In some Arab countries, for example, the *hamzah* in the word مسؤول "responsible" is commonly written above a dotless

yaa' مسئول. In addition, the spelling of loanwords may likely vary according to the differences in vowel spelling. Some people prefer to use short vowels when writing loanwords in the Arabic script, while others prefer to use long vowels. For example, فيلم *fylm* "Film" might also be written: فلم *film* [Habash, 2010]. These inconsistencies increase the problem of data insufficiency, especially if the data set is not large enough in the training phase. Many examples can be shown in Table 4.3.

English Word	Arabic spelling
Syria	سوريا
	سورية
Africa	افريقيا
	افريقية
Film	فـيلـم
	فـلـم
Studio	أستوديو
	أستديو

Table 4.3: Examples of Arabic orthography variations.

4.2 Available Linguistic Resources: Textual Genre and NE types

Large annotated corpora and gazetteers are essential when implementing and evaluating Arabic NER systems [Abdul-Hamid and Darwish, 2010; Benajiba *et al.*, 2007; Darwish, 2013]. Unfortunately, it is not easy to access comprehensive and updated lists of freely available Arabic corpora, as they are scattered in various personal and research groups sites [Zaghouani, 2014]. Language data providers, such as the Linguistic Data Consortium (LDC) and the Evaluations and Language Resources Distribution Agency (ELDA), provide

updated and comprehensive lists of corpora, but they are not always free and sometimes require calendar-year membership to access to the corpora released in specific years.

Publicly available corpora are preferable in order to make comparisons between NER systems easier and rank them according to their annotation capability. One of the earliest publicly available corpora for Arabic NER is ACE data sets, which are developed and annotated by the Linguistic Data Consortium (LDC) as part of the Automatic Content Extraction (ACE) program. The ACE data released in 2003 includes Broadcast News (BN) and Newswire (NW) genres and contains around 55K tokens. The ACE 2004 corpus includes BN and NW from Arabic Treebank (ATB) genres and contains around 155K. An additional genre, WebLogs (WL), was added in ACE 2005 which contains around 113K tokens. The ACE corpora are available under paid license agreements, which makes accessing and utilising them difficult for the researcher [Benajiba *et al.*, 2009; Doddington *et al.*, 2004].

ANERcorp[1] is the first freely available Arabic NE annotated corpora that is considered a standard data set to evaluate and compare different NER systems in the literature [Abdallah *et al.*, 2012; AbdelRahman *et al.*, 2010; Benajiba *et al.*, 2008b; Darwish, 2013; Oudah and Shaalan, 2012]. A collection of three gazetteers is also released with ANERcorp, containing names lists of Person (2,309 names), Location (1,950 names), and Organisation (262 names). ANERcorp is derived from the newswire domain and covers the CoNLL four classes: Person, Location, Organisation and Miscellaneous. ANERcorp contains approximately 150K tokens (11% of the tokens are NEs) and follows the framework used to build the training and test data sets in the CONLL 2002 conference. So, each token of the data set is tagged as belonging to one of the following classes: *B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, B-MISC,* and *I-MISC* where B indicates the Beginning of the entity name and I indicates the continuation (Inside) the entity name. *PER, LOC, ORG* indicate Person, Location and Organisation name respectively. The name of an entity that does not belong to any of the three previous mentioned classes will be given the tag *MISC*. The ANERcorp data set is composed of ANERcorp training set (80% of the data set), and ANERcorp test set (20%

[1]ANERcorp is available at `http://www1.ccls.columbia.edu/~ybenajiba/downloads.html`.

of the data set) [Benajiba *et al.*, 2007].

Another free NER corpus available online is a set of Arabic Wikipedia articles manually annotated for named entities by Mohit *et al.* [2012] as part of AQMAR (American and Qatari Modelling of Arabic) project. The AQMAR[2] corpus contains 74k tokens, derived from 28 Arabic Wikipedia articles and representing four domains: history, science, technology, and sports. Each article was tagged by 1 of 2 annotators, who were encouraged to think of up to 3 article-specific entity classes to be added to the standard four (Person, Organisation, Location, and generic Miscellaneous). For example, the Wikipedia article *Computer_Software* in AQMAR corpus has, in addition to the standard four classes (PER, ORG, LOC, and MIS-0), three more classes (added by annotators): MIS-1 for English entities, MIS-2 for name of software (e.g. Microsoft Word), and MIS-3 for name of computer component (e.g. CPU).

A recent survey for identifying free Arabic language resources reported only two NE annotated corpora that are available for free, ANERcorp and AQMAR, and four different sets of gazetteers: JRC-Names[3], Named Entity Translation Lexicon (NETLexicon)[4], Arabic Named Entities List (ArabicNEs)[5], and ANERgazet[6] [Zaghouani, 2014].

The fine-grained NE corpora for Arabic most recently released are $\text{WikiFANE}_{\text{Selective}}$ and $\text{WikiFANE}_{\text{Whole}}$[7]. The two corpora are automatically collected from the Arabic version of Wikipedia by Alotaibi and Lee [2013]. The ACE (2008) taxonomy of named entities is used for WikiFANE corpora with some modifications. The modifications include adding one new class "Product" and 9 fined-grained classes to the Person class: Politician, Athlete, Businessperson, Artist, Scientist, Police, Religious, Engineer, and Other. $\text{WikiFANE}_{\text{Whole}}$ contains all sentences retrieved from Wikipedi articles, composing of 2,023,496 tokens. On the other hand, $\text{WikiFANE}_{\text{Selective}}$ is compiled by selecting only the sentences that have at least one named entity phrase. The total number of tokens for the compiled corpus is 2,021,177.

[2]AQMAR is available at `http://www.ark.cs.cmu.edu/ArabicNER/`.
[3]JRC-Names is available at `https://ec.europa.eu/jrc/en/language-technologies/jrc-names`.
[4]NETLexicon is available at `http://nlp.qatar.cmu.edu/resources/NETLexicon/`.
[5]ArabicNEs is available at `https://sourceforge.net/projects/arabicnes/`.
[6]ANERgazet is available at `http://www1.ccls.columbia.edu/~ybenajiba/downloads.html`.
[7]WikiFANE data sets are available at `http://www.arabic-ner.com/`.

Another two fine-grained Arabic NE corpora, NewsFANE$_{\text{Gold}}$ and WikiFANE$_{\text{Gold}}$, were manually annotated and released by Alotaibi and Lee [2014]. The NewsFANE$_{\text{Gold}}$ is a newswire corpus that uses the same textual data appearing in ANERcorp, but has been re-annotated to the fine-grained level. The WikiFANE$_{\text{Gold}}$ is drawn from the Arabic version of Wikipedia and contains 500K tokens. Both of these corpora use the same two-level taxonomy applied to WikiFANE$_{\text{Selective}}$ and WikiFANE$_{\text{Whole}}$.

4.3 Basic Computational Tasks

The rich morphology of Arabic causes a set of challenges that should be handled before developing an NLP system. The following presents the basic computational tasks that are required to prepare Arabic text in order to perform NER.

4.3.1 Tokenisation

The standard preprocessing step for many NLP tasks is tokenisation, which divides a string of written language into its component tokens [Jurafsky and Martin, 2009]. Tokens are usually separated by whitespace characters (e.g., a space or line break) or by punctuation marks. For example, tokenisation in English usually involves splitting punctuation marks, and some affixes like possessives [Chang *et al.*, 2008; Jurafsky and Martin, 2009; Manning and Schütze, 1999]. On the other hand, morphologically rich languages, like Arabic, require a more extensive tokenisation process to separate different types of clitics from the word. This complex tokenisation is usually called word segmentation [Green and DeNero, 2012; Monroe *et al.*, 2014]. Simple tokenisation, which splits text into tokens based on whitespace and punctuation marks, is usually the first step to produce other word segmentation schemes, as explained in detail in Section 4.3.2. Table 4.4 illustrates how simple tokenisation works on a text.

Input in English: And the 2016 UEFA European Football Championship will be held in France.		
	Qalam Transliteration	**Arabic Text**
Input	Wstgaam bTwlat alywrw likrt alqdm2016 fy farnsaa.	وستقام بطولة اليورو لكرة القدم2016 في فرنسا.
Simple okenization	Wstgaam bTwlat alywrw likrt alqdm 2016 fy farnsaa .	وستقام بطولة اليورو لكرة القدم 2016 في فرنسا .

Table 4.4: An example of a simple tokenisation scheme.

4.3.2 Word Segmentation

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 analysis of the text. These are great benefits for many NLP tasks such as POS tagging. A large number of possible segmentation levels can be applied to Arabic text, according to the types of clitics to be split. The optimal segmentation scheme depends entirely on the task and the implementation of the task itself. So, what is appropriate for NER is not necessarily appropriate for Machine Translation (MT) [Habash, 2010]. The following is a list of some commonly used segmentation schemes:

- Decliticisation scheme (D1,D2,D3, and WA): This scheme has three different degrees and is responsible for splitting off clitics. It was introduced by [Habash and Sadat, 2006]. The first degree, D1, splits off the conjunction clitics that precede the word (w+...و ,and f+...فـ) and the infrequent interrogative clitic. D2 does the same as D1, and also splits off particles (l+...لـ, k+...كـ, b+...بـ, and s+...سـ). D3 performs the tasks of D1 and D2, in addition to splitting the definite article (al+ الـ) and all pronominal enclitics (e.g., ك...+ak, ه...+hu). WA is a special case of D1 in which the segmenter splits off only the conjunction clitic (w+...و).

- Penn Arabic Treebank segmentation (TB): This scheme is similar to D3, but does not split off the definite article (al+ الـ). TB is the scheme that was used in the Arabic

Treebank [Maamouri *et al.*, 2004].

- Decliticisation scheme (S2): This scheme was introduced by [Badr *et al.*, 2008]. S2 is basically similar to D3, except that the split clitics are agglutinated into one prefix and one suffix, such that any given word is split into three parts: prefix+ stem +suffix.

- Morphemes segmentation scheme (MR): It splits off the word into its stem and affixed morphemes.

- Lemmas segmentation scheme (LEM): This scheme converts every word to its lemma. LEM can be used with other segmentation schemes [Habash, 2010].

- English-like segmentation (ENX): It performs decliticisation similarly to D3, but uses Lemmas and POS tags instead of the regenerated words. The POS tag set is the Bies reduced Arabic Treebank tag set [Diab *et al.*, 2004; Maamouri *et al.*, 2004].

Table 4.5 provides examples that clarify the differences between the segmentation schemes mentioned above:

4.3.3 POS Tagging

Part-of-speech (POS) tagging, also called word-category disambiguation, is the process of assigning appropriate morpho-syntactic tags to every word in a sentence based on the context. The tags are chosen from a comprehensive, well-defined tag set [Habash, 2010]. In the traditional Arabic POS classification, Arabic words are grammatically analysed into three main parts-of-speech: verb, noun, and particle [Khoja, 2001]. These three parts-of-speech can be further sub-categorised into more detailed parts-of-speech that collectively encompass the whole Arabic language [Haywood and Nahmad, 1962]. The POS tag set for Arabic can be very large since Arabic has a rich morphology where the words consist of several segments including inflectional affixes, stems, and clitics. However, there are a variety of Arabic POS tag sets in which some tag sets are complete and cover all morphological information, while other tag sets ignore some or all morphological features. In addition, The tag set size differs depending on whether the text is segmented into morphemes or not

Input in English: and this will include the tax that would be imposed on his company.		
	Qalam Transliteration	**Arabic Text**
Input	wsyshml dhlk alZraayb alty stufrD `lae shrkth .	وسيشمل ذلك الضرائب التي ستفرض على شركته .
D1	w syshml dhlk alZraayb alty stufrD `lae shrkth .	و+ سيشمل ذلك الضرائب التي ستفرض على شركته .
WA	wsyshml dhlk alZraayb alty stufrD `lae shrkth .	وسيشمل ذلك الضرائب التي ستفرض على شركته .
D2	W+ s+ yshml dhlk alZraayb alty s+ tufrD `lae shrkth .	و+ س+ يشمل ذلك الضرائب التي س+ تفرض على شركته .
D3	W+ s+ yshml dhlk al+ Zraayb alty s+ tufrD `lae shrkt h .	و+ س+ يشمل ذلك ال+ ضرائب التي س+ تفرض على شركة +ـه .
S2	Ws+ yshml dhlk al+ Zraayb alty s+ tufrD `lae shrkt +h .	و س+ يشمل ذلك ال+ ضرائب التي س+ تفرض على شركة +ـه .
TB	W+ s+ yshml dhlk alZraayb alty s+ tufrD `lae shrkt h .	و+ س+ يشمل ذلك الضرائب التي س+ تفرض على شركة +ـه .
MR	W+ s+ y shml dhlk al+ Zraayb alty s+ t+ ufrD `lae shrkt +h .	و+ س+ ي+ شمل ذلك ال+ ضرائب التي س+ ت+ فرض على شركة +ـه .
LEM	shml dhlk Zraayb alty frD `lae shrkt .	شمل ذلك ضرائب التي فرض على شركة .
ENX	w+ s+ shml$_{VBP}$ dhlk$_{DT}$ al+ Zraayb$_{NN}$ alty$_{WP}$ s+ frD$_{VBP}$ `lae$_{IN}$ shrkt$_{NN}$ +h .	و+ س+ شمل ذلك ال+ ضرائب التي س+ فرض على شركة +ـه . (DT VBP NN WP NN IN VBP NN)

Table 4.5: Examples of different segmentation levels.

(and on which segementation scheme is used). So, the POS tag of an unsegmented word is constructed by concatenating the POS tags of its tokens. For example, "his house" is written as one word in Arabic, بيته *baythu*. Thus, on one hand, the POS-tag of the word without segmentation (also known as clitic tokenisation) is *NOUN+PRON_3MS*. On the other hand, the word after segmentation consists of two tokens: بيت and ه so that the first token can be tagged as *NOUN* and the second token can be tagged as *PRON_3MS*. Therefore, the possessive pronoun is part of the word tag in the first case, while it is a tag of its own in the second case. Although there are many different sizes of tag sets, there is no optimal POS tag set. Every applications has different requirements. That is, the larger sets can help better in higher order tasks (under gold/oracle conditions) [Habash, 2010],

but they tend to be hard to predict accurately [Marton *et al.*, 2010]. The smaller tag sets, on the other hand, can be predicted accurately, and they are proved to be useful for some tasks such as Base Phrase Chunking [Diab, 2007a,b]. The following explains some of the Arabic POS tag sets with different degrees of granularity and sizes.

4.3.3.1 Buckwalter Tag Set

The Buckwalter tag set is a form-based tag set that can be used for tokenised and untokenised text [Habash, 2010]. The untokenised tag set is used by the Buckwalter Arabic Morphological Analyser (BAMA), which assigns complex tags to untokenised/unsegmented words. For example, the tag for the Arabic word وسيعلمون *wsy'lmwn* "and they will know" is *CONJ+FUT+IV3MS+IV+IVSUFF_SUBJ:MP_MOOD:I*. The Buckwalter tokenised tag set is utilised in the Penn Arabic Treebank (PATB) to assigns tags to tokenised words. For example, the previously mentioned word وسيعلمون *wsy'lmwn* "and they will know" can be segmented to و س ي علم ون *w s y 'lm wn* and then annotated by the following tags: *w/CONJ*, *s/FUT*, *y/IV3MS*, *'lm/IV*, *wn/IVSUFF_SUBJ:MP_MOOD:I*. Both Buckwalter tokenised and untokenised tag sets use the same basic 70 tags such as *CONJ* for conjunction, *IV* for imperfective verb, and *IV<PGN>* for imperfective verb prefix. These 70 basic tags are combined to form morpheme tags, which are concatenated to form the word tags. Buckwalter tokenised and untokenised tags differ only in the number of basic tags that they can combine. The *CONJ* tag, from the above examples, is used as a tag of its own in the Buckwalter tokenised tag set. In contrast, The *CONJ* tag is used in the token tags which constitute the word tag in the Buckwalter untokenised tag set. The tokenised Buckwalter tag set is around 500 tags or so, while the untokenised Buckwalter tag set might reach thousands of tags [Bies and Maamouri, 2003]. Several variants of Buckwalter tags are used in different versions of the PATB as well as the BAMA analyser [Habash, 2010].

4.3.3.2 The Reduced Tag Set (RTS)

The Reduced Tag Set (RTS) is also referred to as the Bies tag set and as the PennPOS tag set [Diab *et al.*, 2004]. Developers of the RTS Ann Bies and Dan Bikel were inspired by

the Penn English Treebank POS tag set to reduce the Buckwalter tag set to a manageable size [Marcus *et al.*, 1993]. The RTS ignores many distinctions in Arabic. For example, *JJ* is used for all adjectives regardless of their inflections [Habash, 2010]. In addition, although Arabic has a rich variety of demonstrative pronouns, as can be seen in Table 4.6, all demonstrative pronouns are considered *DT* in the RTS.

Demonstrative of Proximity						
		Masculine		**Feminine**		**English Translation**
Singular		هذا	hadhaa	هذه	hadhihi	This
Dual	Nominative	هذان	hadhaani	هاتان	haataani	These
	Genitive/accusative	هذين	hadhyni	هاتين	haatyni	These
Plural		هؤلاء	ha'wlaa'i	هؤلاء	ha'wlaa'i	These
Demonstrative of Distance						
		Masculine		**Feminine**		**English Translation**
Singular		ذلك	dhalika	تلك	tilka	That
Plural		أولئك	'uwla'ika	أولئك	'uwla'ika	Those

Table 4.6: Arabic demonstrative pronouns.

The Bies tag set is used widely for Arabic POS tagging [AlGahtani *et al.*, 2009; Diab *et al.*, 2004; Habash and Rambow, 2005; Kulick *et al.*, 2006] although it is considered to be linguistically rough [Habash, 2010]. Appendix A illustrates the list of 24 tags that compose the Reduced Tag set.

4.3.3.3 The Extended Reduced Tag Set (ERTS)

The ERTS is a subset of the Buckwalter tokenised tag set and a superset of the RTS tag set [Habash, 2010]. It encodes all information contained in the RTS tag set as well as additional morphological features such as number, gender, and definiteness. ERTS consists of 72 tags instead of 24 tags in standard RTS [Diab, 2009]. For example, nouns in RTS might be represented as *NN* or *NNS* indicating only number, whereas ERTS nouns can be expressed using definiteness (the presence of the definite article), gender, and number such as *DNN, DNNS, NNF,* and *NNMDu.* In ERTS, *D* represents the presence of the definite article.

F and *M*, defining feminine and masculine respectively, represents the gender. Number is represented by *Du* for dual, *S* for plurals, and the absence of a tag for singular [Diab, 2007a].

4.3.4 Stemming

Stemming is the process of reducing derived or inflected words to their stems or original roots. Algorithms of stemming are commonly referred to as stemmers [Goldsmith *et al.*, 2001]. Research shows that Arabic stemming is a specifically challenging task because of its highly inflected and derivational nature [Aljlayl and Frieder, 2002; Larkey *et al.*, 2002, 2007; Nwesri *et al.*, 2005].

The work on stemming can be divided into two main types according to the aims of the stemming process. Some work tries to reduce words to their original roots (root-extraction stemmers), and 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]. Therefore, researchers can decide which type of stemmers to use in their systems depending on the applications requirements and the experimental results. The following explains the root-extraction stemmers and light stemmers in detail with their advantages and disadvantages.

4.3.4.1 Root-extraction Stemmers

Root-extraction stemmers, which are also known as aggressive or strong stemmers, aim to reduce words to their original roots. Converting a word to its root may result in the mapping of many related words to that single root, although each one of these terms has a unique meaning. This makes the root-extraction stemmer a poor candidate for applications that require the highly accurate matching of similar words [El-Beltagy and Rafea, 2011; Goweder *et al.*, 2004].

One of the earliest Arabic root-extraction stemmers is the Khoja's stemmer developed by Khoja and Garside [1999] from Lancaster University. This stemmer requires root dictio-

nary and pattern-matching to find the roots of the words after removing affixes using the following procedures:

1. Remove diacritical marks used for vowelisation.
2. Remove numbers, punctuation marks, and stop words.
3. Remove the definite article (ال, the).
4. Remove the inseparable conjunction (و, and).
5. Remove suffixes.
6. Remove prefixes.
7. Match the resulting word against a list of patterns. If a match is found, extract the characters in the pattern representing the root.
8. Match the extracted root against a list of "valid" roots.
9. Replace weak letters (ا و ي) with (و).
10. Replace all letters containing *hamzah* with (ا).
11. Examine two-letter roots to see if they should contain a double character. If so, add this character to the root [Taghva *et al.*, 2005].

The Khoja's stemmer is widely used and is now the standard algorithm for root stemming in Arabic [Sawalha and Atwell, 2008].

4.3.4.2 Light Stemmers

The light stemmer is another type of Arabic stemmer, which reduces words to their stems instead of their original roots. So, the light stemmer targets a specific subset of prefixes and suffixes to be removed [Al Ameed *et al.*, 2005; Darwish, 2002; Larkey *et al.*, 2007]. Although a light stemmer can correctly conflate many forms of the same word into one large group, it can fail to conflate other variants that should go together. For example,

when using a light stemmer, past tense verbs do not get conflated with their present tense forms, because they retain some affixes and internal differences [Larkey *et al.*, 2007].

Larkey *et al.* [2002, 2007] developed a number of light stemmers with small differences between them. These light stemmers were evaluated on the Text Retrieval Conference (TREC) data set and the results revealed that light10 stemmer was the best. The following steps show the procedure followed by light10 stemmer to produce the stem of an input word:

1. Remove punctuation, diacritical marks, and non-letters.
2. Replace (أ، إ، آ) with (ا).
3. Replace final (ى) with (ي).
4. Replace final (ة) with (ه).
5. Remove the conjunction (و, and) provided that the the remainder of the word is 3 or more characters long.
6. Remove the definite articles (ال، وَال، بَال، كَال، فَال، لل) provided that the remainder of the word is at least two characters long.
7. Remove the suffixes (هَا، ان، ون، ين، يه، ية، ه، ة، ي) provided that the the remainder of the word is at least two characters long.

In step 5, Larkey *et al.* [2007] set the length restriction to overcome the problem of removing the letter و *waaw*, which constitutes part of the word, instead of removing the conjunction و *wa*. However, this restriction on length still does not warrant that this prefix will not be removed when it should not be.

4.3.5 Arabic Normalisation

A large number of Arabic NLP tasks require the text be free of punctuation or diacritical marks like 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 including NER, another issue that should be addressed in raw Arabic text is inconsistent variations. For example, different forms of *alif* (إ، أ، آ، ا) might be written interchangeably; another example is *alif maqSuwrah* and the regular dotted *yaa’* (ي، ى) which are usually used interchangeably at the final position of the word. The same is true for *taa’ marbuwTah* and *haa’* (ة، ه). These misspelling errors in Arabic affect 11% of all words in the Penn Arabic Treebank (PATB) Habash [2010]. The Arabic normaliser should provide a different level of orthographic normalisation that can be carried out on Arabic text to reduce noise and data sparsity. This includes the normalisation of different letters that can be written interchangeably.

4.4 Tools for Processing Arabic Text

In the following we present Arabic tools that are used extensively in the literature [Habash, 2010] in order to preprocess the Arabic text when developing NER systems.

4.4.1 AMIRA

AMIRA[8] is a statistical toolkit for the computational processing of Arabic morphology. It is built as a successor to the Asvmt toolkit developed at Stanford University [Diab *et al.*, 2004; Soudi *et al.*, 2007]. AMIRA uses Support Vector Machines in a sequence modelling framework. It is based on supervised learning and does not depend explicitly on knowledge of deep morphology. The toolkit contains a tokeniser (AMIRA-ToK) and a POS tagger (AMIRA-POS). AMIRA-TOK focuses mainly on clitic tokenisation. So, AMIRA-TOK segments off the following set of clitics:

- Conjunction proclitics (w+...و ,and f+...ف).
- Prepositional proclitics (l+...ل, k+...ك, and b+...ب).
- Future marker proclitic (s+...س).

[8]AMIRA ia available at `https://www.flintbox.com/public/project/8335/`.

- Verbal particle proclitic (l+...لـ).
- Definite article (al+ الـ).
- Pronominal enclitics indicating possessive /object pronouns.

More details about segmentation schemes can be found in Section 4.3.1.

AMIRA-POS primarily uses the ERTS POS tag set, but it offers an option to tag with ERTS or RTS tag set (see Section 4.3.3). AMIRA-POS also assumes that the text is decliticised (clitic tokenised). The user has the flexibility to input raw or clitic tokenised text in a scheme that is consistent with one of the schemes supported by AMIRA-TOK [Habash, 2010]. If the input is raw text, AMIRA-POS runs AMIRA-TOK on the input and then performs POS tagging. The user, however, can request that the POS tags be assigned to the surface words. In this case, the ERTS tag set is appended with clitic POS tags to form more complex POS tags.

4.4.2 BAMA

BAMA[9] stands for Bukwalter Arabic Morphological Analyser. It contains three Arabic-English lexicon files: prefixes (299 entries), suffixes (618 entries), and stems (82,158 entries). The lexicons are supported by three morphological compatibility tables used for controlling prefix-stem combinations (1,648 entries), prefix-suffix combinations (598 entries), and stem-suffix combinations (1,285 entries). Prefix and suffix lexicon entries cover all possible concatenations of Arabic prefixes and suffixes respectively. The following information is specified for each lexicon entry: the morphological compatibility category, an English gloss, and POS data. Stem lexicon entries are gathered around their specific lexeme, which is not used in the analysis process. Compatibility tables define which morphological categories are allowed to co-occur [Habash, 2010]. The analysis algorithm is simple because morphotactics and orthographic rules are built directly into the dictionary itself instead of being specified in terms of general rules that interact to realise the output [Buckwalter, 2004]. So, Arabic words are segmented into all possible sets of prefix, suffix, and stem strings.

[9]BAMA 1.0/1.2 are both publicly available at `https://catalog.ldc.upenn.edu/LDC2002L49`. BAMA 2.0 is available through the Linguistic Data Consortium at `https://catalog.ldc.upenn.edu/LDC2004L02`.

The valid segmentation is specified when the three strings exist in the lexicons and are three-way compatible (prefix-stem, prefix-suffix, and stem-suffix)[Buckwalter, 2002].

4.4.3 MADA+TOKAN

The combined package (MADA+TOKEN)[10] is built on top of BAMA as a natural successor in order to meet the growing requirements of many Arabic NLP applications [Habash *et al.*, 2009]. The package consists of two components. The MADA component, which stands for Morphological Analysis and Disambiguation for Arabic, is a utility that takes raw Arabic text and provides a solution to many basic problems in Arabic NLP, including diacritisation (insertion of short-vowel diacritical marks), morphological disambiguation (determining the full morphological information for each word given its context), POS tagging (determining the specific grammatical category of each word), and stemming (reducing each word to its stem) [Habash and Rambow, 2005]. The TOKAN component allows the user to specify any decliticisation scheme that can be generated from disambiguated analyses. [Habash, 2010].

4.5 Approaches to Arabic NER

4.5.1 Rule-based Approach

The rule-based approach, which is also known as the knowledge engineering approach, relies on regular expressions and heuristic rules to identify NEs. This section presents some of the studies that make use of the rule-based approach to Arabic NER in the literature.

Abuleil and Evens [2004] developed a rule-based NER system that depends on a set of rules, trigger words (i.e., NE indicators that help identify NEs within text), and some special verbs. The research is based on the intuition that an NE appears close to one of these trigger words in Arabic text. The research assumes that the NE should not be more than three words away from the trigger word or the special verb. The research also assumes that the longest NE is 7 words. Therefore, their system marks 10 words to the left of the trigger word/special verb and 10 words to the right of them to identify and mark the name

[10]MADA+TOKEN package is available at `https://flintbox.com/public/project/8348`.

phrases. After marking the name phrases that can include NEs, their technique searches in the lexicon for the part-of-speech tags for each word (particle, verb, noun, adjective) in the name phrase. Then, graphs are used to represent the words in these phrases and the relationships between them. Finally, some rules are used to generate and classify NEs before saving them in a database. Their system was tested on 500 articles from the Al-Raya newspaper (2003). It managed to extract 78.4% of the NEs found in the text and obtained a precision of 90.4% for Person, 93.0% for Location, and 92.3% for Organisation.

Shaalan and Raza [2009] adopted a rule-based approach to develop an NER system for Arabic (NERA). The system depends on two types of resources: a gazetteer representing a dictionary of names, and grammar in the form of regular expressions. The NERA system also has a filtration mechanism that is used in order to exclude invalid NEs and to disambiguate identical or overlapping textual matches returned by different named entity extractors. Rejecting invalid NEs is based on a blacklist (rejecter) dictionary that is built by analysing the local lexical context of named entities during grammar rule formulation. Manually constructed corpora from ACE, and the Web are used in the evaluation process. The NERA system achieved an F-measure of 87.7%, 85.9%, and 83.15% for Person, Location, and Organisation respectively.

Elsebai *et al.* [2009] developed a system for extracting Arabic persons' names. The system uses two main components: the General Architecture for Text Engineering (GATE) environment [Cunningham *et al.*, 2011] and the the Buckwalter Arabic Morphological Analyser (BAMA). A set of trigger words is used in the development of the heuristics in order to indicate the phrases that probably include persons' names. The BAMA is also used to formulate the heuristics by assigning a POS-tag to each word. Elsebai *et al.* [2009] evaluated their NER system using around 700 news articles extracted from the Aljazeera television web site. Their system achieved an F-measure of 89% where precision is 93% and the recall is 86%.

4.5.2 Machine Learning Approach

This section explains the machine learning algorithms that are used for Arabic NER in the literature.

One of the earliest systems to exploit supervised learning techniques for Arabic NER is ANERsys 1.0, developed by Benajiba *et al.* [2007] based-on Maximum Entropy (ME). So, Benajiba *et al.* [2007] trained the ME model using a set of language-independent features. They used their own gazetteer (ANERgazet). The trained ME model was evaluated on ANERcorp test set (see Section 4.2 for detailed information about the data set) and achieved an F-measure equal to 55.23%. The system outperformed the baseline by 12 percentage points without using any POS-tag information or text segmentation. The baseline model tagged each word by assigning the class that was most frequently assigned to the word in the training corpus. The ANERsys 1.0 system had difficulties detecting NEs that contain more than one token (multi-word NEs).

The ANERsys 2.0, developed by Benajiba and Rosso [2007], is an enhanced version of the aforementioned system (ANERsys 1.0). The ANERsys 2.0 uses part-of-speech tags to improve NE boundary detection. It also adopts a 2-step architecture for better multi-word NE recognition. The first step of ANERsys 2.0 concerns only the delimitation of NEs. The delimitation of the boundaries of NEs is conducted by an ME-based module and a POS-tag-based module. The results of these two modules are combined into a third module, as illustrated in Figure 4.1. The second step is to classify each of the NEs delimited in the previous step. According to the evaluation results using the ANERcorp test set, the ANERsys 2.0 leads to an increase in performance of 10 percentage points over the first version where the overall F-measure = 65.91%.

Benajiba and Rosso [2008] employed Conditional Random Fields (CRFs) in order to improve the performance of the ANERsys 1.0 and ANERsys 2.0 systems. They trained a CRF-based classifier with different sets of language-independent and Arabic-specific features such as the part-of-speech tags, base phrase chunks, gazetteer, and nationality. They also studied the impact of tokenising the data. The reported results of evaluating the CRF-

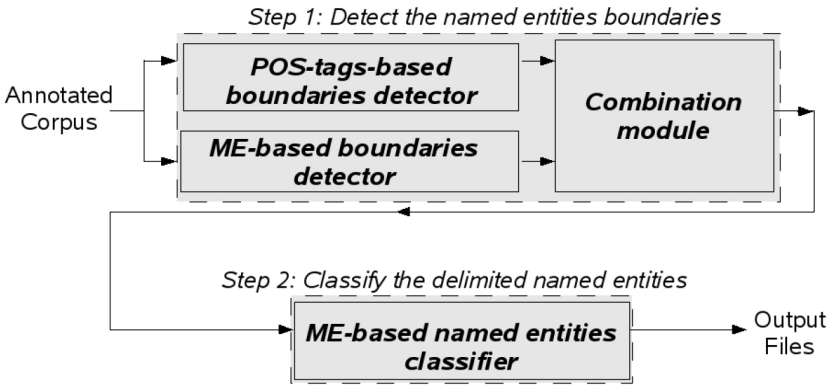


Figure 4.1: Generic architecture of ANERsys 2.0 system [Benajiba and Rosso, 2007].

based classifier showed an overall F-measure = 70.67% with tokenised data and F-measure = 67.76% on raw text. The CRF-based classifier achieved the best results when all the features were combined. The overall performance in terms of F-measure was 79.21% on the ANERcorp test set.

Benajiba *et al.* [2008a] applied Support Vector Machines (SVMs) for Arabic NER. They also investigated the use of lexical, contextual, morphological, gazetteer, and shallow syntactic features in the SVMs classifier. The impact of the different features was also measured in isolation and combined. The evaluation was considered based on the ACE data sets. The results showed that the high performance was achieved using a combination of all features. The overall performance in terms of F-measure was 82.71% for ACE 2003, 76.43% for ACE 2004, and 81.47% for ACE 2005.

Benajiba *et al.* [2008b] examined the impact of using different sets of features for each NE type instead of simultaneously using a single set of features for all NE types. So, they created a classifier for each NE type that was independently based on an optimal feature set, and then combined the different classifiers for a global NER system. An incremental feature selection method was used to select an optimised feature set and to observe the errors. In order to create a classifier for each NE type, they adopted two discriminative approaches: Support Vector Machines (SVMs) and Conditional Random Fields (CRFs). They investigated many sets of features for each NE type in isolation first, then they ranked the features according to the best performance of the two classifiers for each NE

type. In total, 16 features representing the lexical, contextual, morphological, gazetteer, and shallow syntactic features were examined. Finally, the classifications of all individual classifiers (one classifier per NE type) are combined. In case of conflicts, where the same token was annotated as two different NE types, they used a simple heuristic based on the classifier precision for that specific tag, favoring the tag with the highest precision. The evaluation process was conducted using the ACE data sets. They concluded that each feature plays a role in recognising the NE to different degrees and each NE type is sensitive to different features. The best system's overall performance in terms of F-measure was 83.5%, 76.7%, and 81.31% for ACE 2003, ACE 2004, and AC 2005 respectively. The study also reported that it can not be concluded whether CRFs is better than SVMs or vice versa.

Abdul-Hamid and Darwish [2010] developed a CRF-based classifier with a simplified feature set that relied primarily on character level features (leading and trailing letters in a word). They also used a variety of features such as word sequence features, word length, word position, and word n-gram probability-based features to capture the distribution of NEs in the text. The evaluation was carried out using ANERcorp and the ACE 2005 data set. They reported an F-measure of 81% and 76% for the ANERcorp and the ACE 2005 data sets respectively.

Darwish [2013] developed a CRF-based classifier, and presented cross-lingual features for Arabic NER that made use of the linguistic properties and knowledge bases of another language. In his study, English capitalisation features and an English knowledge base (DBpedia) were exploited as discriminative features for Arabic NER. A large Machine Translation (MT) phrase table and Wikipedia cross-lingual links were used for translation between Arabic and English. The classifier was evaluated on the ANERcorp data set and achieved an overall F-measure equal to 84.3%.

The semi-supervised learning is a relatively new area of research [Nadeau and Sekine, 2007]. Most studies that revolve around semi-supervised methods for NER are conducted in English [Collins and Singer, 1999; Jones, 2005; Nadeau *et al.*, 2006; Riloff and Jones, 1999]. In terms of Arabic NER, AbdelRahman *et al.* [2010] presented an integrated approach to Arabic NER in which they combined bootstrapping semi-supervised pattern recognition

and the Conditional Random Fields (CRFs) classifier. So, the semi-supervised pattern recogniser extracts all patterns to let CRF-based classifier identify more NEs. The CRF-based classifier in turn is trained using some local optimal features, including pattern index, with the aim of generating potential seeds to help remove noisy patterns. Their proposed integrated method achieved F-measures of 67.80%, 87.80%, 70.34% for Person, Location, and Organisation respectively when tested on ANERcorp test set. They also concluded that bootstrapping semi-supervised pattern recognition is a promising technique for Arabic NER.

Mohit *et al.* [2012] considered the domain adaptation problem which usually occurs when supervised learning on newswire text leads to poor target-domain recall. They explored the precision-recall tradeoff and proposed a recall-oriented learning method that enhanced recall over precision. So, a model was built using the structured perceptron and trained on newswire labelled data. Then, the loss function used in the training was manipulated in order to improve recall over precision. This function measures the entity recognition errors for each token in the corpus. That is, the penalties are imposed when recall errors happen. Mohit *et al.* [2012] also adopted a semi-supervised learning method (self-training) in order to tag large unlabelled data (397 Arabic Wikipedia articles). The recall-oriented perceptron was evaluated when used alone or combined with self-training. The evaluation set consists of 20 Wikipedia articles annotated manually to identify domain-specific entity types in addition to the three standard categories. The experimental results showed recall improvements of nearly 8 percentage points over the baseline (regular perceptron). Integrating a recall-oriented perceptron within self-training produced a recall improvement of about 4 percentage points compared to baseline when used within self-training.

Alotaibi and Lee [2012] conducted a series of experiments in attempt to classify Arabic Wikipedia articles into a predefined set of NE types. They explored the use of Naïve Bayes, Multinomial Naïve Bayes, Support Vector Machines, and Stochastic Gradient Descent for classifying Wikipedia articles, and achieved F-measures ranging from 78% to 90% using different language-independent and Arabic-specific features. Their research showed that the use of enhanced Arabic-specific features does not always lead to the best performance,

especially when combined with the *TF-IDF* statistic.

Alotaibi and Lee [2013] developed a methodology to automatically create two NE annotated sets from Arabic Wikipedia. The corpora were built using the mechanism that transforms links into NE annotations by classifying the target articles into named entity types. They used POS tagging, morphological analysis, and linked NE phrases to detect other mentions of NEs that appear without links in text. The data sets created by Alotaibi and Lee [2013] are $\text{WikiFANE}_{\text{Whole}}$ and $\text{WikiFANE}_{\text{Selective}}$. The former set contains all sentences retrieved from the Wikipedia articles while the latter set contains only the sentences that have at least one named entity phrase. The trained models on the $\text{WikiFANE}_{\text{Whole}}$ and $\text{WikiFANE}_{\text{Selective}}$ performed well when tested on various newswire test sets, achieving F-measures of 56.39% and 52.62% respectively when tested on ANERcorp data set. Their trained models, however, did not surpass the performance of the NE classifier that was trained and tested on data sets drawn from the same domain and/or corpous.

4.5.3 Hybrid Approach

The hybrid approach integrates the rule-based approach with the machine learning-based approach in order to enhance the overall performance of the NER system [Shaalan, 2014]. Abdallah *et al.* [2012] presented a hybrid system for Arabic NER. The rule-based component of their hybrid system is a re-implementation of the NERA system developed by Shaalan and Raza [2009], while the machine-learning based component employs the Decision Trees. The feature set includes language-independent features, Arabic-specific features, and the NE tags predicted by the rule-based component. The evaluation of their hybrid system using ANERcorp data set showed F-measures of 92.8%, 87.39%, and 86.12% for the Person, Location, and Organization NEs respectively.

Oudah and Shaalan [2012] extended the study of Abdallah *et al.* [2012] by investigating two more machine learning methods: SVMs and Logistic Regression. They also experimented with more features by including morphological features and an English-gloss capitalisation feature. The evaluation was conducted using the ANERcorp data set. The experimental results proved that their best hybrid Arabic NER system outperformed the

rule-based and the machine learning-based components when they were run individually. The F-measure performance was 94.4%, 90.1%, and 88.2% for Person, Location, and Organisation NEs respectively.

4.6 Summary

This chapter highlighted the challenges that emerged when developing Arabic NER systems because of Arabic's characteristics and peculiarities. In addition, the basic computational tasks and common tools used in Arabic NER systems were described in detail. The chapter also described the approaches used in the Arabic NER field and the most recent research. It showed that many Arabic NER systems were created using grammar-based techniques as well as machine learning methods.

The main goal of the new approaches to NER and one of the possible directions in NER research is to reduce the annotation labour by employing minimally-supervised methods, such as semi-supervised learning and distant learning, which require less effort and limited human intervention.

In the following chapters we present the experiments that we have carried out to explore the minimally-supervised techniques for Arabic NER and to exploit these techniques to overcome the challenges of Arabic such as rich morphology and the absence of capitalisation. Chapter 5 presents our semi-supervised learning approach to Arabic NER in which we propose a new way to produce and generalise the extraction patterns. Chapter 6 presents our methodology to exploit Wikipedia structure to automatically develop an Arabic NE annotated corpus. We also introduce a mechanism based on the high coverage of Wikipedia in order to address the challenges particular to tagging NEs in Arabic text. So, our method does not require POS tagging or morphological analysis. Chapter 7 describes our novel approach in which we combine the two minimally supervised methods in order to obtain better results. Up to our knowledge, our proposed approach is the first to combine minimal supervision approaches.

Part III

Thesis Contributions

Chapter 5

Semi-supervised Learning

In order to lay a proper foundation on which to present our research, in previous chapters we explained the background and the state-of-the-art works related to NER in general and Arabic NER in particular. Our research is presented in this chapter, and in the next two chapters (i.e., Chapter 6 and 7).

This chapter presents a semi-supervised learning approach to Arabic NER. Our approach relies on the *bootstrapping* technique, which starts with a handful of seed instances for the NE type of interest, and learns the extraction patterns, which are exploited to identify more instances for that NE type (candidate NEs). Then, the candidate NEs are sorted according to a ranking measure, so that the best of them are selected as the next seed instances when the process repeats. Additionally, we propose a new way to produce and generalise the extraction patterns. Filtering criteria are proposed in order to remove noisy extraction patterns. We also present and compare two ranking measures for determining the most reliable candidate NEs: one being the number of distinct patterns used in extracting candidate NEs and another based on Pointwise Mutual Information (PMI). We evaluate our algorithm in order to extract the three standard NE types (Person, Location, Organisation) and to extract three new specialised types of NE (Politicians, Sportspersons, Artists). Section 5.1 illustrates the architecture of the proposed algorithm and its main components in detail. Section 5.2 describes the corpora used in the experiments and the evaluation process. Section 5.3 presents the experimental setup and results for evaluating the algorithm's ability to recognise standard and specialised entities. The experimental results are discussed in Section 5.4. Conclusions about our semi-supervised approach to

Arabic NER are drawn in Section 5.5.

5.1 The Algorithm

The most common Semi-supervised Learning (SSL) technique is *bootstrapping* [Nadeau and Sekine, 2007], which only requires a set of seed instances to initiate the learning process. Our SSL algorithm begins with seed instances of a given NE type (e.g., فرنسا *faransaa* "France", المتحف البريطاني *almtHaf albryTaany* "British Museum", and قلعة هوارد *qlʻt hwaard* "Castle Howard" can be used as seed instances for Location), and learns patterns that are used to extract more instances (*candidate NEs*). These candidate NEs are sorted so that the best of them will be selected as seed instances for the next iteration. Figure 5.1 shows the three components of our algorithm.

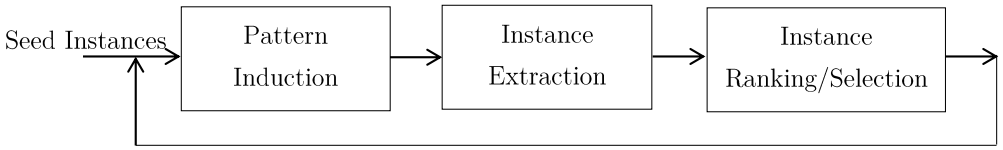


Figure 5.1: The three components of our SSL algorithm.

In order to represent the NE extraction patterns, we proposed a new way that uses a combination of tokens and POS-tags. Therefore, the training corpus should be automatically assigned part-of-speech tags. In addition, for each particular type of NE, lists of *trigger words* (i.e., NE indicators that help identify NEs within text) were used in the algorithm to generalise the extraction patterns and to filter them. The lists of trigger nouns were semi-automatically extracted from randomly selected Arabic Wikipedia articles. To be exact, for each type of NE, 100 to 200 pages from Arabic Wikipedia were crawled randomly, prepared, and pos-tagged to be used to construct trigger nouns. We extracted nouns that appear most frequently before or after the NE and stored them as trigger nouns. Trigger verbs are the most frequent verbs (stems) that appear before or after NEs in the Arabic Wikipedia articles. Trigger verbs and nouns, which surround NEs, were identified in order to find the most common Arabic NE indicators. Some examples of trigger nouns are: السيد *alsayd* "Mr.", السيدة *alsaydh* "Mrs.", and بن *bn* "the son of" for Person; مدينة *madynah*

"city", and ولاية *wilaayah* "state" for Location; شركة *sharikah* "company", and وكالة *wikaalah* "agency" for Organisation. Algorithm 5.1 shows the pseudo-code of our SSL algorithm.

Algorithm 5.1: Semi-supervised algorithm for Arabic NER.

Input: Seed Instances for each NE type
/* Initial seed instances to start learning process */
Data Preparation: Part-of-speech-tagged training corpus
Lists of *trigger words* for each NE type
Reliable_NEs = {Seed Instances}
m=Reliable_NEs.size()
Initial_P = { }
/* The set of initial patterns */
Final_P = { }
/* The set of final patterns*/
Candidate_NEs = { }
repeat
1. $m = m+1$
2. Generate the extraction patterns from the training corpus using *Reliable_NEs* and save them to *Initial_P*
3. Generalise the patterns in *Initial_P* using *trigger words*, and save them to *Final_P*
4. Filter the patterns in *Final_P* to remove noisy ones
5. Apply the patterns in *Final_P* to the training corpus and save the extracted entities to *Candidate_NEs*
6. Rank all extracted entities in *Candidate_NEs* using a ranking measure
7. *Reliable_NEs* = Top m extracted entities in *Candidate_NEs*
until *the end condition*

The following sections explain the proposed pattern's representation, generalisation, and the filtering steps we applied to them. They also explain the two ranking measures we used for determining the most reliable extracted entity.

5.1.1 Pattern Induction

This section illustrates the first component of our semi-supervised algorithm. It describes the process of extracting patterns from the text and generalising them to produce the final set of patterns. This section also presents the filtering criteria which are used to remove noisy patterns.

5.1.1.1 Initial Patterns

Our algorithm infers a set of surface patterns that contain seed instances in the training corpus. So, for each seed instance x, all sentences containing the term x are first retrieved. Since words preceding or following the target word may be useful for determining its category, the algorithm extracts a number of tokens n on each side of the seed x without crossing sentence boundaries. Figure 5.2 is an example of initial patterns containing the seed instance محمد *muhammad* "Muhammad" and its surrounding tokens.

Arabic Pattern:

- نوه / VBD الدكتور/NN محمد/NNP البشر /NNP سفير/NN المملكة/NN العربية/JJ السعودية/JJ في /IN المغرب/NNP

English Translation:

- Dr. /NN Mohammed/NNP Albshr/NNP the/DT ambassador/NN of/IN Saudi/NNP Arabia/NNP in/IN Morocco/NNP indicated/VBD that/DT

Figure 5.2: An example of an initial patterns extracted from data by the SSL algorithm.

We refer to each *Token/POS-tag* pair as *TP pair* (e.g., "indicated/VBD" represents one *TP pair*). Noun tokens in TP pairs are kept in their inflected form, while verb tokens are replaced with their roots. So, the inflected verbs كتبت *katabt* "wrote" and تكتب *taktub* "writes" are changed to كتب *katab* "write" (see Section 2.3.1 for more information about inflectional and derivational features in Arabic). A root stemmer is used instead of a light stemmer because the latter fails to conflate related forms that should group together (see Section 4.3.4). A root stemmer is preferred in our case to obtain as many general patterns as possible.

5.1.1.2 Generalisation

In the next step, the initial patterns are generalised in order to produce more effective extraction patterns that contain only useful evidence (e.g., NE indicators) and more general information about the categories of the words in the patterns. For that purpose, verbs and nouns that are not among *trigger words* should be removed, as they do not provide

useful evidence of the existence of NEs in the text, and they also restrict the patterns excessively. In contrast, prepositions represent important clues, since they usually precede NEs. Therefore, we keep them without any change in the patterns. Detailed information about word categories are neglected because our proposed method does not require this information. For example, all POS tags used for verbs to indicate the tense (e.g., *VBP, VBD, VBN*) are converted to one form: *VB*. In summary, all extracted initial patterns should complete the following steps in order to generate the final patterns:

- TP pairs that contain nouns, and verbs are stripped of their *Token* parts, unless they are in the corresponding lists of trigger words. For example, TP pair (السيد/NN "Mr./NN") stays unchanged since (السيد "Mr.") is in the list of trigger nouns, while (قلم/NN "pen/NN") is changed to only ' */ NN*' as (قلم "pen") is not among trigger nouns.
- TP pairs that contain prepositions are not changed.
- TP pairs that contain other part of speech categories (e.g., proper nouns, adjectives, coordinating conjunctions) are stripped of their *Token* parts. For example, (مفيد/JJ 'useful/JJ') is converted to only ' */JJ*' without the *Token* part.
- All POS tags used for verbs (e.g., *VBP, VBD, VBN*) are converted to one form: *VB*.
- All POS tags used for nouns (e.g., *NN, NNS*) are converted to one form: *NN*.
- All POS tags used for proper nouns (e.g., *NNP, NNPS*) are converted to one form: *NNP*.
- The seed instance is replaced with the NE class tag (e.g., <PersonName>, <Location>, <Organisation>).

Figure 5.3 shows the final pattern resulting from the initial pattern, after the constrained processes mentioned above are applied.

Arabic Pattern

- / VB الدكتور/NN <PersonName>/NNP /NNP سفير/NN /NN /JJ /JJ في /IN /NNP

English Translation

- Dr. /NN <PersonName>/NNP /NNP /DT ambassador/NN of/IN /NNP /NNP in/IN /NNP /VB that/DT

Figure 5.3: An example of a final pattern resulting from the SSL algorithm.

All final patterns that are generated from the algorithm are gathered to form the pattern set (*Final_P*). In the final step, two more patterns are generated from every pattern in *Final_P*. The algorithm splits every final pattern into two parts, where each seed instance is located in the leftmost or rightmost position in the pattern. The two patterns generated from our previously mentioned example can be seen in Figure 5.4.

Final Pattern: / VB الدكتور/NN <PersonName>/NNP /NNP سفير/NN /NN /JJ /JJ في/IN /NNP

1- Rightmost position:

- / VB الدكتور/NN <PersonName>/NNP (Arabic Pattern)

  Dr. /NN <PersonName>/NNP (English Translation)

2- Leftmost position:

- <PersonName>/NNP /NNP سفير/NN /NN /JJ /JJ في /IN /NNP

  <PersonName>/NNP /NNP /DT ambassador/NN of/IN /NNP /NNP

Figure 5.4: An example of generating more patterns from a final pattern.

The rationale behind this is to increase the generality of the patterns by making them shorter in length, thus increasing their ability to collect more *candidate NEs* in the matching process against the text. For example, the short pattern "*Dr./NN <PersonName >*" might successfully match more NEs in the text than the long pattern illustrated in Figure 5.3. However, short patterns, which have TP-pairs containing no *Token* parts at all but POS taggings, are a potential source of noise. Therefore, the final pattern set (*Final_P*) is filtered every time a new pattern is added to it. Repeated patterns are not added. In

addition, the pattern that contains a number of *TP-pairs* less than a predefined threshold (*filtering_threshold*) should contain at least one TP-pair with a *Token* part. For example, if we set *filtering_threshold* = 6, then the pattern "*/VB /NN <PersonName >/NNP /NNP*" is rejected and not added to the set of final patterns because it consists of four *TP-pairs* (<*filtering_threshold*) and does not contain a *TP-pair* with a *Token* part. Algorithm 5.2 shows the filtering steps we applied to the set of patterns.

Algorithm 5.2: Filtering the extraction patterns algorithm.

Input: A set of final patterns $Final\_P = \{p_1, p_2, \ldots, p_n\}$

```
1  for i ← 1 to n do
2      TP ← split p_i into TP-pairs
3      /* e.g., the pattern (Dr./NN <PER>/NNP /DT ambassador/NN) contains four
          TP */
4      if (TP.size() >= filtering_threshold) then
5          keep p_i in the set Final_P
6      else
7          token_found = FALSE
8          for j ← 1 to TP.size() do
9              if TP_j contains Token part then
10                 token_found = TRUE
11         if token_found then
12             keep p_i in the set Final_P
13         else
14             remove p_i from the set Final_P
```

5.1.2 Instance Extraction

In this phase, our SSL algorithm retrieves the set of instances I from the training corpus that match any of the patterns in *Final_P*. First of all, we must ensure that the generalisation steps used in inducing patterns have already been applied to the training corpus in order to prepare it for the matching process (e.g., VBD, VBP, and VBN are converted to VB and so on). The matching of patterns in *Final_P* against the corpus is done using regular expressions (regexes). For example, the regex for the pattern " */VB* الدكتور*/NN <PersonName >/NNP*" is depicted in Figure 5.5.

```
Arabic Regex: [^/]*/VB\s\bدكتور\b/NN\s([^/]*)/NNP
English Translation: [^/]*/VB\s\bDr.\b/NN\s([^/]*)/NNP
```

Figure 5.5: An example of regex automatically generated from a final pattern.

Our SSL algorithm automatically converts extraction patterns into regular expressions that can be used to match against the text. Algorithm 5.3 illustrates the steps of this process.

Algorithm 5.3: Converting extraction patterns to RegEx algorithm.

Input: A set of final patterns $Final\_P = \{p_1, p_2, \ldots, p_n\}$

```
for i ← 1 to n do
    TP ← split p_i into TP-pairs
    for j ← 1 to TP.size() do
        Token = match TP_j against "[^/]*"
        POS = match TP_j against "/([a-zA-Z_]+)"
        TokRegex = " "
        Regex = " "
        if Token.isEmpty then
            TokRegex = "[^/]*"+"/"+POS+"\s"
            Regex.append(TokRegex)
        else
            if Token.contains("<") AND Token.contains(">") then
                TokRegex = "([^/]*)"+"/"+POS+"\s"
                Regex.append(TokRegex)
            else
                TokRegex = "\b" +Token+"\b"+"/"+POS+"\s"
                Regex.append(TokRegex)
```

The generation of regexes from final patterns is achieved without any modification, regardless of whether the POS tags assigned to the proper nouns by the POS tagger are accurate or not. Because of the absence of capitalisation in Arabic, Arabic POS taggers may mistake some organisations and locations for nouns (NN) or adjectives (JJ), especially Arabic names that have an interpretation as a common noun or adjective. For example, الامارات العربية المتحدة "United Arab Emirates", and الامم المتحدة "United Nations" might be tagged as الامارات/*NNS* العربية/*JJ* المتحدة/*JJ* and الامم/*NNS* المتحدة/*JJ*.

In order to increase the number of NEs collected in each iteration, we allowed our SSL algorithm to automatically add the information of average NE length[1] to the produced regexes. Figure 5.6 presents an example of adding the average NE length to the produced regex.

```
Arabic Regex: [^/]*/VB\s\bدكتور|السكتور\b/NN(\s([^/]*)/NNP){1,2}
English Translation: [^/]*/VB\s\bDr.\b/NN(\s([^/]*)/NNP){1,2}
```

Figure 5.6: An example of automatically generated regex with average NE length.

5.1.3 Instance Ranking/Selection

Our SSL algorithm ranks each extracted instance (i.e., candidate NE) in the set of instances I according to the number of different patterns that are used to extract it [Baroni *et al.*, 2010]. This ranking measure arises from the intuition that a candidate NE extracted by five different patterns for a NE type of interest is more likely to belong to that entity type than a candidate NE extracted by only one pattern [Riloff and Jones, 1999].

We prefer a threshold defined by distinct patterns instead of a frequency because pattern variety is a better cue to semantics than absolute frequency [Baroni *et al.*, 2010]. In addition, we used distinct patterns to overcome errors caused by the absence of capitalisation in Arabic. For example, in the following sentence و قد كتب اعلى اللوحة "and he has written at the top of the board" the word اعلى "the top" is not a person name nor a proper noun. However, it was wrongly detected and classified as a person name by the following pattern:

/CC /RP كتب*/VB <PersonName >/NNP /NN .*

This pattern is relatively similar to the following pattern in English (with some differences in the word order as the verbs usually come before the nouns in Arabic while the opposite is true in English):

/CC <PersonName >/NNP write/VB /NN .

Therefore, the problem of considering the word اعلى "the top" to be a person name is not

[1]The statistics about the length of proper nouns we derived from two Arabic NER corpora (ANERcorp and AQMAR) showed that the average NE length is 2 tokens (see Table 5.1).

due to the weakness of the pattern, but the wrong POS tag associated to the word itself, since the POS tagger considers اعلى "the top" to be an NNP every time the word appears in this context. Subsequently, if the above sentence or a similar one happens to appear three times in the text, the word اعلى "the top" might be ranked before the correct NE محمد "Muhammad", which appears only twice in the text, but in different contexts: الروائي محمد "The novelist Muhammad", and "و قد كتب محمد الرواية "and Muhammad wrote the novel".

Our algorithm ranks the instances according to the number of distinct patterns, and discards all but the top m, where m is set to the number of instances from the previous iteration, plus one. These m instances are used in the next iteration, and so on. For example, if we start the algorithm with 10 seed instances, the following iteration will start with 11, and the next one will start with 12, and so on. This procedure is necessary in order to ensure that reliable instances from the previous iteration are the only ones to be included in the next iteration.

An alternative to distinct patterns is the Pointwise Mutual Information (PMI) statistic, which is commonly used in text mining [Turney *et al.*, 2010]. For this reason, we also used the Pointwise Mutual Information (PMI) to measure the association strength of the instance i in the set of instances (I) across each pattern in the set of patterns ($Final\_P$). A reliable instance is one that is associated with as many patterns in $Final\_P$ as possible.

PMI of instance i can be computed as follows:

$$pmi(i) = \sum_{P \epsilon Final\_P} \log \frac{|i,p|}{|i| * |p|} \tag{5.1}$$

where $|i,p|$ is the frequency of the instance i extracted by pattern p. $|p|$ is the frequency of the pattern p in the corpus. $|i|$ is the frequency of the instance in the corpus.

5.2 Data

This section describes the two corpora we used in our experiments: ANERcorp and ACE 2005.

ANERcorp, which is freely available for research purposes, contains approximately 150K

tokens (11% of the tokens are NEs). It is composed of a training set and a test set built and tagged especially for the NER task by Benajiba *et al.* [2007]. More details about ANERcorp are found in Section 4.2.

The second data set used in the training phase is ACE 2005. It is available from the Linguistic Data Consortium (LDC) and has approximately 113K tokens. The genres utilised in ACE 2005 are Broadcast News, NewsWire, and WebLogs (see Section 4.2). Each document in the corpus has four versions:

1. Source text files with extension ".sgm": These files contain raw Arabic text surrounded by begin text tag <TEXT>and end text tag </TEXT>. This is the only text to be evaluated.

2. APF files with extension ".apf.xml": These files contain the ACE Program Format.

3. AG files with extension ".ag.xml": These files contain the LDC Annotation Graph format and can be viewed with the LDC's annotation tool.

4. TABLE files with extension ".tab": These files store mapping tables between the IDs used in each ag.xml file and their corresponding apf.xml file.

These two corpora were used as follows:

- Training set (90% of ANERcorp training set + ACE 2005): As our SSL algorithm does not require any annotations in the training phase, we removed all annotations from the ANERcorp training set. Regarding the ACE corpus, raw Arabic text has been extracted from files with the 'sgm' extension without any kind of annotations.

- Validation set (10% of ANERcorp training set): This set was used to assign appropriate values to several parameters in our system, such as the number of initial seeds, the criterion to stop the training process, and so on.

- Test set (ANERcorp test set): This set was used for evaluation purposes.

The ANERcorp training set was randomly divided into two parts: 90% of this data set was kept for training, while 10% was dedicated for validation purposes.

5.3 Experiments and Results

5.3.1 Experimental Setup

We found that Arabic proper nouns in ANERcorp and AQMAR corpora (see Section 4.2) are one or two words long 92.09% and 84.39% of the time respectively. Table 5.1 shows the statistics about the length of proper nouns we derived from the two aforementioned corpora.

Corpus	One word	Two words	Three words	Four words	Five words	Six words	Seven words	Eight words	Nine words	> 9 words
AQMAR	55.26%	29.13%	9.50%	4.41%	0.91%	0.51%	0.12%	0.07%	0.02%	0.07%
ANERcorp	61.01%	31.08%	5.49%	1.43%	0.64%	0.16%	0.16%	0.02%	0.01%	0.00%

Table 5.1: The statistics of Arabic proper nouns length in ANERcorp and AQMAR corpora.

Motivated by the statistics in Table 5.1, the average NE length was set to 2 tokens. In addition, the experiments we conducted using the validation set showed that increasing the average length of proper names to more than 2 tokens increased the recall but negatively affected the precision of the collected NEs. They also showed that around 70% of noisy patterns extracted were short patterns (fewer than six *TP-pairs* without *Token* parts). Therefore, we set *filtering_threshold* equal to 6 (see Algorithm 5.2). The window size of tokens when extracting the initial patterns was set to 14, 7 tokens on each side of the seed instance, without crossing sentence boundaries. We used a window size of 7 tokens because, based on the statistics in Table 5.1, the longest Arabic name is 5 to 7 tokens. Thus, we extracted 7 tokens to the left and to the right of the seed instance to identify the borders of the extraction pattern, which includes the NE indicators. For example, if we have the seed instance أحمد "Ahmad" and the text عالم الفلك ابو الحسن علي بن عبدالرحمن بن أحمد بن يونس "The astronomer Abu alHasan Ali ibn Abdulrahman ibn Ahmad ibn Yunus", then choosing 7 tokens to the left and to the right of the seed will result in extracting the important indicator 'عالم الفلك', 'The astronomer'. Therefore, choosing less or more than 7 tokens would have

little to no benefit. In addition, some informal experiments we conducted demonstrated that using larger spans had no significant effect on increasing the quality of the patterns.

ANERcorp and ACE corpora were preprocessed in order to prepare them for our proposed algorithm. Thus, sentence detection was applied to the corpora. We used the sentence detector that we implemented and offered as part of our AraNLP library (see Appendix C). Then, we conducted clitic tokenisation (i.e., decliticisation), since neglecting clitics altogether may cause a loss of important information when generating the patterns. For example, a prepositional proclitic بـ "in" usually comes before location names and keeping this prepositional proclitic attached to the Arabic word will affect the amount of useful information that the trained model can learn from the text. The trained model will regard the preposition as part of the Arabic word itself and will not be able to extract the important clue, that the preposition precedes the Arabic word. Consequently, we chose a decliticisation scheme in which conjunctions, prepositions, and future marks are separated from each token.

In order to find the root for each verb in the corpus (see Section 5.1), we chose Khoja's stemmer, one of the most common and successful approaches to Arabic stemming (More details about Khoja's stemmer are found in Section 4.3.4.1).

Regarding POS tagging, we used the AMIRA toolkit [Diab, 2009] and chose the Reduced Tag Set (RTS), which ignores many morphological features in Arabic, since our proposed method does not require any deep morphological information related to gender, number, or definiteness. The input to AMIRA was our training corpus in the form of one sentence per line; this is the training corpus after applying a sentence splitter to it. So, AMIRA ran AMIRA-TOK on our raw text and then performed POS tagging. We chose the output of this phase to be POS tags assigned to the surface words in our corpus.

5.3.1.1 Parameter Settings

We developed several experimental models according to three parameters that are defined in our proposed algorithm: the number of initial seeds, the ranking measure, and the number of iterations (see Algorithm 5.1).

- **Experiment 1**: We started with simple models, which were trained on the ANERcorp corpus and passed through the three components only once (see Figure 5.1). For each NE class, we only started with five seed instances. We refer to this model as *Simple-Model-5*. We also trained two more models, *Simple-Model-10* and *Simple-Model-20*, which only differed from Simple-Model-5 in the number of seed instances for each NE class; the number of seeds were 10 and 20 respectively. These simple models are considered the baselines.

- **Experiment 2**: We iterated the algorithm through the three components (see Figure 5.1). The number of iterations was set to ten, because the validation set indicated that increasing the number of iterations to more than ten loops does little to improve the performance of the system (F-measure improvement <0.01). We started with 20 seed instances for each NE class and the training corpus was ANERcorp. Candidate NEs were ranked according to the number of distinct patterns in order to select those that ranked the highest as seed instances for the next iteration, as explained in Section 5.1.3. We refer to these trained models, one model for each NE class, as 'Model-A(NE class)'. We also used Pointwise Mutual Information (PMI) as a ranking measure to determine the most reliable candidate NEs and compared it with 'the number of distinct patterns' as a ranking measure.

- **Experiment 3**: In order to see the effect of using a large training corpus on the performance of the trained model, a combination of the ANERcorp training set and ACE 2005, was used in the training phase. We refer to the trained models resulting from this experiment as 'Model-B(NE class)'.

5.3.1.2 Specialised NEs

Semi-supervised learning can be easily adapted to detect new NE types. Therefore, our SSL algorithm was also tested to recognise three specialised NEs: Politicians, Sportspersons, and Artists. These new types were chosen because they constitute the largest percentage of persons' names in ANERcorp. For evaluation purposes, all annotated persons' names in

the ANERcorp test set must be re-annotated to include the three new entities: Politicians, Sportspersons, and Artists. We re-annotated the test set, so that all annotations of persons' names (e.g., *B-PER, I-PER*) were changed to one of the following classes:

- B-POL: The beginning of the name of a POLitician.
- I-POL: The continuous (Inside) of the name of a POLitician.
- B-SPORT: The beginning of the name of a SPORTsperson.
- I-SPORT: The continuous (Inside) of the name of a SPORTsperson.
- B-ART: The beginning of the name of an ARTist.
- I-ART: The continuous (Inside) of the name of an ARTist.
- B-PER: The beginning of the name of a PERson that does not belong to any of the previous classes.
- I-PER: The continuous (Inside) of the name of a PERson that does not belong to any of the previous classes.

Unlike supervised learning, which may require additional annotated data in the training data for new categories of NE, our semi-supervised approach uses the ANERcorp training data without any addition or modification. The methodology was applied without any major modifications. The only modification made related to generating new lists of trigger nouns and verbs for each new type of NE (i.e., Politicians, Sportspersons, Artists). They were generated in the same way as explained in Section 5.1.

5.3.2 Results

Table 5.2 shows the results of the baselines for each NE type when applied to the ANERcorp test set.

		Simple-Model-5	**Simple-Model-10**	**Simple-Model-20**
PER	Precision	88.09	**88.46**	86.32
	Recall	36.01	39.06	**43.20**
	F-measure	51.12	54.19	**57.58**
LOC	Precision	89.96	86.80	**90.55**
	Recall	51.88	55.10	**55.45**
	F-measure	65.81	67.41	**68.78**
ORG	Precision	**85.95**	83.87	84.35
	Recall	22.38	23.66	**27.67**
	F-measure	35.51	36.91	**41.67**

Table 5.2: The results of the baseline models.

Table 5.3 shows the results of evaluating Model-A for the three standard NEs on ANERcorp test set.

Model-A (PER)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	85.57	86.45	86.09	86.09	86.09	86.09	86.13	86.19	86.25	85.91
Recall	44.13	49.82	49.96	49.96	49.96	49.96	50.17	50.17	50.46	51.10
F-measure	58.23	63.21	63.23	63.23	63.23	63.23	63.41	63.42	63.67	**64.08**
Model-A (LOC)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	90.58	88.21	88.03	87.86	87.86	87.86	87.89	87.89	87.89	87.91
Recall	55.72	61.93	61.93	61.93	61.93	61.93	62.06	62.06	62.06	62.48
F-measure	69.00	72.77	72.71	72.65	72.65	72.65	72.75	72.75	72.75	**73.04**
Model-A (ORG)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	83.94	83.15	82.73	83.33	83.33	83.16	83.16	83.16	84.27	84.27
Recall	30.18	34.81	35.16	36.70	36.70	36.87	36.87	36.87	40.30	40.30
F-measure	44.40	49.08	49.35	50.96	50.96	51.09	51.09	51.09	54.52	**54.52**

Table 5.3: The results of Model-A.

Table 5.4 shows the results of evaluating Model-A when we used PMI as a ranking measure instead of using the number of distinct patterns. The comparison of the models when used with different ranking measures is shown in Table 5.5.

Model-A-PMI (PER)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	85.57	85.74	85.74	85.76	85.76	85.76	85.76	85.97	85.97	85.97
Recall	44.13	44.56	44.56	44.63	44.63	44.63	44.63	44.64	44.64	44.64
F-measure	58.23	58.64	58.64	58.71	58.71	58.71	58.71	58.77	58.77	**58.77**
Model-A-PMI (LOC)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	90.6	90.6	90.6	90.6	90.6	90.61	90.61	90.62	90.62	90.62
Recall	55.86	55.86	55.86	55.86	55.86	55.87	55.87	56.00	56.00	56.00
F-measure	69.11	69.11	69.11	69.11	69.11	69.12	69.12	69.22	69.22	**69.22**
Model-A-PMI (ORG)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	85.47	85.53	85.53	85.53	85.53	85.54	85.54	85.54	85.54	85.54
Recall	33.78	33.79	33.79	33.79	33.79	33.80	33.80	33.80	33.80	33.80
F-measure	48.42	48.44	48.44	48.44	48.44	48.45	48.45	48.45	48.45	**48.45**

Table 5.4: The results of Model-A-PMI.

		Model-A	**Model-A-PMI**
PER	Precision	85.91	**85.97**
	Recall	**51.10**	44.64
	F-measure	**64.08**	58.77
LOC	Precision	87.91	**90.62**
	Recall	**62.48**	56.00
	F-measure	**73.04**	69.22
ORG	Precision	84.27	**85.54**
	Recall	**40.30**	33.80
	F-measure	**54.52**	48.45

Table 5.5: The comparison between Model-A and Model-A-PMI.

Regarding the effect of using large corpora on the performance of our SSL algorithm, we trained Model-B on a combination of the ANERcorp training set and ACE 2005, as explained previously. Table 5.6 shows the results of evaluating Model-B on the ANERcorp test set.

Model-B (PER)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	85.37	85.93	85.49	85.54	85.30	85.31	85.14	85.27	84.74	84.70
Recall	45.91	50.81	51.31	51.32	51.32	51.32	51.38	51.59	51.59	52.03
F-measure	59.71	63.86	64.13	64.15	64.08	64.09	64.09	64.29	64.13	**64.46**
Model-B (LOC)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	89.13	87.16	87.16	87.16	87.18	87.18	87.18	87.23	87.12	87.05
Recall	56.55	62.75	62.75	62.75	62.89	62.89	62.89	63.17	63.44	64.00
F-measure	69.20	72.97	72.97	72.97	73.07	73.07	73.07	73.28	73.42	**73.77**
Model-B (ORG)										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	85.15	85.12	84.69	84.28	84.33	84.54	84.54	84.54	84.37	84.37
Recall	32.67	33.79	38.07	38.42	38.59	40.13	40.13	40.13	40.31	40.31
F-measure	47.22	48.38	52.53	52.78	52.95	54.43	54.43	54.43	54.55	**54.55**

Table 5.6: The results of Model-B.

Table 5.7 summarises the trained models with their values for each parameter. It also shows the performance of each model when applied to the ANERcorp test set by computing their F-measures for each type of NE.

Parameters Setting				**Results**		
Trained Models	Training Corpus	Ranking measure	No. of initial seeds	PER F-measure	LOC F-measure	ORG F-measur
Simple Model-5	ANERcorp	--	5	51.12	65.81	35.51
Simple Model-10	ANERcorp	--	10	54.19	67.41	36.91
Simple Model-20	ANERcorp	--	20	57.58	68.78	41.67
Model-A	ANERcorp	Number of distinct Patterns	20	64.08	73.04	54.52
Model-A (PMI)	ANERcorp	Pointwise Mutual Information (PMI)	20	58.77	69.22	48.45
Model-B	ANERcorp + ACE 2005	Number of distinct Patterns	20	**64.46**	**73.77**	**54.55**

Table 5.7: Parameter settings and results of different SSL models.

We tested whether the difference in performance between the aforementioned trained models (see Table 5.7) is significant using a statistical test over the results of these models on the ANERcorp test set. We ran a non-parametric sign test (at significance level $\alpha = 0.01$). The test results (p-values) for the pairwise comparisons of the trained models can be seen in Table 5.8.

	Simple Model 5	Simple Model 10	Simple Model 20	Model A (PMI)	Model A	Model B
Simple-Model-5		0.00018	4.59E-09	4.51E-16	1.85E-14	1.49E-14
Simple-Model-10			6.54E-06	3.20E-16	1.64E-14	7.16E-12
Simple-Model-20				4.62E-16	4.17E-10	3.09E-07
Model-A(PMI)					4.61E-08	4.98E-08
Model-A						0.2668
Model-B						

Table 5.8: The sign test results (p-values) for the pairwise comparisons of the SSL models.

Table 5.9 shows the results of applying our SSL algorithm on the ANERcorp test set in order to extract new types of NE. Table 5.10 compares the performance of our SSL algorithm when extracting standard NEs with its performance when extracting specialised NEs.

POL										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	81.29	82.33	83.44	83.47	83.47	83.47	83.54	83.54	82.87	82.87
Recall	39.51	43.04	49.6	49.61	49.61	49.61	50.56	50.56	50.72	50.72
F-measure	53.17	56.53	62.22	62.23	62.23	62.23	62.99	62.99	62.93	62.93
SPORT										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	87.64	86.42	86.42	86.42	86.42	86.42	86.56	86.56	86.56	86.56
Recall	40.35	41.94	41.94	41.94	41.94	41.94	42.14	42.14	42.14	42.14
F-measure	55.26	56.47	56.47	56.47	56.47	56.47	56.68	56.68	56.68	56.68
ART										
	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5	Iter. 6	Iter. 7	Iter. 8	Iter. 9	Iter. 10
Precision	86.73	84.78	82.25	82.25	82.25	82.00	82.00	82.00	82.00	82.00
Recall	41.42	43.28	44.29	44.29	44.29	47.14	47.14	47.14	47.14	47.14
F-measure	56.06	57.31	57.58	57.58	57.58	59.86	59.86	59.86	59.86	59.86

Table 5.9: The results of the SSL algorithm when detecting specialised NEs.

		Precision	**Recall**	**F-measure**
Specialised NEs	**POL**	82.87	50.72	62.93
	SPORT	86.56	42.14	56.68
	ART	82.00	47.14	59.86
Classical NEs	**PER**	85.91	51.10	64.08
	LOC	87.91	62.48	73.04
	ORG	84.27	40.30	54.52

Table 5.10: The results of the SSL algorithm when detecting standard NEs & specialised NEs.

We compared our SSL algorithm with three different supervised NER classifiers by testing them on the ANERcorp test set. The three supervised systems are ANERsys 1.0 [Benajiba *et al.*, 2007], ANERsys 2.0 [Benajiba and Rosso, 2007], and a CRF-based classi-

fier [Benajiba and Rosso, 2008] (see Section 4.5.2 for more information about these NER classifiers). Table 5.11 shows the results of the three different supervised classifiers and our SSL algorithm.

		ANERsys 1.0	**ANERsys 2.0**	**CRF-based System**	**Our SSL algorithm**
PER	Precision	54.21	56.27	80.41	**85.91**
	Recall	41.01	48.56	**67.42**	51.10
	F-measure	46.69	52.13	**73.35**	64.08
LOC	Precision	82.17	91.69	**93.03**	87.91
	Recall	78.42	82.23	**86.67**	62.48
	F-measure	80.25	86.71	**89.74**	73.04
ORG	Precision	45.16	47.95	84.23	**84.27**
	Recall	31.04	45.02	**53.94**	40.30
	F-measure	36.79	46.43	**65.76**	54.52

Table 5.11: The comparison between our SSL algorithm and three different supervised classifiers.

5.4 Results Discussion

We started our experiments by building three simple models that begin with 5, 10, or 20 seed instances of the NE type of interest. These three baselines follow the steps of Algorithm 5.1, but apply only a single iteration. Since the number of iterations is one, no ranking measure is required to select the best candidate NEs as seed examples for the next iterations. The results in Table 5.2 show generally high precision of the baselines, although recall is low. This indicates the sensible representation of patterns that we identified, and that of the filter algorithm that we applied. Increasing the number of initial seed instances affected the precision negatively, but had a positive impact on the recall, as expected. We wanted, however, to keep the number of initial seed instances as low as possible. This is in line with other studies that applied semi-supervised methods to collect lists of NEs from the web, such as [Etzioni *et al.*, 2005] and [Nadeau *et al.*, 2006] who used 4 and 15 seed examples, respectively. Therefore, we decided to start with 20 seed examples and add more

seed examples through iterations.

In the next experiment we iterated our trained models through steps 1 to 7 in Algorithm 5.1. When using the validation set, we recognised that increasing the number of iterations to more than ten loops does little to improve the performance of the system (F-measure improvement <0.01). Therefore, we set the number of iterations to ten. The best extracted entities of each iteration were used as seed examples for the next iteration. To select the most reliable extracted entities from each iteration, we built two models that use two different ranking measures: (a) Model-A ranked candidate NEs according to the number of distinct patterns used to extract them, and (b) Model-A(PMI) used PMI as a ranking measure. The results obtained using PMI as a measure to select the seed instances for the next iteration demonstrated generally lower performance and particularly lower recall (see Table 5.4). These results can be attributed to PMI's bias towards infrequent words [Turney *et al.*, 2010], resulting in the extraction of fewer patterns for the next iteration. Using PMI, the precision was not affected at all, since very few patterns were added to set *Final_P* in each iteration. In general, PMI results in a lower performance than that achieved when using the number of distinct patterns as a reliable measure for seed selection (see Table 5.5).

The effect of using a larger training corpus on the performance of the model can be seen in Table 5.6, which shows the performance of Model-B. Model-B's parameters were set similarly to Model-A's, but it was trained on a larger corpora (a combination of ANERcorp and ACE 2005). Using large training data increased the recall of the trained model with a small negative effect on precision. The sign test we used to statistically conduct pairwise comparisons of the trained models, however, shows that there is no significant difference between Model-A and Model-B ($p = 0.2668$).

Based on all of our previous experiments, we concluded that the following parameters give the best results: the number of initial seeds is 20, the number of iterations is 10, and the ranking measure to select the most reliable candidate NEs is the number of distinct patterns. For the sake of simplicity, we refer to our system that used the trained models with the previously mentioned parameters as "ASemiNER".

The performance of ASemiNER when applied to specialised NEs is comparable to its performance when extracting standard NEs. The results for the "Politicians" entity show a reasonable increase in recall from iteration to iteration and a good performance overall. For "Sportspersons", the low recall is possibly due to the impact of the lower number of varied contexts in which seeds occur. So, Sportspersons constitute 19% of all persons' names that exist in the training corpus, and they occur in a few contexts. Thus, the diversity of contexts in which seeds appear plays an important role in obtaining a trained model with good performance. The remaining recall errors can presumably be attributed to the diversity of categories. Accordingly, Sportspersons can be broken down into other categories, such as football players, golfers and wrestlers. In contrast, Politician entity recognition has a higher recall than Sportspersons. This can clearly be attributed to two facts: 1) Politicians make up 44% of the persons names in the training corpus, and 2) An efficient model results from using initial seeds like بوش "Bush", which occurs frequently and in a variety of contexts in the training corpus. Overall, our semi-supervised system is easily adaptable when extending the NE hierarchy and performs just as well when recognising the standard Person category.

In comparison with different supervised NER systems [Benajiba and Rosso, 2007, 2008; Benajiba *et al.*, 2007] when applied on the ANERcorp test set, ASemiNER outperforms a sensible supervised system, which depends on a set of simple language-independent features. ASemiNER still cannot compete, however, with more complex supervised systems as seen in Table 5.11. This can be attributed to its relatively low recall which is due to ASemiNER's incapability to detect NEs in ambiguous contexts. For example, detecting برشلونة "Barcelona" as an Organisation name in the following sentence:

"اكد برشلونة ان تشيجرينسكي قد اجتاز الفحوص الطبية و وقع لمدة ٥ سنوات."

"Barcelona confirmed that Chygrynskiy had passed a medical and signed for five years."

ASemiNER also tends to miss NEs that appear in the text isolated from the context. For example, the names of the journalists at the beginning of each news articles usually appear alone attached to the article itself with nothing preceding it. The journalists' names of the articles in Benajiba corpus, in particular, are not, preceded by the words "written by"

and they are followed by different contexts. So, these names are sometimes are followed by punctuation marks, the name of the city that the news is from, or by the news article itself.

5.5 Summary

This chapter presented a semi-supervised algorithm for identifying Named Entities (NEs) in Arabic text without annotated training data, or gazetteers. Our proposed algorithm only requires a seed list of a few instances for each NE type in order to initiate the learning process. Then, it iterates through three phases: 1) pattern induction, 2) instance (candidate NEs) extraction, and 3) instance selection and ranking. Our contributions include the following:

- We proposed a new way to produce the extraction patterns.
- We proposed a new way to generalise and filter the extraction patterns.
- We presented and compared two ranking measures for determining the most reliable candidate NEs: the number of distinct patterns used in extracting candidate NEs, and a measure applying Pointwise Mutual Information (PMI).

In comparison with different supervised NER systems, ASemiNER outperforms sensible supervised systems. It can also be easily adapted to identify new types of NEs and does not generate problems typical of supervised methods, which usually require annotated training data and demand more effort and time to extract specialised types of NEs.

In general, our proposed algorithm uses careful criteria when selecting an instance from candidate NEs for the next iteration, thereby having high precision at the expense of recall. The low recall can also be attributed to some NEs that appear in ambiguous contexts and those NEs that are mentioned in the corpus isolated from the context. This makes it difficult for the algorithm to induce a valid pattern from the contexts.

In the next chapter, we will look at another method of minimally-supervised approaches to NER in order to handle NEs that can not be easily detected by the SSL algorithm.

Chapter 6

Distant Learning

In the previous chapter we presented a semi-supervised learning technique for Arabic NER in order to detect NEs in the text with minimal human intervention. We demonstrated how easily our semi-supervised approach adapts to identify new types of NEs. The proposed algorithm, however, showed limitations when recognising some NEs that appear in ambiguous contexts.

In this chapter we present distant learning as an alternative that requires limited human intervention and avoids manually annotated data. The distant learning approach to NER exploits the high coverage and rich informational structure of online encyclopedias, like Wikipedia, in order to automatically create an NE annotated corpus.

This chapter presents our methodology for exploiting Wikipedia's structure in order to automatically develop an Arabic NE annotated corpus. Each Wikipedia link is transformed into an NE type of the target article in order to produce the NE annotation. Other Wikipedia features - redirects, anchor texts, and inter-language links - are used to tag additional NEs, which appear without links in Wikipedia texts. We propose a filtering algorithm to eliminate ambiguity when tagging candidate NEs. We also introduce a mechanism based on the high coverage of Wikipedia in order to address two challenges particular to tagging NEs in Arabic text: rich morphology and the absence of capitalisation. Section 6.1 outlines the types of Wikipedia articles and the structural information about these articles. Section 6.2 summarises the proposed methodology. Sections 6.3, 6.4, and 6.5 describe the proposed algorithm in detail. The manually annotated data used for the evaluation, the experimental setup, and the results are reported in Section 6.7. The results are discussed in Section 6.8.

We draw some conclusions in Section 6.9.

6.1 Structure of Wikipedia

Wikipedia is a free online encyclopedia project written collaboratively by thousands of volunteers, using MediaWiki[1]. Each article in Wikipedia is uniquely identified by its title. The title is usually the most common name for the entity explained in the article. For example, 'United Kingdom' is the title of the Wikipedia article describing Britain.

6.1.1 Types of Wikipedia Pages

6.1.1.1 Content Pages

Content pages (i.e., Wikipedia articles) contain the majority of Wikipedia's informative content. Each content page describes a single topic and has a unique title. In addition to the text describing the topic of the article, content pages may contain tables, images, links, and templates.

6.1.1.2 Redirect Pages

A redirect page is used if there are two or more alternative names that can refer to one entity in Wikipedia. Thus, each alternative name is used as a title whose article contains a redirect link to the actual article for that entity. For example, 'UK' is an alternative name for the 'United Kingdom', so, the article with the title 'UK' simply links to the article with the title 'United Kingdom'.

6.1.1.3 Disambiguation Pages

A Disambiguation page is created for an ambiguous name that denotes two or more entities in Wikipedia. For example, the disambiguation page for the Arabic name زهرة *zuhrah* "Venus" enumerates 5 titles related to multiple entities such as مركز الزهرة (شركة دبلجة) "Venus Centre (dubbing company)" and الزهرة (كوكب) "Venus (planet)".

[1] An open source wiki package written in PHP.

6.1.1.4 Lists Pages

Wikipedia offers several ways to group articles. One method is to group articles by lists. The items on these lists include links to articles in a particular subject area, and may include additional information about the listed items. For example, 'list of scientists' contains links to articles of scientists and links to more specific lists of scientists, such as a list of Brazilian scientists and a list of British scientists.

6.1.2 Structure of Wikipedia Articles

6.1.2.1 Categories

According to Wikipedia's guidelines[2], every article in the Wikipedia collection should fall under at least one category. Categories should be on vital topics that are useful to the reader. For example, the Wikipedia article about the United Kingdom in Wikipedia is associated with a set of categories that includes 'Countries bordering the Atlantic Ocean', 'Countries in Europe', and 'English-speaking countries and territories'. Categories allow Wikipedia articles to be associated with more than one topic.

6.1.2.2 Infobox

An infobox is a fixed-format table added to the top right-hand or left-hand corner of articles to provide a summary of some unifying parameters shared by the articles. Infobox templates contain important facts and parameters that related articles have in common. For example, every scientist has a name, date of birth, birthplace, nationality, and field of study.

6.1.3 Links

A link is a method used by Wikipedia to link pages within wiki environments. Links are enclosed in doubled square brackets. A vertical bar, the 'pipe' symbol, is used to create a link while labelling it with a different name on the current page. Look at the following two examples:

[2]Wikipedia's guidelines are available at `http://en.wikipedia.org/wiki/Category:Wikipedia_guidelines`.

1 - [[a]] is labelled 'a' on the current page and links to taget page 'a',

2 - [[$a|b$]] is labelled 'b' on the current page, but links to target page 'a'.

In the second example, the *anchor text* (i.e., *link label*) is 'b', while 'a', a *link target*, refers to the title of the target article. In the first example, the anchor text shown on the page and the title of the target article are the same.

6.2 Creating NE Corpora by Leveraging Wikipedia Structure and Features

In order to transform Wikipedia into an NE annoted corpus, we exploited many of Wikipedia's concepts such as links, anchor texts, and redirects [Mika *et al.*, 2008; Nothman *et al.*, 2008, 2013]. Our approach to automatically developing an annotated NE corpus led us to adopt the following steps:

1. Classify Wikipedia articles into a specific set of NE types.
2. Annotate the Wikipedia text as follows:
    - Identify matching phrases in the title and the first sentence of each article and label the matching phrases according to the article type.
    - Label linked phrases in the text according to the NE type of the target article.
    - Compile a list of alternative titles for articles and filter out ambiguous ones.
    - Identify and label matching phrases in the list and the Wikipedia text.
3. Filter sentences to prevent noisy sentences being included in the corpus.

All of our experiments were conducted on the 26 March 2013 Arabic version of the Wikipedia dump[3]. A parser was created to handle the mediawiki markup and to extract structural information from the Wikipedia dump, such as a list of redirect pages along with their target articles, a list of pairs containing link labels and their target articles in the form '*anchor text, target article*', and essential information for each article (e.g., title, body text, categories, and templates).

[3]Arabic Wikipedia dump is available at `http://dumps.wikimedia.org/arwiki/`.

6.3 Classifying Wikipedia Articles into NE Types

Categorising Wikipedia articles is the initial step in producing NE training data. Therefore, all Wikipedia articles need to be classified into a specific set of NE types.

6.3.1 Data set and Annotation

In order to develop a Wikipedia document classifier, we used a set of 4,000 manually classified Wikipedia articles[4]. The set was manually classified using the ACE (2008) taxonomy and a new class (Product). In total, there were eight coarse-grained categories: Facility, Geo-Political, Location, Organisation, Person, Vehicle, Weapon, and Product. As our work adheres to the CoNLL definition, we mapped these classified Wikipedia articles into CoNLL NE types – Person, Location, Organisation, Miscellaneous, or Other – based on the CoNLL 2003 annotation guidelines [Tjong Kim Sang and De Meulder, 2003](see Appendix B for detailed list of CoNLL-2003 NE types with associated categories of names).

6.3.2 Classification of Wikipedia Articles

Many researchers have already addressed the task of classifying Wikipedia articles into named entity types [Dakka and Cucerzan, 2008; Tardif *et al.*, 2009]. Up to our knowledge, only Alotaibi and Lee [2012] have experimented with classifying the Arabic version of Wikipedia into NE classes (see Section 4.5.2).

We conducted three experiments that used simple bag-of-words features extracted from different portions of the Wikipedia document and metadata. We included the following portions of the document in each experiment:

Experiment 1: It involved tokens from the article title and the entire article body.

Experiment 2: Rich metadata in Wikipedia proved effective for the classification of articles [Alotaibi and Lee, 2012; Tardif *et al.*, 2009]. Therefore, in Experiment 2 we included tokens from categories, tokens from the article title and the first sentence of the document, and templates - specifically 'Infobox'.

[4]The set of Arabic Wikipedia articles manually classified is available at `http://www.arabic-ner.com/`.

Experiment 3: It involved the same set of tokens as Experiment 2, except that categories and infobox features were marked with suffixes to differentiate them from tokens extracted from the article body text. We incorporated this step of distinguishing tokens based on their locations in the document, because it has been reported to improve the accuracy of document classification [Alotaibi and Lee, 2012; Tardif *et al.*, 2009].

In order to optimise features, we implemented a filtered version of the bag-of-words article representation (e.g., removing punctuation marks and symbols) to classify the Arabic Wikipedia documents instead of using a raw data set. In addition, Alotaibi and Lee [2012] demonstrated the great impact of applying clitic tokenisation as opposed to the neutral effect of stemming. So, we implemented the filtered features proposed in the study of Alotaibi and Lee [2012], which included removing punctuation marks and symbols, filtering stop words, and normalising digits. We extended the features, however, by utilising the clitic tokenisation scheme that involves separating conjunctions, prepositions, and pronouns from each word.

The feature set was represented using the Term Frequency-Inverse Document Frequency ($TF - IDF$) method, a numerical statistic that reflects how important a token is to a document.

6.3.3 Results of Classifying Wikipedia Articles

As for the learning process, our Wikipedia documents classifier was trained using Liblinear[5]. 80% of the 4,000 manually classified Wikipedia articles were dedicated to the training stage, while 20% were specified to test the classifier. Table 6.1 is a comparison of the precision, recall, and F-measure of the classifiers that resulted from the three aforementioned experiments. The Wikipedia document classifier in Experiment 3 performed better than the other classifiers. Therefore, we selected it to classify all of the Wikipedia articles. At the end of this stage, we obtained a list of pairs containing each Wikipedia article and its NE Type. We stored this list in a database in preparation for the next stage: developing the NE-tagged training corpus.

[5]Liblinear is available at `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

		PER	LOC	ORG	MISC	NON	Overall
Experiment 1	Precision	73.10	67.90	60.12	58.19	65.88	65.04
	Recall	60.07	68.87	61.77	52.89	55.77	59.87
	F-measure	65.95	68.38	60.93	55.41	60.40	62.35
Experiment 2	Precision	92.11	82.23	88.97	86.03	83.06	86.48
	Recall	86.19	90.14	89.67	88.61	87.55	88.43
	F-measure	89.05	86.00	89.32	87.30	85.24	87.45
Experiment 3	Precision	93.88	86.77	89.21	87.93	86.45	88.85
	Recall	95.10	92.14	90.63	91.08	88.01	91.39
	F-measure	**94.49**	**89.37**	**89.91**	**89.47**	**87.22**	**90.10**

Table 6.1: The results of the three Wikipedia document classifiers.

6.4 Annotation Process

After classifying Wikipedia articles into a set of NE types, each Wikipedia link in the article text is transformed into an NE type of the target article in order to produce the NE annotation. Other Wikipedia features - redirects, anchor texts, and inter-language links - are exploited to tag additional NEs, which appear without links in Wikipedia texts. Next, we explain the steps that our methodology followed to produce the NE annotated corpus from Wikipedia in detail.

6.4.1 Utilising Titles of Articles and Link Targets

To identify corresponding words in the article title and the entire body of text and then to tag the matching phrases with the NE type can be a risky process, especially for terms with more than one meaning. For example, the title of the article describing the city كان *kaan* "Cannes" can also, in Arabic, refer to the past verb كان *kaan* "was". The portion of the Wikipedia article unlikely to produce errors during the matching process is the first sentence, which usually contains the definition of the term the Wikipedia article is written about [Zesch *et al.*, 2008].

To successfully complete this step, it is necessary to be aware that article titles often

contain abbreviations or shortened names (e.g., personal first or last name instead of full name), while the article's first sentence is usually made up of unabbreviated and full names. This pattern makes it difficult to identify matching terms in the title and first sentence, and frequently appears in biographical Wikipedia articles. For example, one article is entitled ابو بكر الرازي "Abu Bakr Al-Razi", but the first sentence states the full name of the person: ابو بكر محمد بن يحيى بن زكريا الرازي "Abu Bakr Mohammad Bin Yahia Bin Zakaria Al-Razi". In our estimation, the best solution to this problem is partial matching. In this case, the system should first identify all corresponding words in the title and the first sentence. Second, the system should annotate them and all words that fall between, provided that:

- the sequence of the words in the article title and the text are the same in order to avoid errors in tagging. For example, if the title of the article is نهر التايمز "The River Thames", but the first sentence reads .... التايمز هو نهر يقع في "The Thames is a river flowing through southern England....", then the text will not be properly tagged.
- the number of tokens located between matched tokens is less than or equal to five[6].

Figure 6.1 shows one example of partial matching.

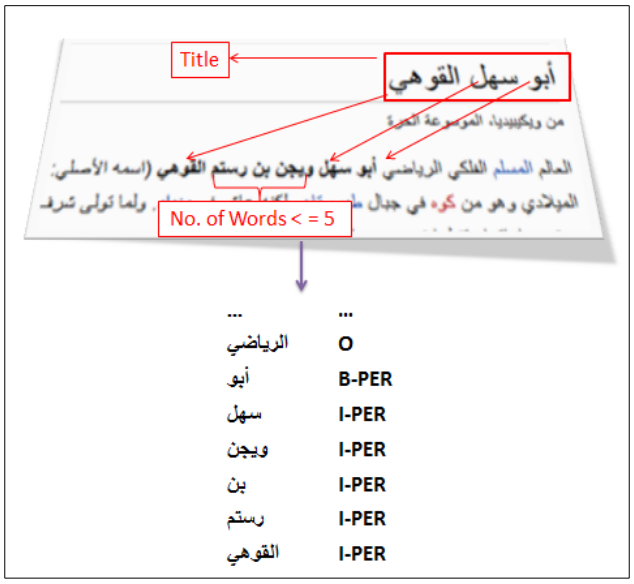


Figure 6.1: An example of the partial matching.

[6]The statistics about the length of proper nouns we derived from two Arabic NER corpora (ANERcorp and AQMAR) showed that the longest proper Arabic names are 5 to 7 tokens in length (see Table 5.1).

The next step is to transform the links between Wikipedia articles into NE annotations according to the link target type. Therefore, the link ([[اوباما|باراك اوباما]]/[[Barack Obama|Obama]]) would be changed to (اوباما PER) (Obama PER), since the link target (Barack Obama) is the title of an article about a person. By the end of this stage, all NE anchor texts (anchor texts referring to NE articles) on Wikipedia should be annotated based on the NE type of the target article.

6.4.2 Dictionaries of Alternative Names

Depending only on NE anchor texts in order to derive and annotate data from Wikipedia results in a low-quality data set, as Wikipedia contains a fair amount of NEs that are mentioned without links. This is due to the fact that each term on Wikipedia is more likely to be linked on its first appearance in the article [Nothman *et al.*, 2008]. These unlinked NE phrases can be found simply by identifying the matching terms in the list of article titles and the text. The process is not as straightforward as it seems, however, because identifying corresponding terms may prove ineffective, especially in the case of a morphologically rich language in which unlinked NE phrases are sometimes found agglutinated to prefixes and conjunctions. In order to detect unlinked and inflected forms of NEs in Wikipedia text, we extended the list of article titles that were used in the previous step to find and match the possible NEs by including NE anchor texts. Adding NE anchor texts to the list assists in finding NEs in the text that may be morphologically inflected, while eliminating the need for any morphological analysis. Table 6.2 shows examples from the dictionary of NE anchor texts.

Anchor Texts	English Gloss
والمغرب	and Morocco
بالمغرب	in Morocco
كالمغرب	such as Morocco
للمغرب	to Morocco
وكالمغرب	and such as Morocco

Table 6.2: Examples from the dictionary of NE anchor texts.

Spelling variations resulting from the varied transliteration of foreign NEs in some cases prevent the accurate matching and identification of unlinked NEs, if only the list of NE anchor texts is used. For example, انجلترا "England" has been written in Arabic Wikipedia in five different ways: انجلتره, انغلترا, انغلتره, انكلترا, انكلتره. The best resolution to this issue is to compile a list of redirected page titles that send the reader to articles describing NEs. We refer to these titles as *NE redirects*. We call the list of NE redirects and anchor texts *a list of alternative names*, since they can be used as alternative names for article titles.

The list of alternative names is used to find unlinked NEs in the text by matching phrases from the list with identical terms in the article text. This list is essential for managing spelling and morphological variations of unlinked NEs, as well as misspellings. Consequently, the process increases the coverage of NE tags augmented within the plain texts of Wikipedia articles.

6.4.2.1 Filtering the Dictionaries of Alternative Names

Identifying matching phrases in the list of alternative names and the text inevitably results in a lower quality corpus due to noisy names. The noisy alternative names usually occur with meaningful NEs. For example, the article on the person ابو عبدالله الامين "Abu Abdullah Alamyn" has an alternative name consisting only of his last name الامين *alamyn*, which means "custodian" or "secretary". Therefore, annotating every occurrence of الامين *alamyn* as PER would lead to incorrect tagging and ambiguity. The same applies to the city with the name الجديده *aljadydah*, which literally means "new". Thus, the list of alternative names should be filtered to remove such noisy alternative names.

Capitalisation probability measure for Arabic words: We introduce a capitalisation probability measure for Arabic words, which are never capitalised, in order to omit one-word NE phrases that are ambiguous when taken out of context because they usually have an interpretation as a common noun or adjective in Arabic. This require finding the English gloss for each one-word alternative name and then computing its probability of being capitalised using the English Wikipedia. To find the English gloss for Arabic words, we exploit Wikipedia Arabic-to-English cross-lingual links that provide us with a reasonable number

of Arabic and corresponding English terms. If the English gloss for the Arabic word could not be found using inter-language links, we resort to an online translator. Before translating the Arabic word, a light stemmer is used to remove prefixes and conjunctions in order to get the translation of the word itself without its associated affixes. For example, the English translation of the Arabic word كلندن *kalandan* is "such as London". So, in order to get the translation of the main word itself, لندن *landan* "London", we used a light stemmer to remove affixes. The capitalisation probability is computed as follows:

$$Pr[EN] = \frac{f(EN)_{isCapitalised}}{f(EN)_{isCapitalised} + f(EN)_{notCapitalised}}$$

where: EN is the English gloss of the alternative name; $f(EN)_{isCapitalised}$ is the number of times the English gloss EN is capitalised in English Wikipedia; and $f(EN)_{notCapitalised}$ is the number of times the English gloss EN is not capitalised in English Wikipedia. This way, we build a list of Arabic words along with their probabilities of being capitalised.

It is evident that one-word NEs, that have an interpreation as a common noun, adjective, or verb in Arabic, usually achieve a low probability. By specifying a capitalisation threshold constraint, we prevent such words from being included in the list of alternative names. After a set of experiments, we decided to set the capitalisation threshold equal to 0.75.

Contrastingly, multi-word alternative names (e.g., مصطفى محمود "MusTafae Mahmud", احمد عادل "Ahmad Adel") rarely cause errors in the automatic annotation process. Wikipedians, however, at times append personal and job titles to the person's name contained in the anchor text, which refers to the article about that person. Examples of such anchor texts are رئيس مجلس الوزراء محمد بن راشد "President of the Council of Ministers Muhammad bin Rashid" and حاكم دبي محمد بن راشد "Ruler of Dubai Muhammad bin Rashid". As a result, the system mistakenly annotates words like *Dubai, Council,* and *Ministers* as PER.

To solve this problem, we omitted the multi-word alternative name if any of its words belonged to the list of apposition words which usually appear adjacent to NEs such as رئيس "President", وزير "Minister", and حاكم "Ruler". The filtering algorithm managed to exclude 22.95% of the alternative names from the original list. Algorithm 6.1 shows pseudo code of the filtering algorithm.

Algorithm 6.1: Filtering alternative names algorithm.

Input: A set $L = \{l_1, l_2, \ldots, l_n\}$ of all alternative names of Wikipedia articles
Output: A set $RL = \{rl_1, rl_2, \ldots, rl_n\}$ of reliable alternative names

```
for i ← 1 to n do
    T ← split l_i into tokens
    if (T.size() >= 2) then
        /* All tokens of T do not belong to apposition list          */
        if (! containAppositiveWord(T)) then
            add l_i to the set RL
    else
        light_stem ← findLightStem(l_i)
        english_gloss ← translate(light_stem)
        /* Compute Capitalisation Probability for English gloss      */
        cap_prob ← compCapProb(english_gloss)
        if (cap_prob > 0.75) then
            add l_i to the set RL
```

Some statistics about the dictionaries derived from Wikipedia by exploiting Wikipedia's structure and adopting the filtering algorithm are reported in Table 6.3.

Dictionary	Number of Entries
Redirects	182,808
• List of NE *Redirects*	94,606
• Filtered list of NE *Redirects*	74,073
Anchor Texts	689,171
• List of NE *Anchor Texts*	130,692
• Filtered list of NE *Anchor Texts*	99,512

Table 6.3: Dictionaries derived from Wikipedia.

6.4.3 Post-processing

The goal of post-processing was to address some issues that arose during the annotation process as a result of different domains, genres, and conventions of entity types. For example, nationalities and other adjectival forms of nations, religions, and ethnic groups are considered MISC in the CoNLL NER task in the English corpus, while the Spanish corpus consider them *NOT* named entities [Nothman *et al.*, 2013]. As far as we know, almost

all Arabic NER data sets that followed the CoNLL guidelines in the annotation process consider nationalities *NOT* named entities. We found that there are 1521 nationalities in ANERcorp corpus and they are considered NOT NEs. In Wikipedia all nationalities are linked to articles about the corresponding countries, which makes our methodology tag them as LOC. We decided to consider them *NOT* named entities in accordance with the CoNLL-style Arabic data sets. Therefore, in order to resolve this issue, we compiled a list of nationalities, and other adjectival forms of religion and ethnic groups, so that any anchor text matching an entry in the list was retagged as a *NOT* named entity.

The list of nationalities and apposition words used in Section 6.4.2.1 were compiled by exploiting the 'List of' articles in Wikipedia such as *list of people by nationality*, *list of ethnic groups*, *list of adjectival forms of place names*, and *list of titles*. Some English versions of these 'List of' pages have been translated into Arabic, either because they are more comprehensive than the Arabic version, or because there is no corresponding page in Arabic.

6.5 Wikipedia-derived Corpus (WDC)

After the annotation process was completed, the last step was to incorporate sentences into the corpus. As a result, we obtained an annotated data set made up of approximately ten million tokens. However, in order to obtain a corpus with a large number of tags without affecting its quality, we created a data set called Wikipedia-derived corpus (*WDC*), which included only sentences with at least three annotated named entity tokens.

The *WDC* data set contains 165,119 sentences consisting of around 6 million tokens. The annotation style of the *WDC* data set followed the CoNLL format, where each token and its tag are placed together in the same file in the form $< token > \backslash s < tag >$. The NE boundary is specified using the *IOB2* representation scheme, where *B-* indicates the beginning of the NE, *I-* refers to the continuation (Inside) of the NE, and *O* indicates that the word is not a NE. The *WDC* data set is available online to the community of researchers[7].

[7]The WDC data set is available at `https://sites.google.com/site/mahajalthobaiti/`.

6.6 Data

To evaluate the quality of our methodology, we used *WDC* as training data to build an NE classifier. Then, we tested the resulting classifier on data sets from different domains. We used three data sets: ANERcorp, NEWS, and TWEETS.

ANERcorp, which is freely available for research purposes, contains approximately 150K tokens (11% of the tokens are NEs). It is composed of a training set and a test set built and tagged especially for the NER task by Benajiba *et al.* [2007]. More details about ANERcorp are found in Section 4.2.

The NEWS data set contains news snippets from the RSS feed of the Arabic version of news.google.com from October 2012. The RSS consists of the headline and the first 50 to 100 words in the news articles. This set contains approximately 15K tokens. The third test set (TWEETS) contains a set of 1,423 tweets authored between 23rd of November 2011 and 27th of November 2011. It has approximately 26K tokens [Darwish, 2013].

6.7 Experiments and Results

6.7.1 Experimental Setup

All experiments to train and build the classifiers were conducted using Conditional Random Fields (CRFs)[8]. Regarding the features used in all our experiments, we used the most successful features reported in Arabic NER work [Abdul-Hamid and Darwish, 2010; Benajiba *et al.*, 2008b; Darwish, 2013]. These features include:

- The words immediately before and after the current word in their raw and stemmed forms.
- The first 1, 2, 3, 4 characters in a word.
- The last 1, 2, 3, 4 characters in a word.
- The appearance of the word in the gazetteer.

[8]We used CRFsuite which is available at `http://www.chokkan.org/software/crfsuite/`.

- The stemmed form of the word.

We used ANERgazet, the gazetteer developed by Benajiba *et al.* [2008b] (see Section 4.2). A light stemmer was used to determine the stemmed form of the word by using simple rules to remove conjunctions, prepositions, and definite articles [Larkey *et al.*, 2002].

We built an NE classifier by training a classifier on the automatically NE annotated corpus from Wikipedia (*WDC*) (see Section 6.2). We named that classifier the Distant Learning (*DL*) classifier. We also built another NE classifier by training it on manually annotated data (ANERcorp training set). We called that classifier the *gold-standard classifier*. These two classifiers used the same aforementioned features.

We compared the DL classifier and gold-standard classifier by evaluating them on data sets from different domains. Firstly, we decided to test them on Wikipedia. Thus, a subset, containing around 14k tokens, of the *WDC* set was allocated for testing purpose. It was checked and annotated manually by one native speaker of Arabic according to the CoNLL tagging guidelines. Secondly, the gold-standard classifier and DL classifier were tested on the ANERcorp test data. The objective of this comparison was to show how well the DL classifier works on a newswire domain, which is more specific than Wikipedia's open domain. Thirdly, the gold-standard classifier and the DL classifier were tested on NEWS corpus, which is also a newswire based data set. Finally, we tested the gold-standard classifier and DL classifier on data extracted from a social network, namely Twitter (TWEETS corpus).

We also compared our approach with other approaches for automatically producing NE annotations from Arabic Wikipedia. So, our DL classifier and other two classifiers trained on two corpora automatically annotated from Arabic Wikipedia ($\text{WikiFANE}_{\text{Whole}}$ and $\text{WikiFANE}_{\text{Selective}}$) were evaluated on the ANERcorp test set (see Section 4.2 for more details about WikiFANE data sets).

In order to show the effect of combining our corpus (*WDC*) with a manually annotated data set from a different domain, we merged *WDC* with the ANERcorp training set. Then, we trained a classifier on the combined corpus and tested its performance on the four aforementioned data sets: ANERcorp test set, NEWS set, Wikipedia set, and TWEETS set. We refer to that classifier as the *combined data classifier*.

6.7.2 Results

We compared the NE classifier (gold-standard classifier) that we trained on the manually annotated data (ANERcorp training set) with the state-of-the-art NE classifier trained on the same data set by Darwish [2013]. We refer to his NE classifier as the *state-of-the-art classifier*. Table 6.4 compares the performance of the state-of-the-art classifier with the performance of our gold-standard classifier.

	Gold-standard classifier			**State-of-the-art classifier**		
	Precision	Recall	F-measure	Precision	Recall	F-measure
PER	88.20	69.70	77.87	87.00	77.70	**82.09**
LOC	94.07	80.90	86.99	92.30	87.80	**89.99**
ORG	84.20	58.70	69.17	81.40	66.00	**72.90**

Table 6.4: The comparison between our gold-standard classifier and the state-of-the-art classifier.

Table 6.5 shows the results of our DL classifier and the results of the two classifiers trained on WikiFANE data sets, which were automatically collected from Wikipedia by Alotaibi and Lee [2013].

NER Classifier	Precision	Recall	F-measur
$\text{WikiFANE}_{\text{Whole}}$ Classifier	81.53	43.10	56.39
$\text{WikiFANE}_{\text{Selective}}$ Classifier	88.10	37.52	52.63
DL Classifier	76.44	56.42	**64.92**

Table 6.5: The comparison between the DL classifier and the two WikiFANE classifiers

Table 6.6 shows the results of the comparison of the DL classifier and gold-standard classifier when tested on four different data sets: Wikipedia data set, ANERcorp test set, NEWS, and TWEETS corpora. The detailed results of each classifier on different data sets can be found in Table 6.7, Table 6.8, Table 6.9, and Table 6.10.

		Gold-standard Classifier	**DL Classifier**
Wikipedia set	PER	41.57	86.40
	LOC	43.06	79.36
	ORG	20.59	86.46
	Overall	35.41	**84.09**
ANERcorp set	PER	77.87	57.69
	LOC	86.99	70.95
	ORG	69.17	64.45
	Overall	**78.15**	64.92
NEWS set	PER	57.80	56.26
	LOC	65.17	60.78
	ORG	35.23	31.01
	Overall	**53.74**	50.12
TWEETS set	PER	34.57	41.43
	LOC	40.47	39.67
	ORG	15.10	24.36
	Overall	30.99	**35.78**

Table 6.6: The comparison between the gold-standard classifier and DL classifier in terms of F-measure on different data sets.

	Gold-standard Classifier			**DL Classifier**		
	Precision	Recall	F-measure	Precision	Recall	F-measure
PER	61.05	31.52	41.57	86.82	85.99	86.40
LOC	69.55	31.19	43.06	78.64	80.11	79.37
ORG	45.86	13.28	20.59	87.89	85.08	86.46
Overall	58.82	25.33	**35.41**	84.45	83.73	**84.09**

Table 6.7: The results of the gold-standard and DL classifiers on the Wikipedia test set.

	Gold-standard Classifier			**DL Classifier**		
	Precision	Recall	F-measure	Precision	Recall	F-measure
PER	88.2	69.7	77.87	80.01	45.11	57.69
LOC	94.07	80.9	86.99	75.21	67.14	70.95
ORG	84.2	58.7	69.17	74.1	57.02	64.45
Overall	88.82	69.77	**78.15**	76.44	56.42	**64.92**

Table 6.8: The results of the gold-standard and DL classifiers on the ANERcorp test set.

	Gold-standard Classifier			**DL Classifier**		
	Precision	Recall	F-measure	Precision	Recall	F-measure
PER	74.80	47.10	57.80	78.28	44.35	56.62
LOC	84.10	53.20	65.17	86.19	46.49	60.40
ORG	73.20	23.20	35.23	69.10	20.02	31.05
Overall	77.37	41.17	**53.74**	77.86	36.95	50.12

Table 6.9: The results of the gold-standard and DL classifiers on the NEWS data set.

	Gold-standard Classifier			**DL Classifier**		
	Precision	Recall	F-measure	Precision	Recall	F-measure
PER	45.70	27.8	34.57	53.10	34.12	41.54
LOC	79.90	27.10	40.47	67.01	27.70	39.20
ORG	44.40	9.10	15.10	51.23	16.34	24.78
Overall	56.67	21.33	30.99	57.11	26.05	**35.78**

Table 6.10: The results of the gold-standard and DL classifiers on the TWEETS data set.

Table 6.11 shows the results of the combined data classifier when tested on the three test sets.

	ANERcorp + WDC		
	Precision	Recall	F-measure
Wiki	89.21	61.01	72.46
ANERcorp	86.06	62.33	72.30
NEWS	79.67	39.01	52.37
TWEETS	58.33	26.00	35.97

Table 6.11: The results of combining WDC with the ANERcorp data set.

The difference in performance of the three aforementioned classifiers - the DL classifier, gold-standard classifier, and combined data classifier - was tested using a statistical test. We ran a sign test over the results of these three classifiers on different data sets: the ANERcorp test set, NEWS data set, TWEETS data set, and Wikipedia test set. The alpha level ($\alpha = 0.01$) was used as a significance criterion for the statistical test. Table 6.12 shows the p-values for the pairwise comparisons of the three classifiers.

	ANERcorp set	Wiki set	NEWS set	TWEETS set
DL classifier vs. Gold-standard classifier	< 2.2E-16	< 2.2E-16	< 2.2E-16	1.34E-06
DL classifier vs. Combined data classifier	< 2.2E-16	< 2.2E-16	< 2.2E-16	**0.2416**
Gold-standard classifier vs. Combined data classifier	< 2.2E-16	< 2.2E-16	1.45E-05	< 2.2E-16

Table 6.12: The sign test results (p-values) for the pairwise comparisons of the gold-standard classifier and DL classifier.

6.8 Results Discussion

The results of the gold-standard classifier when tested on the ANERcorp test set showed high precision and recall, producing an overall F-measure equal to 78.15%. The high performance of the gold-standard classifier was expected, as it was trained on a manually annotated corpus (ANERcorp training set) and tested on data from the same corpus. The gold-standard classifier that we developed appeared to perform on par with the state-of-the-art NE classifier, as seen in Table 6.4.

The DL classifier was built using the same features utilised when building the gold-standard classifier, but it was trained on the (WDC) corpus - the NE annotated data set that we created automatically from Wikipedia. The comparison between the DL classifier and the two classifiers trained on the WikiFANE data sets, which were automatically collected from Wikipedia by Alotaibi and Lee [2013], revealed that our proposed methodology produced a data set that outperformed the two other data sets in terms of recall and F-measure.

The results in Table 6.6 show that our DL classifier outperforms the F-measure of the gold-standard classifier by around 48 percentage points when tested on the Wikipedia set. The sign test in Table 6.12 shows a siginficant difference in performance between the

gold-standard classifier and the DL classifier ($p < 0.01$). The obvious difference in the performance of the two classifiers can be attributed to the difference in annotation convention for different domains. For example, many NE keywords in Arabic Wikipedia, which appear in the text along with NEs (e.g., جامعة "university", مدينة "city", شركة "company"), are usually considered part of NE names. So, the phrase 'Shizuoka Prefecture' that is mentioned in some Arabic Wikipedia articles is considered an entity and linked to an article that talks about Shizuoka, making the system annotate all words in the phrase as NEs as follows: محافظة *B-LOC* شيزوكا *I-LOC* "Shizuoka *B-LOC* Prefecture *I-LOC*". On the other hand, in the ANERcorp corpus, only the word after the NE keyword ولاية "Prefecture" is considered an NE. In addition, although sport facilities (e.g., stadiums) are categorised in Wikipedia as Location, they are not considered entities in the ANERcorp corpus.

The gold-standard classifier outperforms the F-measure of the DL classifier by around 13 points when tested on the ANERcorp test set ($p < 0.01$). This can be attributed to the fact that training and test data sets for the gold-standard classifier are drawn from the same corpus. In addition, 69% of NEs in the test data are existing in the training set [Darwish, 2013].

The results of the gold-standard classifier on the NEWS data set (as seen in Table 6.9) are lower than those on the ANERcorp test set (the overall F-measure is 53.74% on the NEWS data set, while it is 78.15% on the ANERcorp test set). The DL classifier appears to perform as well as the gold-standard classifier on the NEWS set, producing an F-measure equal to 50.12%. In addition, the results of testing the DL classifier and gold-standard classifier on data set from the Twitter prove that models trained on open-domain data sets like Wikipedia perform better on social network text than classifiers trained on domain-specific data sets.

When tested on the ANERcorp test set, NEWS test set, and Wikipedia set, the combined data classifier achieves results that fall between the results of the classifiers trained on each corpus separately (i.e., DL classifier and gold-standard classifier). The results of the combined data classifier when tested on the fourth test set (TWEETS) show little improvement over the performances of the two classifiers trained on each corpus separately.

In fact, the sign test (see Table 6.12) reveals significant difference in performance between the combined data classifier and the other two classifiers - the gold-standard classifier and the DL classifier - over the four data sets. Only the *DL classifier* vs. *combine data classifier* comparison over the TWEETS data set shows absolutely no significant difference in performance. ($p = 0.2416$).

Generally, although the DL classifier does not outperform the gold-standard classifier which is trained and tested on data from the same corpus, it performs on par with the gold-standard classifier on non-corresponding test sets where the training and test sets are not from the same corpus and/or domain. In addition, the DL classifier outperforms the gold-standard classifier on social network and open-domain text such as Wikipedia and Twitter.

6.9 Summary

This chapter presented a methodology that requires minimal time and human intervention to generate an NE annotated corpus from Wikipedia. The corpus created with our method (*WDC*) contains around 6 million tokens representing different genres, as Wikipedia is considered an open domain. The evaluation results show the high quality of the *WDC*. So that, the DL classifier trained on *WDC* can compete with those trained on manually annotated corpora. Furthermore, the data set *WDC* outperforms other NE corpora generated automatically from Arabic Wikipedia by 8 to 12 points in terms of F-measure. Our main contributions include:

- We introduced a capitalisation probability measure for Arabic words, which are never capitalised, using the high coverage of Wikipedia.
- We proposed a mechanism to handle the rich morphology in Arabic, and eliminate the need to perform any deep morphological analysis by exploiting Wikipedia features such as anchor texts and redirects.
- We automatically developed a large automatic NE corpus and made it available online.

The results of our distant learning approach to Arabic NER show high recall but low precision, whereas our semi-supervised method (see Chapter 5) has very high precision but comparatively low recall. These differences in strengths can be exploited to produce better results. Therefore, we plan, in the next chapter, to investigate the differences between the two minimally-supervised methods and to exploit these differences to obtain better results.

Chapter 7

Classifier Combination

In the previous chapters we presented two minimally supervised methods: semi-supervised learning and distant learning. These methods are alternatives to supervised methods and do not require manually annotated data. However, the performance of such methods tends to be lower than that achieved with supervised methods. Semi-supervised methods tend to have very high precision but comparatively low recall, whereas distant learning tends to achieve higher recall but lower precision. This complementarity suggests that better results may be obtained by combining the two types of minimally supervised methods.

In this chapter, we present a novel approach to Arabic NER using a combination of semi-supervised and distant learning techniques. We combine the two minimally supervised methods (explained in Chapter 5 and 6) in order to obtain better results, exploiting their complementary strengths. Various classifier combination schemes are used to combine minimal supervision methods. In particular, we test the recently proposed Independent Bayesian Classifier Combination (IBCC) scheme [Kim and Ghahramani, 2012; Levenberg *et al.*, 2014], and compare it with traditional voting methods for classifier combination. Section 7.1 presents an error analysis that digs deeper into explaining where each classifier complements the other. Section 7.2 supplies a brief description of the classifier combination schemes used in this chapter. The data sets used in the experiments and the evaluation results are discussed in Section 7.3 and Section 7.4 respectively. The results are discussed in Section 7.5. Finally, the conclusions are drawn in Section 7.6.

7.1 The Case for Classifier Combination

We use *SSL* to refer to our semi-supervised classifier and *DL* to refer to our distant learning classifier. Table 7.1 summarises the results of both classifiers when tested on the ANERcorp test set (see Chapter 5 and Chapter 6 for detailed results).

NEs	Classifiers	Precision	Recall	F-measure
PER	SSL	85.91	51.10	64.08
	DL	80.01	45.11	57.69
LOC	SSL	87.91	62.48	73.04
	DL	75.21	67.14	70.95
ORG	SSL	84.27	40.30	54.52
	DL	74.10	57.02	64.45
Overall	SSL	**86.03**	51.29	64.27
	DL	76.44	**56.42**	**64.92**

Table 7.1: The comparison between the SSL classifier and DL classifier.

As is apparent in Table 7.1, the SSL classifier tends to be more precise at the expense of recall. The distant learning technique is lower in precision than the semi-supervised learning technique, but higher in recall. Generally, preference is given to the distant supervision classifier in terms of F-measure.

The two classifiers have different strengths. Our semi-supervised algorithm iterates between pattern extraction and candidate NEs extraction and selection. Only the candidate NEs of which the classifier is most confident are added at each iteration, which results in the high precision. The SSL classifier performs better than distant learning when detecting NEs that appear in *reliable/regular patterns*. Usually these patterns are learned easily during the training phase, either because they contain important NE indicators, also known as trigger words which help in identifying a NE within text, or because they are supported by many reliable candidate NEs. For example, the SSL classifier has a high probability to successfully detect اوباما "Obama" and لويس فان خال "Louis van Gaal" as persons' names in the following sentences:

- صرّح الرئيس اوباما الذي يزور بريطانيا ...

  "President Obama said on a visit to Britain ..."

- قال لويس فان خال مدرب مانشستر يونايتيد ان ...

  "Louis van Gaal the manager of Manchester United said that ..."

The patterns extracted from such sentences in the newswire domain are learned easily during the training phase, as they contain good NE indicators like الرئيس "president" and مدرب "manager".

Our distant learning method relies on Wikipedia structure and links to automatically create NE annotated data. It also depends on Wikipedia features, such as inter-language links and redirects, to handle the rich morphology of Arabic without the need to perform excessive preprocessing steps (e.g., POS tagging, deep morphological analysis), which has a slightly negative effect on the precision of the DL classifier. The recall, however, of the DL classifier is high, covering as many NEs as possible in all possible domains. Therefore, the DL classifier is better than the SSL classifier at detecting NEs that appear in ambiguous contexts (they can be used for different NE types) and with no obvious clues (NE indicators). For example, detecting فيراري "Ferrari" and نوكيا "Nokia" as organisation names in the following sentences:

- تقدم الونسو على سائق رينو، الذي حرم فيراري ...

  "Alonso got ahead of the Renault driver who prevented Ferrari from ... "

- جاء خطاب نوكيا بعد يوم من اعلان اتمام الصفقة

  "Nokia's speech came a day after the completion of the deal"

The strengths and weaknesses of the SSL and DL classifiers indicate that a classifier ensemble may perform better than its individual components.

7.2 Classifier Combination Methods

Classifier combination methods are suitable for utilising the predictions of multiple classifiers to enable higher accuracy classifications. Dietterich [2000b] reviews many methods for constructing ensembles and explains why classifier combination techniques often perform better than any individual classifier. Tulyakov *et al.* [2008] introduce various categories of

classifier combinations according to different criteria, including the type of the classifier's output and the level at which the combinations operate. Several empirical and theoretical studies have been conducted to compare ensemble methods such as boosting, randomisation, and bagging techniques [Bauer and Kohavi, 1999; Dietterich, 2000a; Maclin and Opitz, 1997]. Ghahramani and Kim [2003] explore a general framework for a Bayesian model combination that explicitly models the relationship between each classifier's output and the unknown true label. As such, multiclass Bayesian Classifier Combination (BCC) models are developed to combine predictions of multiple classifiers. Their proposed method for BCC in the machine learning context is derived directly from the method proposed in [Haitovsky *et al.*, 2002] for modelling disagreement between human assessors, which in turn is an extension of [Dawid and Skene, 1979]. Similar studies on modelling data annotation using a variety of methods are presented in [Carpenter, 2008; Cohn and Specia, 2013]. Simpson *et al.* [2013] present a variant of BCC in which they consider the use of a principled approximate Bayesian method, variational Bayes (VB), as an inference technique instead of using Gibbs Sampling. They also alter the model so as to use point values for hyper-parameters, instead of placing exponential hyper-priors over them.

The following sections detail the combination methods used in this chapter to combine the minimally supervised classifiers for Arabic NER.

7.2.1 Voting

Voting is the most common method in classifier combination because of its simplicity and acceptable results [Van Erp *et al.*, 2002; Van Halteren *et al.*, 2001]. Each classifier is allowed to vote for the class of its choice. It is common to take the majority vote, where each individual classifier is given one vote and the class with the highest number of votes is chosen. In the case of a tie, when two or more classes receive the same number of votes, a random selection is taken from among the winning classes. It is useful, however, if individual classifiers are distinguished by their quality. For this purpose, weights are used to encode the importance of each individual classifier [Van Erp *et al.*, 2002].

Equal voting assumes that all classifiers have the same quality [Van Halteren *et al.*, 2001].

Weighted voting, on the other hand, gives more weight to classifiers of better quality. So, each classifier is weighted according to its overall precision, or its precision and recall on the class it suggests.

Formally, given K classifiers, a widely used combination scheme is through the linear interpolation of the classifiers' class probability distribution as follows

$$P(C|\mathrm{S}_1^K(w)) = \sum_{k=1}^{K} P_k(C|S_k(w)) \cdot \lambda_k(w) \tag{7.1}$$

where $P_k(C|S_k(w))$ is an estimation of the probability that the correct classification is C given $S_k(w)$, the class for the word w as suggested by classifier k. $\lambda_k(w)$ is the weight that specifies the importance given to each classifier k in the combination.

$P_k(C|S_k(w))$ is computed as follows

$$P_k(C|S_k(w)) = \begin{cases} 1, & \text{if } S_k(w) = C \\ 0, & \text{otherwise} \end{cases} \tag{7.2}$$

For equal voting, each classifier should have the same weight (e.g., $\lambda_k(w) = 1/K$). In case of weighted voting, the weight associated with each classifier can be computed from its precision and/or recall as illustrated above.

7.2.2 Independent Bayesian Classifier Combination (IBCC)

Using a Bayesian approach to classifier combination (BCC) provides a mathematical combination framework in which many classifiers, with various distributions and training features, can be combined to provide more accurate information. This framework explicitly models the relationship between each classifier's output and the unknown true label [Levenberg *et al.*, 2014]. This section describes the Bayesian approach to the classifier combination we adopted which, like the work of Levenberg *et al.* [2014], is based on Simpson *et al.* [2013] simplification of Ghahramani and Kim [2003] model.

For *ith* data point, true label t_i is assumed to be generated by a multinomial distribution with the parameter δ: $p(t_i = j|\delta) = \delta_j$, which models the class proportions. True labels

may take values $t_i = 1...J$, where J is the number of true classes. It is also assumed that there are K individual classifiers. The output of the classifiers are assumed to be discrete with values $l = 1...L$, where L is the number of possible outputs. The output $c_i^{(k)}$ of the classifier k is assumed to be generated by a multinomial distribution with parameters $\pi_j^{(k)} : p(c_i^{(k)} = l | t_i = j, \pi_j^{(k)}) = \pi_{j,l}^{(k)}$ where $\pi^{(k)}$ is the confusion matrix for the classifier k, which quantifies the decision-making abilities of each individual classifier.

As in Simpson *et al.* [2013], we assume that parameters $\pi_j^{(k)}$ and δ have Dirichlet prior distributions with hyper-parameters $\alpha_{0,j}^{(k)} = [\alpha_{0,j1}^{(k)}, \alpha_{0,j2}^{(k)}, ..., \alpha_{0,jL}^{(k)}]$ and $\nu = [\nu_{0,1}, \nu_{0,2}, ..., \nu_{0,J}]$ respectively. Given the observed class labels and based on the above prior, the joint distribution over all variables for the IBCC model is

$$p(\delta, \Pi, t, c | A_0, \nu) = \prod_{i=1}^{I} \{\delta_{t_i} \prod_{k=1}^{K} \pi^{(k)}_{t_i, c_i^{(k)}}\} p(\delta|\nu) p(\Pi|A), \tag{7.3}$$

where $\Pi = \{\pi_j^{(k)} | j = 1...J, k = 1...K\}$ and $A_0 = \{\alpha_{0,j}^{(k)} | j = 1...J, k = 1...K\}$. The conditional probability of a test data point t_i being assigned class j is given by

$$p(t_i = j) = \frac{\rho_{ij}}{\sum_{y=1}^{J} \rho_{iy}}, \tag{7.4}$$

where

$$\rho_{ij} = \delta_j \prod_{k=1}^{K} \pi_{j, c_i^{(k)}}. \tag{7.5}$$

In our implementation we used point values for A_0 as in [Simpson *et al.*, 2013]. The values of hyper-parameters A_0 offered a natural method to include any prior knowledge. Thus, they can be regarded as pseudo-counts of prior observations and they can be chosen to represent any prior level of uncertainty in the confusion matrices, Π. Our inference technique for the unknown variables (δ, π, and t) was Gibbs sampling as in [Ghahramani and Kim, 2003; Simpson *et al.*, 2013]. Figure 7.1 shows the directed graphical model for IBCC. The $c_i^{(k)}$ represents observed values. Circular nodes are variables with distributions and square nodes are variables instantiated with point values.

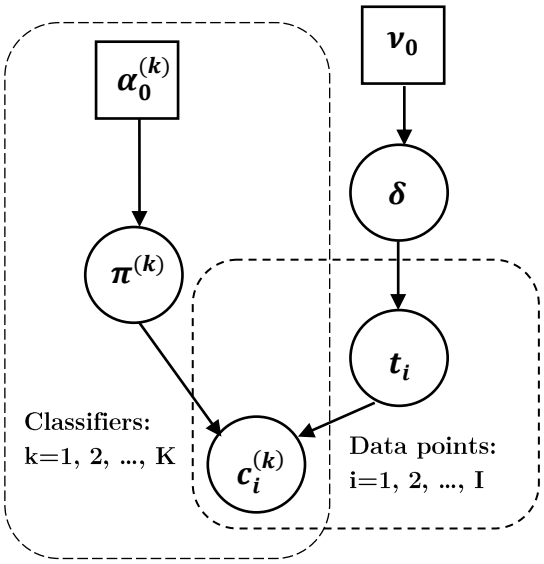


Figure 7.1: The directed graph of IBCC.

7.3 Data

In this section, we describe the two data sets we used:

- Validation set[1](NEWS + BBCNEWS): 90% of this data set is used to estimate the weight of each individual classifier and 10% is used to perform error analysis.
- Test set (ANERcorp test set): This data set is used to evaluate different classifier combination methods.

The validation set is composed of two data sets: NEWS and BBCNEWS. The NEWS set contains around 15k tokens collected by Darwish [2013] from the RSS feed of the Arabic (Egypt) version of news.google.com from October 2012. We created the BBCNEWS corpus by collecting a representative sample of news from BBC in May 2014. It contains around 3k tokens and covers different types of news such as politics, economics, and entertainment. BBCNEWS was annotated by one person, a native Arabic speaker, using the CoNLL tagging guidelines

The ANERcorp test set makes up 20% of the whole ANERcorp data set. The ANERcorp data set is a newswire corpus built and manually tagged, especially for the

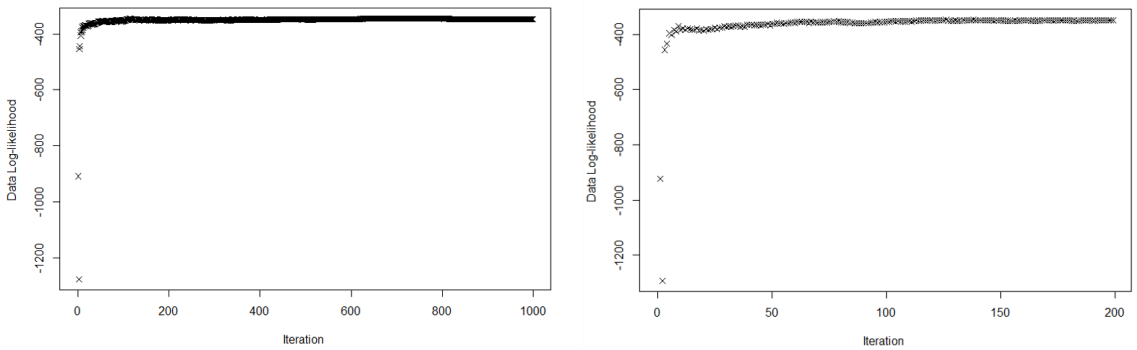
[1]Also known as development set.

Arabic NER task by Benajiba *et al.* [2007], and contains approximately 150K tokens (see Section 4.2 for details).

7.4 Experiments and Results

7.4.1 Experimental Setup

In the IBCC model, the validation data was used as known t_i to ground the estimates of model parameters. The hyper-parameters were set as $\alpha_j^{(k)} = 1$ and $\nu_j = 1$ [Kim and Ghahramani, 2012; Levenberg *et al.*, 2014]. The initial values for random variables were set as follows: (a) the class proportion δ was initialised to the result of counting t_i and (b) the confusion matrix π was initialised to the result of counting t_i and the output of each classifier $c^{(k)}$. We carried out likelihood analysis. We ran Gibbs sampling well past stability (i.e., 1000 iterations). Stability was actually reached in approximately 100 iterations. Figure 7.2(a) shows the scatterplot of the the observed data log-likelihood during the 1000 iterations. Figure 7.2(b) shows a closer picture for only the first 200 iterations. The convergence of the sample label values are noticeable after approximately 100 iterations.



(a) 1000 iterations (b) 200 iterations

Figure 7.2: The progress of the sampler versus iteration.

All parameters required in voting methods were specified using the validation set. We examined two different voting methods: equal voting and weighted voting. In the case of equal voting, each classifier was given an equal weight, (1/K) where K was the number of classifiers to be combined. In weighted voting, total precision was used in order to give

preference to classifiers with good quality.

The SSL and DL classifiers were trained with two different algorithms using different training data. The SSL classifier was trained on ANERcorp training data, while the DL classifier was trained on a corpus automatically derived from Arabic Wikipedia, as explained in Chapter 5 and 6. At the beginning of our experiments we proposed a simple combined classifier that makes the final decision based on the agreed decisions of the individual classifiers, namely the SSL classifier and DL classifier. That is, if the individual classifiers agree on the NE type of a certain word, then it is annotated by an agreed NE type. In the case of disagreement, the word is considered *not* named entity. This simple combined classifier forms the baseline in our experiments.

In the next step, several experiments were carried out in which the SSL and DL classifiers were combined using the three classifier combination methods: equal voting, weighted voting, and IBCC.

We also introduced the classifier combination restriction in order to control how and when the predictions of individual classifiers should be combined. The importance of restricting the classifier combination appeared when we conducted an error analysis of the three combined classifiers using the validation set. The error analysis shows that 10.01% of the NEs were correctly detected by the semi-supervised classifier, but considered *not* NEs by the distant learning classifier. At the same time, the distant learning classifier managed to correctly detect 25.44% of the NEs that were considered *not* NEs by the semi-supervised classifier. We also noticed that false positive rates, i.e. the possibility of considering a word an NE when it is actually *not* an NE, are very low (0.66% and 2.45% for the semi-supervised and distant learning classifiers respectively). These low false positive rates and the high percentage of the NEs that are detected and missed by the two classifiers in a mutually exclusive way can be exploited to obtain better results, more specifically, to increase recall without negatively affecting precision. Therefore, we restricted the combination process to only include situations where the individual classifiers agree or disagree on the NE type of a certain word. The combination process is ignored in cases where the individual classifiers only disagree on detecting NEs. For example, if the individual classifiers disagree on

whether a certain word is an NE or not, the word is automatically considered an NE. Figure 7.3 provides some examples that illustrate the restrictions we applied to the combination process.

Predictions of SSL Classifier	Predictions of DL Classifier	
B-PER	B-LOC	*Apply Combination Method*
O	B-LOC	B-LOC
B-ORG	O	B-ORG
B-PER	B-PER	*Apply Combination Method*

Figure 7.3: Examples of restricting the combination process.

7.4.2 Results

Table 7.2 shows the results of the simple combined classifier, which is considered a baseline in our experiments.

NEs	Precision	Recall	F-measure
PER	97.31	24.69	39.39
LOC	98.35	40.01	56.88
ORG	97.38	33.20	49.52
Overall	97.68	32.63	48.92

Table 7.2: The results of the baseline.

Table 7.3 shows the results of the three classifier combination methods - equal voting, weighted voting and IBCC - when applied to combine the SSL and DL classifiers.

The results of restricting the combination process that we applied in order to control how and when the predictions of individual classifiers should be combined are seen in Table 7.4.

NEs	Combination Methods	Precision	Recall	F-measure
PER	Equal Voting	79.99	41.88	54.97
	Weighted Voting	80.15	44.24	57.01
	IBCC	77.87	63.86	70.17
LOC	Equal Voting	86.87	30.66	45.32
	Weighted Voting	87.48	30.23	44.93
	IBCC	81.52	59.86	69.03
ORG	Equal Voting	97.01	29.97	45.79
	Weighted Voting	98.11	30.98	47.09
	IBCC	95.44	34.31	50.47
Overall	Equal Voting	87.96	34.17	49.22
	Weighted Voting	88.58	35.15	50.33
	IBCC	84.94	52.68	**65.03**
NEs	**Individual Classifiers**	**Precision**	**Recall**	**F-measure**
Overall	SSL	86.03	51.29	64.27
	DL	76.44	56.42	64.92

Table 7.3: The results of various combination methods.

NEs	Combination Methods	Precision	Recall	F-measure
PER	Equal Voting	74.46	61.88	67.59
	Weighted Voting	77.77	63.50	69.91
	IBCC	77.88	64.56	**70.60**
LOC	Equal Voting	74.04	71.36	72.68
	Weighted Voting	74.05	73.70	73.86
	IBCC	76.20	75.91	**76.05**
ORG	Equal Voting	76.01	63.97	69.47
	Weighted Voting	76.30	66.60	71.12
	IBCC	78.91	66.65	**72.26**
Overall	Equal Voting	74.84	65.74	69.99
	Weighted Voting	76.04	67.93	71.76
	IBCC	77.66	69.04	**73.10**
NEs	**Individual Classifiers**	**Precision**	**Recall**	**F-measure**
Overall	SSL	86.03	51.29	64.27
	DL	76.44	56.42	64.92

Table 7.4: The results of various combination methods when restricting the combination process.

We tested the significance of the difference in performance between the three classifier combination methods - equal voting, weighted voting, and IBCC - by conducting two different statistical tests on the results of these combination methods on an ANERcorp test set. The alpha level ($\alpha = 0.01$) was used as a significance criterion for all statistical tests. First, we ran a non-parametric sign test, as seen in Table 7.5.

Combination Without Restriction			
	Equal Voting	Weighted Voting	IBCC
Equal Voting			
Weighted Voting	**0.3394**		
IBCC	<2.2E-16	<2.2E-16	
Combination With Restriction			
	Equal Voting	Weighted Voting	IBCC
Equal Voting			
Weighted Voting	1.78E-07		
IBCC	<2.2E-16	1.97E-06	

Table 7.5: The sign test results (p-values) for the pairwise comparisons of the combination methods.

Second, we used a bootstrap sampling [Efron and Tibshirani, 1994], which is becoming the de facto standard in NLP [Søgaard *et al.*, 2014]. Table 7.6 compares each pair of the three combination methods using a bootstrap sampling over documents with 10,000 replicates. It shows the p-values and confidence intervals of the difference between means.

Combination Without Restriction		
Combination Methods Comparison	p-value	[95% CI]
Weighted Voting, Equal Voting	**0.508**	[-0.365, 0.349]
IBCC, Equal Voting	0.000	[4.800, 6.122]
IBCC, Weighted Voting	0.000	[4.783, 6.130]
Combination With Restriction		
Combination Methods Comparison	p-value	[95% CI]
Weighted Voting, Equal Voting	0.000	[0.270, 0.600]
IBCC, Equal Voting	0.000	[0.539, 0.896]
IBCC, Weighted Voting	0.000	[0.157, 0.426]

Table 7.6: The bootstrap test results (p-values and CI) for the pairwise comparisons of the combination methods.

7.5 Results Discussion

The results of the baseline show very high precision, which indicates that both individual classifiers are mostly accurate. The individual classifiers also commit different errors that are evident in the low recall. The accuracy and diversity of the individual classifiers are the main conditions for a combined classifier to demonstrate better accuracy than any of its components [Dietterich, 2000b]. The results of the baseline confirm that the SSL and DL classifiers can be combined to aggregate their best decisions, and to improve overall performance.

Combining the SSL and DL classifiers using the three classifier combination methods - IBCC, equal voting, and weighted voting - shows that the IBCC scheme outperforms all voting techniques and individual classifiers in terms of F-measure. Regarding precision, voting techniques show the highest scores. However, the high precision is accompanied by a reduction in recall for both voting methods. The sign test shows a small p-value ($p < 0.01$) for each pair of the three combination methods, as seen in Table 7.5, suggesting that these methods are significantly different. The only comparison where no significance was found is *equal voting* vs. *weighted voting*, when we used them to combine the data without any restrictions ($p = 0.3394$). In addition, the bootstrap sampling shows there are highly significant difference, between almost all three methods of combination. The one exception is the comparison between *equal voting* and *weighted voting*, when they are used as a combination method without restriction, which produces an insignificant difference ($p = 0.508$, CI = -0.365 to 0.349). Generally, the IBCC scheme performs significantly better than voting-based combination methods whether we impose restrictions on the combination process or not, as can be seen in Table 7.3 and Table 7.4. It also has relatively high precision compared to the precision of individual classifiers and much better recall.

Restricting the combination process in this way increases recall without negatively affecting the precision, as seen in Table 7.4. The increase in recall makes the overall F-measure for all combination methods higher than those of individual classifiers. This way of using the IBCC model results in a performance level that is superior to all of the individual clas-

sifiers and other voting-based combined classifiers. Therefore, the IBCC model leads to a 12% increase in the performance of the best individual classifier, while voting methods increase the performance by around 7% - 10%. These results highlight the role of restricting the combination, which affects the performance of combination methods and gives more control over how and when the predictions of individual classifiers should be combined.

7.6 Summary

In this chapter, we combined two minimally supervised methods using a variety of classifier combination schemes. Our main contributions include the following:

- We presented a novel approach to Arabic NER using a combination of semi-supervised learning and distant supervision.
- We used the Independent Bayesian Classifier Combination (IBCC) scheme for NER, and compared it to traditional voting methods.
- We introduced the classifier combination restriction as a means of controlling how and when the predictions of individual classifiers should be combined.

The research presented in this chapter demonstrated that combining the two minimal supervision approaches using various classifier combination methods led to better results for NER. The use of IBCC improved the performance by 8 percentage points over the best individual classifier, whereas the improvement in the performance when using voting methods was only 4 to 6 percentage points. Although all combination methods resulted in an accurate classification, the IBCC model achieved better recall than other traditional combination methods. Our experiments also showed how restricting the combination process can increase the recall ability of all the combination methods without negatively affecting the precision.

The approach we proposed can be easily adapted to new NE types and different domains without the need for human intervention. In addition, there are many ways to restrict the combination process according to the applications' preferences, either producing high accuracy or recall.

Part IV

Concluding Remarks

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this work we attempted to move the state-of-the-art in NER forward, focusing on Arabic NER. In particular, our research aimed to overcome the problems raised by traditional supervised NER systems when changing the domain or expanding the set of NE classes: (a) the need for domain-specific experts and (b) the time and effort required for new manually annotated data. Our proposed solution involved two stages of work. First, we developed two new minimally supervised methods that require less human intervention, namely semi-supervised learning and distant learning techniques, and compared their performance levels to those of supervised methods. Second, we addressed the issue of performance of minimally-supervised methods, analysing their strengths and weaknesses, and combining them to obtain better results.

8.1.1 Minimally-supervised Methods

Our **semi-supervised learning** technique does not require annotated training data, or gazetteers. It relies on the bootstrapping technique, which starts with only a few sets of seed instances for a particular NE class (e.g., "London" and "Paris" are two seed instances for the Location class) in order to initiate the learning process. Depending on the seed instances, the algorithm learns the extraction patterns, which are exploited to identify more instances for that NE type (candidate NEs). Then, the candidate NEs are sorted according to a ranking measure in order to select the best of them to serve as the next seed

instances when the process repeats. Our contributions to this part of the work include the following:

- We proposed a new way to produce extraction patterns.
- We proposed a new way to generalise and filter extraction patterns.
- We presented and compared two ranking measures for determining the most reliable candidate NEs: the number of distinct patterns used in extracting candidate NEs and a measure applying Pointwise Mutual Information (PMI).

In a comparison with different supervised NER systems, our semi-supervised classifier (SSL classifier) outperforms some supervised systems that use simple language-independent features. The SSL classifier also can be easily adapted to identify new types of NEs and does not suffer from the problems typical of supervised methods that require annotated training data, and demand more effort and time to extract specialised types of NEs.

Our **distant learning** technique exploits Wikipedia structure to automatically develop an Arabic NE annotated corpus. Each Wikipedia link is transformed into an NE type of the target article in order to produce the NE annotation. Other Wikipedia features, such as redirects, anchor texts, and inter-language links, are used to tag additional NEs, which appear without links in Wikipedia texts. We also exploited the high coverage of Wikipedia in order to address the challenges particular to tagging NEs in Arabic text as follows:

- We introduced a capitalisation probability measure for Arabic words, which are never capitalised, using the high coverage of Wikipedia and the inter-language links.
- We proposed a mechanism to handle the rich morphology in Arabic, and eliminate the need to perform any deep morphological analysis by exploiting Wikipedia features such as anchor texts and redirects.

Our distant learning technique managed to automatically develop a large NE annotated corpus. We referred to that corpus as the Wikipedia-derived corpus (*WDC*). The *WDC* contains about 6 million tokens representing different genres, as Wikipedia is considered an open domain.

The *WDC* corpus was used to train an NE classifier (DL classifier) which was tested on data from different domains. The evaluation results showed that the DL classifier trained on *WDC* can compete with those trained on manually annotated corpora. In a comparison with our semi-supervised algorithm, the DL classifier shows high recall but low precision, while our SSL classifier tends to have very high precision but comparatively low recall.

8.1.2 Combination of Minimally-supervised Methods

The semi-supervised learning and distant learning techniques display different strengths. This complementarity suggests that better results may be obtained by combining them. Therefore, we combined our semi-supervised algorithm with the distant learning technique using a variety of classifier combination schemes. Our main contributions include the following:

- We presented a novel approach to Arabic NER using a combination of semi-supervised learning and distant supervision.
- We used a new approach to classifier combination, the Independent Bayesian Classifier Combination (IBCC) scheme, for NER for the first time and compared it to traditional voting methods.
- We introduced classifier combination restriction as a means of controlling how and when the predictions of individual classifiers are combined.

Our experiments demonstrate that combining the two minimal supervision approaches using various classifier combination methods leads to better results for NER where the performance of the combined classifier outperforms the individual classifiers (SSL classifier and DL classifier). Although all combination methods result in an accurate classification, the IBCC model achieves better recall than other traditional combination methods. The evaluation results also show how restricting the combination process can increase the recall ability of all the combination methods without negatively affecting the precision.

A pipeline for Arabic NLP

During our work on this thesis we developed a Java-based library named "AraNLP" for the processing of Arabic text. AraNLP is an attempt to group most of the essential Arabic text preprocessing tools into one library by integrating or accurately adapting existing tools and by developing new ones when required. The library includes a sentence detector, tokeniser, word segmenter, root stemmer, light stemmer, part-of-speech tagger (POS-tagger), normaliser, and a punctuation and diacritical mark remover (see Appendix C for detailed information about the library).

8.2 Future Work

The scope of our future work is vast. One major route to consider is to use our approaches to Arabic NER - semi-supervised technique, distant learning technique, and a combination of the two - to recognise fine-grained sets of named entity classes and not only for standard classes (Person, Location, and Organisation). Although we have only tested our semi-supervised technique in order to identify three specialised types of NEs (Politicians, Sportspersons, Artists), more entities may be tested in the future.

Another possible direction for our work would involve experimentation with more minimally-supervised learning algorithms, such as self-training and co-training, and investigating their performance for Arabic NER when they work exclusively or cooperatively with the other minimally-supervised techniques.

There is also plenty of room for more research on domain adaptation problem where NER systems developed for one domain do not necessarily perform well on other domains. More investigation is required to fine-tune NER systems to perform well in new domains and to handle the differences in annotation convention for different domains.

References

Abdallah, S., Shaalan, K., and Shoaib, M. (2012). Integrating Rule-Based System with Classification for Arabic Named Entity Recognition. In *Proceedings of the 13th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 311–322. Springer.

AbdelRahman, S., Elarnaoty, M., Magdy, M., and Fahmy, A. (2010). Integrated Machine Learning Techniques for Arabic Named Entity Recognition. *IJCSI International Journal of Computer Science Issues*, **7**(4), 27–36.

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.

Abney, S. (2010). *Semisupervised Learning for Computational Linguistics*. Champan & Hall/CRC Press, Taylor & Francis Group.

Abouenour, L., Bouzoubaa, K., and Rosso, P. (2012). IDRAAQ: New Arabic Question Answering System Based on Query Expansion and Passage Retrieval. In *CLEF 2012 Workshop on Question Answering For Machine Reading Evaluation (QA4MRE), Rome, Italy*. Online Working Notes at `http://clef2012.clef-initiative.eu/index.php?page=Pages/proceedings.php`.

Abuleil, S. and Evens, M. (2004). Extracting Names from Arabic Text for Question-Answering Systems. In *Proceedings of Coupling approaches, coupling media and coupling languages for information retrieval (RIAO 2004)*, pages 638–647.

Al Ameed, H., Al Ketbi, S., Al Kaabi, A., Al Shebli, K., Al Shamsi, N., Al Nuaimi, N., and Al Muhairi, S. (2005). Arabic Light Stemmer: A New Enhanced Approach. In *Proceedings*

of the second international conference on innovations in information technology, pages 1–9.

Al-Johar, B. (1999). *A Portable Natural Language Interface from Arabic to SQL*. Ph.D. thesis, University of Sheffield.

AlGahtani, S., Black, W., and McNaught, J. (2009). Arabic Part-of-Speech Tagging Using Transformation-Based Learning. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools, Cairo, Egypt*, pages 66–70.

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.

Alotaibi, F. and Lee, M. (2012). Mapping Arabic Wikipedia into the Named Entities Taxonomy. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012), Mumbai, India*, pages 43–52.

Alotaibi, F. and Lee, M. (2013). Automatically Developing a Fine-Grained Arabic Named Entity Corpus and Gazetteer by Utilizing Wikipedia. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing, Nagoya, Japan*, pages 392–400.

Alotaibi, F. and Lee, M. (2014). A Hybrid Approach to Features Representation for Fine-grained Arabic Named Entity Recognition. pages 984–995.

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

An, J., Lee, S., and Lee, G. G. (2003). Automatic Acquisition of Named Entity Tagged Corpus from World Wide Web. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 2*, pages 165–168. Association for Computational Linguistics.

Aoun, J. E., Benmamoun, E., and Choueiri, L. (2009). *The Syntax of Arabic*. Cambridge University Press.

Badr, I., Zbib, R., and Glass, J. (2008). Segmentation for English-to-Arabic Statistical Machine Translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 153–156. Association for Computational Linguistics.

Baroni, M., Murphy, B., Barbu, E., and Poesio, M. (2010). Strudel: A Corpus-Based Semantic Model Based on Properties and Types. *Cognitive Science*, **34**(2), 222–254.

Bauer, E. and Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine learning*, **36**(1/2), 105–139.

Benajiba, Y. and Rosso, P. (2007). ANERsys 2.0: Conquering the NER Task for the Arabic Language by Combining the Maximum Entropy with Pos-Tag Information. In *Proceedings of the Workshop on Natural Language-Independent Engineering, IICAI-2007, Pune, India*, pages 1814–1823.

Benajiba, Y. and Rosso, P. (2008). Arabic Named Entity Recognition Using Conditional Random Fields. In *Proceedings of the Workshop on HLT & NLP within the Arabic World, LREC, Marrakech, Morocco*, volume 8, pages 143–153.

Benajiba, Y., Rosso, P., and Benedíruiz, J. M. (2007). ANERsys: An Arabic Named Entity Recognition System Based on Maximum Entropy. In *Proceedings of the Eighth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2007)*, pages 143–153. Springer.

Benajiba, Y., Diab, M., and Rosso, P. (2008a). Arabic Named Entity Recognition: An SVM-Based Approach. In *Proceedings of 2008 Arab International Conference on Information Technology (ACIT)*, pages 16–18.

Benajiba, Y., Diab, M., and Rosso, P. (2008b). Arabic Named Entity Recognition Using Optimized Feature Sets. In *Proceedings of the Conference on Empirical Methods in Nat-*

ural Language Processing, Hawaii, USA, pages 284–293. Association for Computational Linguistics.

Benajiba, Y., Diab, M., and Rosso, P. (2009). Arabic Named Entity Recognition: A Feature-Driven Study. *Audio, Speech, and Language Processing, IEEE Transactions on*, **17**(5), 926–934.

Bies, A. and Maamouri, M. (2003). Penn Arabic Treebank Guidelines. Technical Report TB-1-28-03, Linguistic Data Consortium, University of Pennsylvania, Philadelphia.

Bikel, D., Schwartz, R., and Weischedel, R. (1999). An Algorithm that Learns What's in a Name. *Machine learning*, **34**(1), 211–231.

Bikel, D. M., Miller, S., Schwartz, R., and Weischedel, R. (1997). Nymble: A High-Performance Learning Name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics.

Blake, B. J. (1994). *Case (Cambridge Textbooks in Linguistics)*. Cambridge University Press, Cambridge.

Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. (1998). NYU: Description of the MENE Named Entity System As Used In MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7), Fairfax, Virginia*.

Brunstein, A. (2002). Annotation Guidelines for Answer Types. Technical Report LDC2005T33, Linguistic Data Consortium, Philadelphia.

Buckwalter, T. (2002). Buckwalter Arabic Morphological Analyzer Version 1.0. Technical Report LDC Catalog No.: LDC2002L49, Linguistic Data Consortium, University of Pennsylvania, Philadelphia.

Buckwalter, T. (2004). Buckwalter Arabic Morphological Analyzer Version 2.0. Technical Report LDC Catalog No.: LDC2004L02, Linguistic Data Consortium, University of Pennsylvania, Philadelphia.

Carpenter, B. (2008). Multilevel bayesian models of categorical data annotation. Unpublished manuscript. Online at http://lingpipe-blog.com/lingpipe-white-papers/; accessed 15 March 2015.

Chang, P.-C., Galley, M., and Manning, C. (2008). Optimizing chinese word segmentation for machine translation performance. In *ACL 2008 Workshop on Statistical Machine Translation, Columbus, Ohio*, pages 224–232. Association for Computational Linguistics.

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-supervised Learning*. MIT press, Cambridge MA.

Chinchor, N. A. (1998). Overview of MUC-7/MET-2. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*. Online at `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/overview.html`.

Cohn, T. and Specia, L. (2013). Modelling Annotator Bias with Multi-Task Gaussian Processes: An Application to Machine Translation Quality Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 32–42.

Collins, M. and Singer, Y. (1999). Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Maryland, USA*, pages 189–196.

Cunningham, H., Maynard, D., and Bontcheva, K. (2011). *Text Processing with Gate*. Gateway Press CA.

Dakka, W. and Cucerzan, S. (2008). Augmenting Wikipedia with Named Entity Tags. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 545–552.

Darwish, K. (2002). Building a Shallow Arabic Morphological Analyzer in One Day. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 1–8.

Darwish, K. (2013). Named Entity Recognition Using Cross-Lingual Resources: Arabic as an Example. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1558–1567.

Dawid, A. P. and Skene, A. M. (1979). Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Applied statistics*, **28**(1), 20–28.

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. (2007a). Improved Arabic Base Phrase Chunking with a New Enriched POS Tag Set. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, pages 89–96. Association for Computational Linguistics.

Diab, M. (2007b). Towards an Optimal POS Tag Set for Modern Standard Arabic Processing. In *Proceedings of Recent Advances in Natural Language Processing (RANLP), Borovets, Bulgaria*, pages 91–96.

Diab, M. (2009). Second Generation Amira Tools for Arabic Processing: Fast and Robust Tokenization, POS Tagging, and Base Phrase Chunking. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools*, pages 285–288.

Diab, M., Hacioglu, K., and Jurafsky, D. (2004). Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 149–152. Association for Computational Linguistics.

Dietterich, T. G. (2000a). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine learning*, **40**(2), 139–157.

Dietterich, T. G. (2000b). Ensemble Methods in Machine Learning. In J. Kittler and F. Roli, editors, *Multiple classifier systems*, pages 1–15. Springer.

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., and Weischedel, R. (2004). The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC-2004), Lisbon, Portugal*, volume 4, pages 837–840.

Efron, B. and Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. Champan & Hall/CRC Press.

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). Paper no. 2.

Elgibali, A. (2005). *Investigating Arabic: Current Parameters in Analysis and Learning*, volume 42. Brill.

Elkateb, S., Black, W., Vossen, P., Farwell, D., Rodríguez, H., Pease, A., and Alkhalifa, M. (2006). Arabic WordNet and the Challenges of Arabic. In *Proceedings of the Arabic NLP/MT Conference, London, UK*, pages 15–24.

Elsebai, A., Meziane, F., and Belkredim, F. Z. (2009). A Rule Based Persons Names Arabic Extraction System. *Communications of the IBIMA*, **11**(6), 53–59.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A., Shaked, T., Soderland, S., Weld, D., and Yates, A. (2005). Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, **165**(1), 91–134.

Farber, B., Freitag, D., Habash, N., and Rambow, O. (2008). Improving NER in Arabic Using a Morphological Tagger. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 2509–2514.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named Entity Recognition through Classifier Combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Ghahramani, Z. and Kim, H.-C. (2003). Bayesian Classifier Combination. Technical report, University College London, London, UK.

Goldsmith, J. A., Higgins, D., and Soglasnova, S. (2001). Automatic Language-Specific Stemming in Information Retrieval. In Peters, Carol, editor, *Cross-Language Information Retrieval and Evaluation*, pages 273–283. Springer.

Goweder, A., Poesio, M., and De Roeck, A. (2004). Broken Plural Detection for Arabic Information Retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 566–567. ACM.

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.

Grishman, R. and Sundheim, B. (1996). Message Understanding Conference-6: A Brief History. In *Proceedings of the 16th International Conference on Computational linguistics, Copenhagen, Denmark*, volume 96, pages 466–471.

Guo, J., Xu, G., Cheng, X., and Li, H. (2009). Named Entity Recognition in Query. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, USA*, SIGIR '09, pages 267–274. ACM.

Habash, N. (2010). Introduction to Arabic Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, **3**(1), 1–187.

Habash, N. and Rambow, O. (2005). Arabic Tokenization, Part-Of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

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

NAACL, New York City, USA, NAACL-Short '06, pages 49–52. Association for Computational Linguistics.

Habash, N., Soudi, A., and Buckwalter, T. (2007). On Arabic Transliteration. In A. Soudi, G. Neumann, and A. Van den Bosch, editors, *Arabic computational morphology: knowledge-based and empirical methods*, pages 15–22. Springer.

Habash, N., Rambow, O., and Roth, R. (2009). MADA+ TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, Pos Tagging, Stemming and Lemmatization. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt*, pages 102–109.

Haitovsky, Y., Smith, A., and Liu, Y. (2002). Modelling Disagreements among and within Raters' Assessments from the Bayesian Point of View. In *Draft. Presented at the Valencia meeting 2002*.

Hammo, B., Abu-Salem, H., and Lytinen, S. (2002). QARAB: A Question Answering System to Support the Arabic Language. In *Proceedings of the ACL workshop on Computational Approaches to Semitic Languages*, pages 1–11. Association for Computational Linguistics.

Haywood, J. A. and Nahmad, H. M. (1962). *A New Arabic Grammar of the Written Language*. Lund, Humphries.

Hetzron, R. (2013). *The Semitic Languages*. Routledge, Taylor & Francis Group.

Isozaki, H. and Kazawa, H. (2002). Efficient Support Vector Classifiers for Named Entity Recognition. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jones, R. (2005). *Learning to Extract Entities from Labeled and Unlabeled Text*. Ph.D. thesis, University of Utah.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduc-*

tion to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice-Hall, Upper Saddle River, New Jersey.

Kaye, A. S. (1996). Adaptations of Arabic Script. In P. T. Daniels and W. Bright, editors, *The world's writing systems*, pages 743–762. Oxford University Press.

Kazama, J. and Torisawa, K. (2007). Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

Khoja, S. (2001). APT: Arabic Part-of-Speech Tagger. In *Proceedings of the Student Workshop at NAACL*, pages 20–25.

Khoja, S. and Garside, R. (1999). Stemming Arabic Text. *Lancaster, UK, Computing Department, Lancaster University*. `http://zeus.cs.pacificu.edu/shereen/research.htm`; accessed 23 February 2015.

Kim, H.-C. and Ghahramani, Z. (2012). Bayesian Classifier Combination. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 619–627.

Korayem, M., Crandall, D., and Abdul-Mageed, M. (2012). Subjectivity and Sentiment Analysis of Arabic: A Survey. In A.-E. Hassanien, A.-B. Salem, R. Ramadan, and T.-h. Kim, editors, *Advanced Machine Learning Technologies and Applications*, pages 128–139. Springer.

Krallinger, M., Leitner, F., Rabal, O., Vazquez, M., Oyarzabal, J., and Valencia, A. (2013). Overview of the Chemical Compound and Drug Name Recognition (CHEMDNER) Task. In *Proceedings of the 4th Workshop on BioCreative Challenge Evaluation*, volume 2, pages 2–33.

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

Kudo, T. and Matsumoto, Y. (2001). Chunking with Support Vector Machines. In *Proceedings of the second meeting of the North American Chapter of the Association for*

Computational Linguistics on Language technologies, pages 1–8. Association for Computational Linguistics.

Kulick, S., Gabbard, R., and Marcus, M. (2006). Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference, Prague, Czech Republic*, pages 31–42.

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. S., Ballesteros, L., and Connell, M. E. (2007). Light Stemming for Arabic Information Retrieval. In A. Soudi, G. Neumann, and A. Van den Bosch, editors, *Arabic computational morphology: knowledge-based and empirical methods*, pages 221–243. Springer Netherlands.

Leech, G. (1997). Introducing Corpus Annotation. In R. Garside, G. N. Leech, and T. McEnery, editors, *Corpus Annotation*, pages 1–18. Routledge, Taylor & Francis Group.

Levenberg, A., Pulman, S., Moilanen, K., Simpson, E., and Roberts, S. (2014). Predicting Economic Indicators from Web Text Using Sentiment Composition. *International Journal of Computer and Communication Engineering*, **3**(2), 109–115.

Liao, W. and Veeramachaneni, S. (2009). A Simple Semi-Supervised Algorithm for Named Entity Recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing, Boulder, Colorado, USA*, pages 58–65. Association for Computational Linguistics.

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

Maamouri, M., Bies, A., Buckwalter, T., and Mekki, W. (2004). The Penn Arabic Tree-

bank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109.

Maclin, R. and Opitz, D. (1997). An Empirical Evaluation of Bagging and Boosting. pages 546–551.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT press.

Marcus, M., Marcinkiewicz, M., and Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational linguistics*, **19**(2), 313–330.

Marsh, E. and Perzanowski, D. (1998). MUC-7 Evaluation of IE Technology: Overview of Results. In *Proceedings of the seventh message understanding conference (MUC-7)*.

Marton, Y., Habash, N., and Rambow, O. (2010). Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages, Los Angeles, California*, pages 13–21. Association for Computational Linguistics.

McCallum, A. and Li, W. (2003). Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

Merchant, R., Okurowski, M. E., and Chinchor, N. (1996). The Multilingual Entity Task (MET) Overview. In *Proceedings of the Tipster Text Program Phase II, Vienna, Virginia: Morgan Kaufmann Publishers, Inc*, pages 445–447.

Mihalcea, R. and Chklovski, T. (2003). Open Mind Word Expert: Creating Large Annotated Data Collections with Web Users' Help. In *Proceedings of the EACL 2003 Workshop on Linguistically Annotated Corpora (LINC 2003)*, pages 53–61.

Mika, P., Ciaramita, M., Zaragoza, H., and Atserias, J. (2008). Learning to Tag and Tagging to Learn: A Case Study on Wikipedia. volume 23, pages 26–33.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing, Suntec, Singapore*, ACL '09, pages 1003–1011. Association for Computational Linguistics.

Mohit, B., Schneider, N., Bhowmick, R., Oflazer, K., and Smith, N. (2012). Recall-Oriented Learning of Named Entities in Arabic Wikipedia. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012), Avignon, France*, pages 162–173.

Monroe, W., Green, S., and Manning, C. D. (2014). Word Segmentation of Informal Arabic with Domain Adaptation. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland*, pages 206–211.

Nadeau, D. and Sekine, S. (2007). A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, **30**(1), 3–26.

Nadeau, D., Turney, P., and Matwin, S. (2006). Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In *Proceedings of the 19th Canadian Conference on Artificial Intelligence, Qubec City, Qubec, Canada*, pages 266–277. Springer.

Nothman, J., Curran, J. R., and Murphy, T. (2008). Transforming Wikipedia into Named Entity Training Data. In *Proceedings of the Australian Language Technology Workshop*, pages 124–132.

Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J. R. (2013). Learning Multilingual Named Entity Recognition from Wikipedia. *Artificial Intelligence*, **194**(1), 151–175.

Nwesri, A., Tahaghoghi, S., and Scholer, F. (2005). Stemming Arabic Conjunctions and Prepositions. In *proceedings of the International Symposium on String Processing and Information Retrieval, SPIRE 2005*, pages 206–217. Springer.

Opitz, D. and Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, **11**, 169–198.

Oudah, M. and Shaalan, K. (2012). A Pipeline Arabic Named Entity Recognition Using a Hybrid Approach. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012), Mumbai, India*, pages 2159–2176.

Pantel, P. and Pennacchiotti, M. (2006). Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 113–120. Association for Computational Linguistics.

Pasca, M., Lin, D., Bigham, J., Lifchits, A., and Jain, A. (2006). Organizing and Searching the World Wide Web of Facts-Step One: The One-Million Fact Extraction Challenge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 6, pages 1400–1405.

Payne, T. E. (2006). *Exploring Language Structure: A Student's Guide*. Cambridge University Press.

Poibeau, T. and Kosseim, L. (2001). Proper Name Extraction from Non-Journalistic Texts. *Language and computers*, **37**(1), 144–157.

Polikar, R. (2007). Bootstrap-Inspired Techniques in Computation Intelligence. *Signal Processing Magazine, IEEE*, **24**(4), 59–72.

Ramshaw, L. A. and Marcus, M. P. (1995). Text Chunking Using Transformation-Based Learning. In *In Proceedings of the Third ACL Workshop on Very Large Corpora, Cambridge, MA, USA*, pages 82–94.

Ratinov, L. and Roth, D. (2009). Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Rau, L. F. (1991). Extracting Company Names from Text. In *Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications*, volume 1, pages 29–32. IEEE.

Richman, A. E. and Schone, P. (2008). Mining Wiki Resources for Multilingual Named Entity Recognition. In *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*, pages 1–9.

Riloff, E. and Jones, R. (1999). Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the AAAI Conference on Artificial Intelligence, Orlando, Florida*, pages 474–479. JOHN WILEY & SONS LTD.

Ritter, A., Clark, S., Etzioni, O., *et al.* (2011). Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Ryding, K. C. (2005). *A Reference Grammar of Modern Standard Arabic*. Cambridge University Press.

Saha, S. and Ekbal, A. (2013). Combining Multiple Classifiers Using Vote Based Classifier Ensemble Technique for Named Entity Recognition. *Data & Knowledge Engineering*, **85**, 15–39.

Sang, E. F. and Veenstra, J. (1999). Representing Text Chunks. In *Proceedings of the ninth conference on European Chapter of the Association for Computational Linguistics*, pages 173–179. Association for Computational Linguistics.

Santos, D., Seco, N., Cardoso, N., and Vilela, R. (2006). HAREM: An Advanced NER Evaluation Contest for Portuguese. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), Genoa, Italy*, pages 1986–1991.

Sawalha, M. and Atwell, E. (2008). Comparative Evaluation of Arabic Language Morphological Analysers and Stemmers. In *Proceedings of the 22nd International Conference on Comptational Linguistics (Poster Volume)*, pages 107–110. Coling 2008 Organizing Committee.

Sekine, S. *et al.* (1998). NYU: Description of the Japanese NE System Used for MET-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7), Fairfax, Virginia*.

Sekine, S. and Isahara, H. (1999). IREX Project Overview. In *Proceedings of the IREX Workshop*, pages 7–12.

Sekine, S. and Nobata, C. (2004). Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 1977–1980.

Sekine, S., Sudo, K., and Nobata, C. (2002). Extended Named Entity Hierarchy. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Settles, B. (2004). Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics.

Shaalan, K. (2014). A Survey of Arabic Named Entity Recognition and Classification. *Computational Linguistics*, **40**(2), 469–510.

Shaalan, K. and Raza, H. (2007). Person Name Entity Recognition for Arabic. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, Prague, Czech Republic*, pages 17–24. Association for Computational Linguistics.

Shaalan, K. and Raza, H. (2009). NERA: Named Entity Recognition for Arabic. *Journal of the American Society for Information Science and Technology*, **60**(8), 1652–1663.

Simpson, E., Roberts, S., Psorakis, I., and Smith, A. (2013). Dynamic Bayesian Combination of Multiple Imperfect Classifiers. In T. V. Guy, M. Kárnỳ, and D. H. Wolpert, editors, *Decision Making and Imperfection*, pages 1–35. Springer.

Søgaard, A., Johannsen, A., Plank, B., Hovy, D., and Martinez, H. (2014). What's in a p-value in NLP? In *Proceedings of the eighteenth conference on computational natural language learning (CoNLL14)*, pages 1–10.

Soudi, A., Neumann, G., and Van den Bosch, A., editors (2007). *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Srihari, R. and Li, W. (1999). Information Extraction Supported Question Answering. In *Proceedings of the 8th Text Retrieval Conference (TREC-8), Gaithersburg, Maryland*, pages 185–196. National Institute of Standards and Technology (NIST).

Strassel, S., Mitchell, A., and Huang, S. (2003). Multilingual Resources for Entity Extraction. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition, Sapporo, Japan*, pages 49–56. Association for Computational Linguistics.

Sundheim, B. M. (1996). Overview of Results of the MUC-6 Evaluation. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 423–442. Association for Computational Linguistics.

Taghva, K., Elkhoury, R., and Coombs, J. (2005). Arabic Stemming without a Root Dictionary. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2005)*, volume 1, pages 152–157. IEEE.

Tardif, S., Curran, J. R., and Murphy, T. (2009). Improved Text Categorisation for Wikipedia Named Entities. In *Proceedings of the Australasian Language Technology Association Workshop, Sydney, Australia*, pages 104–108.

Testen, D. (2015). *Semitic Languages*. Encyclopedia Britannica. Online at http://www.britannica.com/EBchecked/topic/534171/Semitic-languages; accessed 23 January 2015.

Thompson, H. S. and McKelvie, D. (1997). Hyperlink Semantics for Standoff Markup Of Read-Only Documents. In *Proceedings of SGML Europe*, volume 97, pages 227–229.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL), Taipei, Taiwan*, pages 155–158.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4, Edmonton, Canada*, pages 142–147. Association for Computational Linguistics.

Tsai, R., Wu, S., Chou, W., Lin, Y., He, D., Hsiang, J., Sung, T., and Hsu, W. (2006). Various Criteria in the Evaluation of Biomedical Named Entity Recognition. *BMC bioinformatics*, **7**(2). Paper no. 92.

Tulyakov, S., Jaeger, S., Govindaraju, V., and Doermann, D. (2008). Review of Classifier Combination Methods. In S. Marinai and H. Fujisawa, editors, *Machine Learning in Document Analysis and Recognition*, pages 361–386. Springer.

Turney, P. D., Pantel, P., *et al.* (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of artificial intelligence research*, **37**(1), 141–188.

UNESCO (2012). The first World Arabic Language Day, 18 December 2012. Online at `http://www.unesco.org/new/en/unesco/events/prizes-and-celebrations/celebrations/international-days/world-arabic-language-day/`; accessed 11 January 2016.

UNESCO (2015). The occasion of World Arabic Language Day 18 December 2015. Online at `http://www.unesco.org/new/en/unesco/events/prizes-and-celebrations/celebrations/international-days/world-arabic-language-day-2015/`; accessed 11 January 2016.

Van Erp, M., Vuurpijl, L., and Schomaker, L. (2002). An Overview and Comparison of Voting Methods for Pattern Recognition. In *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 195–200. IEEE.

Van Halteren, H., Daelemans, W., and Zavrel, J. (2001). Improving Accuracy in Word Class Tagging through The Combination of Machine Learning Systems. *Computational linguistics*, **27**(2), 199–229.

Watson, J. (2007). *The Phonology and Morphology of Arabic*. Oxford University Press.

Zaghouani, W. (2014). Critical Survey of the Freely Available Arabic Corpora. In *Proceedings of the Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools, Reykjavik, Iceland*, pages 1–8.

Zesch, T., Gurevych, I., and Mühlhäuser, M. (2008). Analyzing and Accessing Wikipedia as a Lexical Semantic Resource. In *Proceedings of the Biannual Conference of the Society for Computational Linguistics and Language Technology*, pages 213–221.

Zhang, T., Damerau, F., and Johnson, D. (2002). Text Chunking Based on a Generalization of Winnow. *The Journal of Machine Learning Research*, **2**, 615–637.

Appendix A: Arabic Reduced Tag Set

POS tag	**Definition**
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner / demonstrative pronoun
FW	Foreign word
IN	Subordinating conjunction / preposition
JJ	Adjective
NN	Common noun, singular
NNS	Common noun, plural or dual
NNP	Proper noun, singular
NNPS	Proper noun, plural or dual
NO_FUNC	Unanalysed word
NUMERIC_COMMA	(,) / decimal point / the Arabic letter (ر) used as a comma
PRP	Personal pronoun
PRP$	Possessive personal pronoun
PUNC	Punctuation
RB	Adverb
RP	Particle
UH	Interjection
VB	Imperative verb
VBN	Passive imperfect / perfect verb
VBP	Active imperfect verb
VBD	Active perfect verb
WP	Relative pronoun
WRB	Relative adverb

Appendix B: CoNLL 2003 Annotation Guidelines

- **Person**
  - first, middle and last names of people, animals and fictional characters aliases
- **Location**
  - roads
    - streets, motorways
  - trajectories
  - regions
    - villages, towns, cities, provinces, countries, continents, dioceses, parishes
  - structures
    - bridges, ports, dams
  - natural locations
    - mountains, mountain ranges, woods, rivers, wells, fields, valleys, gardens, nature reserves, allotments, beaches, national parks
  - public places
    - squares, opera houses, museums, schools, markets, airports, stations, swimming pools, hospitals, sports facilities, youth centres, parks, town halls, theatres, cinemas, galleries, camping grounds, NASA launch pads, club houses, universities, libraries, churches, medical centres, parking lots, playgrounds, cemeteries
  - commercial places
    - chemists, pubs, restaurants, depots, hostels, hotels, industrial parks, nightclubs, music venues
  - assorted buildings
    - houses, monasteries, creches, mills, army barracks, castles, retirement homes, towers, halls, rooms, vicarages, courtyards
  - abstract "places"
    - (e.g. *the free world*)
- **Organisation**
  - Companies
    - press agencies, studios, banks, stock markets, manufacturers, cooperatives

  - subdivisions of companies
    + newsrooms
  - brands
  - political movements
    + political parties, terrorist organisations
  - government bodies
    + ministries, councils, courts, political unions of countries (e.g. *the U.N.*)
  - publications
    + magazines, newspapers, journals
  - musical companies
    + bands, choirs, opera companies, orchestras
  - other collections of people
    + sports clubs, sports teams, associations, theatres companies, religious orders, youth organisations
- **Miscellaneous**
  - words of which one part is a location, organisation, miscellaneous, or person adjectives and other words derived from a word which is location, organisation, miscellaneous, or person
  - religions
  - political ideologies
  - nationalities
  - languages
  - programs
  - events
    + conferences, festivals, sports competitions, forums, parties, concerts
  - wars
  - sports related names
    + league tables, leagues, cups
  - titles
    + books, songs, films, stories, albums, musicals, TV programs
  - slogans
  - eras in time
  - types (not brands) of objects
    + car types, planes, motorbikes

Appendix C: A Pipeline for Arabic NLP (AraNLP)

A good number of tools are available for preparing Arabic text and developing Arabic NLP systems. Integration and compatibility problems, however, might occur with some of these 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 applications requirements. Therefore, providing a library equipped with all or most of the tools essential for the processing of Arabic text (e.g., tokenisation, 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. For that reason, we developed AraNLP, a Java-based library that covers various Arabic text preprocessing tools. It is an attempt to group the vital Arabic text preprocessing tools into one library by integrating existing tools and by developing new ones when required. The library includes:

- Sentence detector,
- Tokeniser,
- Light Stemmer,
- Root Stemmer,
- Part-of-speech (POS) Tagger,
- Word Segmenter,
- Normaliser,
- Punctuation and Diacritical marks remover.

AraNLP has already been used in this thesis to prepare the Arabic text for the experiments and it successfully preprocessed the data sets. Figure 1 illustrates a typical processing pipeline applying the tools provided by AraNLP. The library is available free online[1].

[1] AraNLP is available at `https://sites.google.com/site/mahajalthobaiti`

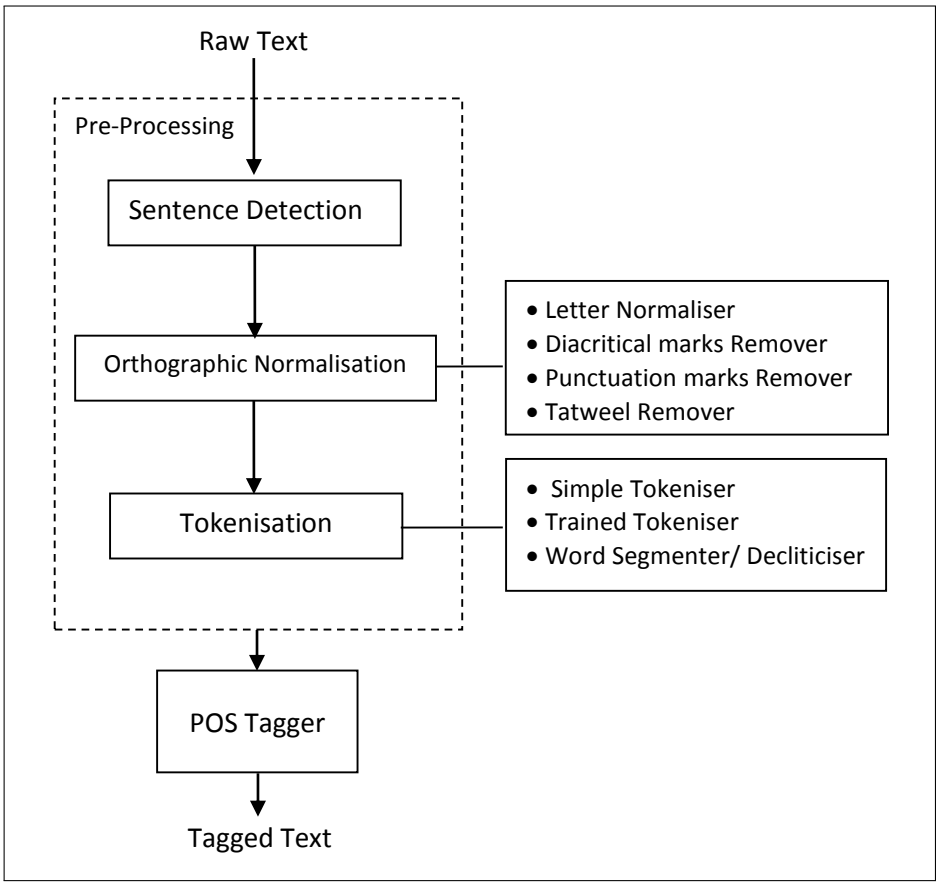


Figure 1: Typical processing pipeline of AraNLP.

C.1 The Modules

C.1.1 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 [7]. 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 marks, as in the study of [12]. 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[2] 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. The trained model performed well on a test data set made up of 871 sentences, with 0.97 precision and recall reaching nearly 0.98.

C.1.2 Tokenisation

The standard preprocessing step for many NLP tasks is tokenisation, which divides a string of written language into its component tokens. For less complex languages, tokenisation usually involves splitting punctuation marks, and some affixes off of the words. On the other hand, morphologically rich languages, like Arabic, require a more extensive tokenisation process to separate different types of clitics and particles from the word. This complex tokenisation is usually called word segmentation [6]. The word segmenter provided by AraNLP is discussed in detail in Section C.1.4.

More relevant to our current topic is simple tokenisation, which only splits off punctuation marks and non-alphanumeric characters from words. Although this type of tokenisation 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 marks 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 marks. In addition, specific domains (e.g., University domain) introduce new types of entities (e.g., Course Code and Room Number) that contain punctuation marks and other symbols that should be considered single tokens [3]. Thus, it is necessary to take

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

into account the careful separation of non-alphanumeric 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 test data set with 21,000 tokens was used to evaluate the trained tokeniser, which achieved a 0.97 precision and recall score.

C.1.3 Stemming

Stemming is the process of reducing derived or inflected words to their stems or original roots. Research has shown that Arabic stemming is challenging because of its highly inflectional and derivational nature [2, 11]. 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 [2, 5].

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 [10] and [11]. 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 [1].

As for the root stemmer, we incorporated the algorithm provided by [9]. 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. 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.

C.1.4 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 analysing the text syntactically. These are great benefits for many NLP tasks such as POS tagging. A large number of possible segmentation levels 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 clitics (e.g., conjunctions, interrogative clitics, definite articles, future verbal particle).
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.

C.1.5 Arabic Normalisation

A large number of NLP tasks require the text be free of punctuation or diacritical marks, if not both. Therefore, we implemented a simple tool to remove punctuation and diacritical marks. It removes all three forms of diacritical marks, as suggested by [4]: vowel, nunation, and the shadda. The tool removes the following default punctuation marks: 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* (ا، آ، أ، إ) might be written interchangeably; another example is *alif maqSuwrah* and regular dotted *yaa’* (ى، ي) which are usually used interchangeably at the final position of the word.

The same is true for *taa' marbuwTah* and *haa'* (ه ،ة). These misspelling errors in Arabic affect 11% of all words in the Penn Arabic Treebank (PATB) [8]. AraNLP provides a different level of orthographic normalisation that can be carried out on Arabic text to reduce noise and data sparsity. This includes normalisation of the hamzated *alif* to a bare *alif* (*alif* without *hamzah*), normalisation of *taa' marbuwTah* to *haa'*, normalisation of the dotless *yaa'* (*alif maqSuwrah*) to *yaa'*, and the removal of *tatweel* (stretching character). AraNLP enables the user to customize the level of normalisation according to the application's need. In addition, the punctuation can easily be added or deleted from the list of punctuation marks.

References

[1] 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.

[2] 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.

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

[4] Diab, M., Ghoneim, M., and Habash, N. (2007). Arabic Diacritization in the Context of Statistical Machine Translation. In *Proceedings of MT-Summit, Copenhagen, Denmark*.

[5] 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). Paper no. 2.

[6] 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.

[7] Grefenstette, G. and Tapanainen, P. (1994). What is a Word, what is a Sentence? Problems of Tokenisation. In *Proceedings of the 3rd International Conference on Computational Lexicography and Text Research, Budapest, Hungary*, pages 79–87.

[8] Habash, N. Y. (2010). Introduction to Arabic Natural Language Processing. *Synthesis Lectures on Human Language Technologies*, **3**(1), 1–187.

[9] Khoja, S. and Garside, R. (1999). Stemming Arabic Text. *Lancaster, UK, Computing Department, Lancaster University*. `http://zeus.cs.pacificu.edu/shereen/research.htm`; accessed 23 February 2015.

[10] 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.

[11] Larkey, L. S., Ballesteros, L., and Connell, M. E. (2007). Light Stemming for Arabic Information Retrieval. In A. Soudi, G. Neumann, and A. Van den Bosch, editors, *Arabic computational morphology: knowledge-based and empirical methods*, pages 221–243. Springer Netherlands.

[12] 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.