Université François Rabelais – Tours, antenne de Blois

Laboratoire d'Informatique

(UPRES EA n°2101)

State of the Art NLP: Multilingualism and Semantic Parsing in Information Retrieval and Extraction

Authors
Tamara MENDT - tammymendt@gmail.com
Poly MURIITHI - kinyamuriithi@gmail.com
Elena SAMOTA - esamota@css.edu
Rizkallah TOUMA - rizkallah.touma@yahoo.com

Supervisors
Agata SAVARY
Jean-Yves ANTOINE
Jakub WASZCZUK

Contact:
Agata SAVARY
Laboratoire d'Informatique
Ecole Polytechnique – Département Informatique
64 avenue Jean Portalis
37200 TOURS – FRANCE
Tél.: 02.47.36.14.14
Fax: 02.47.36.14.36
email: agata.savary@univ-tours.fr

16 June 2014
Technical Report n°308, 47 pages

Contents

1 Introduction

Current technologies are making increasingly vast amounts of information available to common internet users. Though the availability of data is ever growing, finding information relevant to a specific need is not always straightforward. Information Retrieval techniques deal with this problem and continue to receive a lot of attention in today's data-driven world.

Information Retrieval (IR) and Natural Language Processing (NLP) are deeply related because most search engines are based on a document corpus written in natural language text. Traditionally, Information Retrieval systems have been aimed at shallow levels of parsing of input queries and most studies in NLP for Information Retrieval have been directed towards parsing the English language. However, there is an increasing need to expand the scope of this type of systems due to more demanding users seeking precise answers, and to the growing nature of (non-English) online content.

This paper discusses two main problems with non-traditional IR systems: multilingual information retrieval and understanding the logical meaning of queries to find more precise answers. We analyze the challenges posed by both issues and the techniques that are being applied to solve them.

The paper is organized in four main parts. Section two will discuss linguistic features of languages that pose challenges for information retrieval. In particular, we analyze some features that are not present in the English language but that are specific to two languages which were studied in contrast to English, namely Arabic and Russian. Section three will discuss the issue of resource availability to support NLP and the study of languages. In particular, we discuss resources from the three languages we cover in this paper. Section four presents the different language processing levels used by IR systems, highlighting the challenges of each level. We discuss how some of these difficulties have been overcome and which additional difficulties are still open or arise when considering different languages. Finally we discuss two new trends in the retrieval of information: open domain information extraction and question answering. These techniques require deep parsing strategies to correctly interpret the logical meaning of natural language in order to either convert natural language into a structured logical form, or into a query which can be executed on a database of structured information.

2 Language Properties as Challenges to IR and IE

This section discusses different linguistic features which hinder the processing of natural language queries and their mapping to relevant information. We do this as an introduction for the reader to understand why some particular challenges arise when trying to process a natural language input. We classify these features according to their nature and explain the challenges inherent to each property. Many of the properties we present in this chapter are not present in the English language, but are specific to the two languages that we study in the context of multilingualism in IR: Arabic and Russian. Because these two languages are so different from English, they are interesting subjects to understand the challenges posed by multilingualism. We wish to provide the reader with the notion that IR techniques developed for English language search engines, may not be efficient for search engines in other languages. Furthermore, Russian and Arabic were a natural choice of languages because they are native to two of the authors of this paper. We will begin this chapter by briefly describing the two studied languages and continue to present language properties.

Arabic

Arabic is a Semitic language which was first formalized in the 6th century with the birth of Islam. It is written using the Arabic script, which is an abjad [1] consisting of 28 letters and is written from right to left. It is the native language of almost 300 million people and is one of the six official languages of the United Nations.

Many dialects of Arabic exist across the Arab World, with important phonological, morphological, lexical and syntactic differences compared sometimes to those among the Romance languages [22]. The effects of these differences on natural language processing tasks will be discussed later in this section.

Throughout this report, the Habash-Soudi-Buckwalter transliteration system for Arabic will be used [43]. We included the transliteration table in appendix A - Habash-Soudi-Buckwalter Arabic Transliteration Scheme. It should be noted that while most English glosses contain the lowercase letters (a, i, u), these are only added to enhance readability and are usually not present in the original Arabic words

[1]An *abjad* is a writing system where most symbols correspond to consonants while vowels can usually be added by the means of diacritics.

due to the lack of vocalization (see Section 2.3.2).

Russian

Russian is a Slavic language that belongs to the Indo-European family of languages and is natively spoken by 150 million people. Along with Ukrainian and Belorussian, it was derived from the Old Russian language in 14th-15th century. Russian is the eighth most spoken language in the world by the number of native speakers and one of the six official languages of the United Nations.

Russian uses the Cyrillic alphabet and is written left to right, similar to English. The Cyrillic alphabet consists of 33 letters and can be easily transliterated using the Latin alphabet. The Russian language is a morphologically rich and highly inflective language that makes natural language processing more challenging in comparison to English (see Section 2.1.1).

The remainder of this section will discuss linguistic features that were found across the languages, highlighting the problematic nature of each language and paving the way for a detailed discussion on how they are addressed in section 4.

2.1 Morphological Properties

2.1.1 Inflectional Morphology

In linguistics, inflection is the modification of the word to express different grammatical categories such as tense, mood, gender, number, person and voice. Different language families show different degrees of inflection in their morphology. For example, modern English is considered a weakly inflected language as there are only two forms of nouns (singular and plural) and four forms of verbs [2]. On the other hand, both Russian and Arabic are highly inflected languages with a complex agreement system between nouns and several forms for verbs. In the following subsections, we will briefly discuss the inflection for both languages in nouns, adjectives and verbs.

Inflection in Arabic

Arabic verbal morphology is very regular with only a handful of exceptions. Almost all verbs follow a set of predefined patterns in conjugation and inflection. Nonetheless, they are still highly inflected as they inflect for subject (person, number and gender), aspect, mood and voice [42]. As an example, consider the phrase in Table 1:

سوريا دمّرت ولم يفعلوا شيئاً
swryA dum~irat wa+lam yafσalwA šayã
Syria destoyed and+not do anything
"Syria was destroyed and they didn't do anything"

Table 1: Arabic Verbal Inflection

In the example in Table 1, there are two verbs that show inflectional features:

- *dum~irat*: meaning "to destroy", inflects for its subject, Syria, on person (other), number (singular) and gender (feminine). It also inflects for aspect (perfective), mood (indicative) and voice (passive).
- *yafσalwA*: meaning "to do", inflects for its dropped subject on person (other), number (plural) and gender (masculine). It also inflects for aspect (perfective), mood (jussive) and voice (active).

Compared to verbs, Arabic nominal morphology is far more complicated and has many exceptions and special cases. In general, Arabic nouns inflect for gender (masculine and feminine), number (singular, dual and plural), case (nominative, accusative and genitive) and state (definite and indefinite). A list of the rules for a regular noun inflection can be found in [42]. It is worth mentioning that singular nouns inflect for case only by changing the vocalization of the last letter. However, there are many special cases in obtaining the inflected forms of nouns, the most common of which is the "broken plural". **Broken plurals** account for almost 50% of all plurals in Arabic [42] and are formed by atypical stem and suffix

[2]as described in Brinton, Laurel J. (2000). *The structure of modern English: a linguistic introduction.* Amsterdam, Philadelphia: John Benjamins. p. 104.

alternations which do not conform to a predefined rule. For example, Table 2 includes words that have the same ending when in singular but that inflect in different ways when transformed into their broken plural. This makes it more difficult for language processing tools to identify the number of the inflected word, as there is no specific rule to form the broken plural.

Singular		Plural		Translation
رفيقة	rafyq$\hbar$	رفيقات	rafyqAt	Friend (fem.)
غرفة	γurf$\hbar$	غرف	γuraf	Room (fem.)
مدرسة	madr~as$\hbar$	مدارس	madAris	School (fem.)
خليفة	xalyf$\hbar$	خلفاء	xulafA´	Caliph (masc.)

Table 2: Arabic Nominal Inflection - Broken Plural

It can also be noted that, while all the above words have the same ending when they are in singular form, some are feminine and others are masculine, which makes it hard to determine the gender of the word. Many other special cases of Arabic nominal inflection are detailed in [42].

Inflection in Russian

Russian is a language with a rich morphology and hence, it is highly inflective. This property makes morphological analysis more challenging than for a morhologically poor language like English. As previously discussed, Russian is a synthetic and fusional or inflectional language. In this section we define inflection as the phenomenon of the same lexeme having many inflected forms for many different combinations of grammatical features.

Grammatical Inflection

Russian inflectional grammar is very complex. Since the context of this research is to evaluate language processing in the specific application of information retrieval, we will focus on nouns and adjectives. Research shows that verbs are less important to retrieving relevant documents, because nouns and adjectives are the parts of speech that carry the biggest semantic lead in a document [28]. In what follows we present a list of grammatical inflection in Russian nouns and adjectives:

- Nouns can have one of the three possible genders: masculine (masc), feminine (fem), and neuter (neut).
- Nouns decline according to number: singular (sg) or plural (pl), and six grammatical cases: nominative (nom), genetive (gen), dative (dat), accusative (acc), instrumental (inst), and locative (loc), which is also called prepositional (prep).
- Nouns form gender-case combination which, in turn, each have a set of characteristics (hard-stem, soft-stem, etc.)
- Each gender-case combination does not require a distinct suffix.
- Inflectional suffixes can also be present in particles, numerals, and adjectives.
- Adjectives agree in gender, number and case with the noun that they modify.
- Full adjectives inflect in case, gender and number; short adjectives - in gender and number.
- Adverbs, prepositions, and conjunctions are uninflected.

Since full adjectives inflect in case, number, and gender they have more inflected forms than nouns that do not inflect in gender. Table 3 presents all possible inflections for a full adjective новый/ 'novyj'/ new. Note that the 'masc.in' stands for masculine inanimate gender (ex: house) and 'masc.an' represents masculine animate/human gender (ex: man).

case/num	masc.in	masc.an	neut	fem	case/num	masc.in, fem, neut	masc.an
nom/sg	нов-ый	-ый	-ое	-ая	nom/pl	-ые	-ые
gen/sg	нов-ого	-ого	-ого	-ой	gen/pl	-ых	-ых
dat/sg	нов-ому	-ому	-ому	-ой	dat/pl	-ым	-ым
acc/sg	нов-ый	-ого	-ое	-ую	acc/pl	-ые	-ых
inst/sg	нов-ым	-ым	-ым	-ой	inst/sg	-ыми	-ыми
prep/sg	нов-ом	-ом	-ом	-ой	prep/sg	-ых	-ых

Table 3: Adjective Full Inflection

As shown in Table 3, the adjective *new* has fourty-eight possible gender-case-number combinations. However, in the given example only twelve forms are distinct. This example is good to demonstrate both the property of inflection, and the syncretism (reference section 2.1.2).

Agreement in a Sentence
It is important to note that in a sentence, the main verb agrees in person, number, and gender with the subject. Adjectives agree in number, case, and gender with the nouns they modify [49].

Stem Alternation
Another important notion in the Russian language inflection is stem alternation because of its importance for tagging, morphological analysis and generation. Russian, much like Spanish, has various stem alternations for nouns and verbs. According to [39], nouns with alternations in Russian have two stems (Table 4) and verbs with alternations have up to four, as demonstrated in Table 5 for the verb 'to eat' conjugated in Present Tense.

молоток-	молотк-а
molotok (masc.sg.nom)	molotka (masc.sg.gen)
"hammer"	"of hammer"

Table 4: Noun Stem Alternation

The notion shown in Table 4 is sometimes referred to as the *fleeting vowel.* Usually, the vowel 'o' is fleeting in genitive plural (or singular as in the example 4), but letter 'e' follows the same fleeting pattern in dative singular. For instance, 'отец' (father) becomes 'отцу' (to father), without an 'e'.

Ест-ь	est'	to eat
Я ем-	Ja em	I eat
Ты ешь-	Ty esh'	You eat
Он ест-	On est	He eats
Мы ед-им	My edim	We eat
Вы ед-ите	Vy edite	You eat
Они ед-ят	Oni edjat	They eat

Table 5: Verb Stem Alternation

Derivation
Stem alternation is not only typical for nominal inflection, it is also present in derivation, i.e., when a suffix is added to a stem to create a new word, which usually belongs to a different part-of-speech [28]. The new words are generally composed of a prefix, a stem, and a suffix.

спутник	кровавый
с + пут-ь + ник	кров-авый
prefix(с) + stem(пут-ь) + suffix(ник)	stem(кров-ь) + suffix(авый)
sputnik	krovaviy
with + path + 'nik'	blood + 'aviy'
"satelite"	"bloody"

Table 6: Derivation by Means of Adding a Suffix

Table 6 shows two examples of word derivation. A noun *satellite* is formed from a noun *path*, and an adjective *bloody* is derived from a noun *blood.* One example uses a prefix, when the other does not.

Synthetic Languages Both Russian and Arabic belong to the synthetic group of languages. Synthetic languages have a high interpretation-per-word ratio and usually represent morphologically rich languages. There exist two types of morphological arrangement: syncretism and agglutination [40].

2.1.2 Syncretism

Languages with the property of syncretism have a tendency to express several combination sets of grammatical categories by one flexion, and have unpredictable stem alternations. Syncretism is typical for Slavic languages such as Russian or Czech as well as Spanish and Portuguese [40]. This property can often be mistaken for inflection. For the purpose of this report, we define syncretism as a phenomenon of different inflected forms that are realized by the same surface form.

In Table 7 we can observe that the same surface form can correspond to several grammatical categories. The word ‘structures’ is shown in genitive singular, accusative plural, and nominative plural forms. The three inflected forms result in the same surface form - структуры.

анализ структуры	в эти структуры	эти структуры привлечены
analiz struktury	v jeti struktury	jeti struktury privlecheny
analysis structure (gen, sg)	in these structure (acc, pl)	these structure (nom, pl) involve
"analysis of the structure"	"into these structures"	"these structures are involved"

Table 7: Syncretism Example 1

Table 8 presents an even more interesting example of syncretism: the two surface forms represent two different parts of speech: a noun in genetive case and a verb in the past tense.

стали	стали
stali(N.gen.sg.fem)	stali(V.past.pl)
steel	to become
"of steel"	"became"

Table 8: Syncretism Example 2

Syncretism differs from agglutination by the number of surface forms per grammatical, syntactic or semantic change. A syncretic language has many meanings in one form, which creates ambiguity in the natural processing. In contrast, agglunative languages (section 2.1.3) often have one meaning in one surface form.

2.1.3 Agglutination

Agglutinative languages are languages whose morphology is mainly constructed by agglutinating phonetically unchangeable affixes to a stem. Such languages either have no stem alternations or the alternations are predictable. A specific type of agglutination exists in Arabic where it is a common feature of the language to append affixes to the stem. Unlike affixes in English, Arabic has a specific type of affixes called clitics. A **clitic** is a morpheme that has syntactic characteristics of a word but is phonologically dependent on a neighboring word [3] [64].

Take the example in Table 9:

كتبنا
katab+nA
wrote+we
"we wrote"

Table 9: Arabic Clitics - Example 1

[3]http://www-01.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsACliticGrammar.htm

The previous word consists of two main parts: *ktb* which is the verb "to write" and *nA* which is the pronominal clitic indicating that the subject of the verb is "we". More precisely, Arabic clitics can be divided into 5 groups and attach to base words in the following strict order [42]:

[QST + [CNJ + [PRT + [DET + BASE + PRO]]]]

- **QST:** the interrogative particle $\hat{A}$
- **CNJ:** the conjunctions *w* and *f* which mean "and" and "then" respectively. They attach to both verbs and nouns
- **PRT:** particle proclitics, some of which only attach to verbs and others only to nouns
- **DET:** the determiner *Al* which only attaches to nouns
- **PRO:** pronominal clitics which can attach to nouns (as possessives) or to verbs (as objects)

However, only up to 4 clitics can be attached to a single Arabic word (the determiner and the pronominal clitics never coexist on the same word). This is shown in Table 10:

أوبالقلم سيحاربهم؟
Â+wa+bi+Al+qalami   sa+yu+HAriba+hum
QST+and+with+the+pen   will+he+fight+them
"Will he fight them with the pen?"

Table 10: Arabic Clitics - Example 2

2.1.4 Compounding

Compounding is a way to form words by combining two or more lexemes into one unit that holds a specific meaning [24]. Such behavior is typical for German, Dutch, and Finnish, and is also possible, though not regular, in Russian and English. In Russian, compounds are often encountered in specialized fields, such as biology or physics.

ветрогенератор	электромагнитный
ветер+генератор	электро+магнит
vetrogenerator = veter+generator	jelektromagnitnyj = jelektro+magnit
wind+generator	electro+magnet
"wind generator"	"electromagnetic"

Table 11: Compounding in Russian

2.2 Syntactic Properties

2.2.1 Pronoun Dropping

A pro-drop language (pronoun-dropping) is a language in which certain classes of pronouns may be dropped from the sentence under certain conditions. Arabic can be classified under a subcategory of pro-drop languages called Null Subject Languages [4]. These languages allow subject pronouns to be freely dropped as explained in [33].

أنا قرأت الصحيفة
ÂnA   qarÂtu   Al+SaHyfħ
I   read   the+newspaper
"I read the newspaper"

Table 12: Arabic Pronoun Dropping

[4]as defined in Martin Haspelmath, *The European linguistic area: Standard Average European*, in Martin Haspelmath, et al., *Language Typology and Language Universals*, vol. 2, 2001, pp. 1492-1510

In the sentence in Table 12, the personal pronoun *Âna* is almost always dropped in naturally occurring Arabic text. The pronoun can be deduced from the conjugation of the verb "to read", the final "t" in the gloss (the "u" is only included to enhance readability while it is not present in the original Arabic word due to lack of vocalization, see section 2.3.2).

This property is not exclusive to Arabic as it exists in several other languages such as Spanish, Italian, Japanese, and Russian. It introduces ambiguity when determining the subject of a verb because it cannot always be determined from the conjugation of the verb. Moreover, this pro-drop feature is further complicated if the language is an order-free language; a sequence of Noun-Verb can either be Subject-Verb or Object-Verb.

2.2.2 Zero-Copula and Ellipsis

In linguistics, a **copula** is a word in a sentence that is used to link the subject with the predicate. An example of a copula in English is the verb 'to be' [10]. Copulative sentences in Russian have a special zero-form in the present tense, where the subject is directly linked to the predicate without a special indication of such relation, i.e. with the absence of the copula [14].

Он - учитель
On - uchitel'
He - teacher
He is a teacher

Table 13: Zero-Copula in Russian

The same phenomenon exists in Arabic. Although called **equational sentences** by Arab linguists, it is identical to zero-copula constructions in Russian. Table 14 shows two examples of equational sentences in Arabic taken from [42].

الكتاب جديد	الرجل في البيت
Al+kitAbu jadydũ	Al+rajulu fy Al+bayt
the+book new	the+man in the+house
"The book is new"	"The man is in the house"

Table 14: Arabic Equational Sentences

Elliptical constructions or ellipsis is a property that is present in the Russian language. Ellipsis is omitting of one or more words from a clause that are understood from the context of the remaining elements [10]. In Table 15, the verb 'bought' is omitted from the second part of the sentence because it is clear that the action for the entire clause is 'to buy' an object. Elliptical constructions are one of the biggest challenges in the formalization of natural language syntax in many languages [14].

Я купил рубашку, а он галстук
Ja kupil rubashku, a on galstuk
I bought a shirt, and he a necktie
I bought a shirt, and he bought a necktie

Table 15: Challenge in Elliptical Constructions

2.2.3 Order-Free Languages

A free word order language is a language that doesn't exhibit strict syntactic rules regarding the word order in a sentence. In Arabic, the primary word order is Verb-Subject-Object (VSO). However, both Subject-Verb-Object (SVO), and less commonly, Object-Verb-Subject (OVS) can be used in certain contexts [33]. This property is demonstrated by the example in Table 16 where the first sentence follows the VSO order while the second the SVO order.

دمّرت الحرب سوريا	الحرب دمّرت سوريا
dam~arati Al+Harbu swryA	Al+Harbu dam~arat swrya
destroy the+war Syria	the+war destroy Syria
"The war destroyed Syria"	

Table 16: Arabic Order-Free Structure

Russian, as well as Arabic, is a relatively order-free language, which complicates identifying syntactic relationships between the words in the sentence [73]. The example in Table 17 illustrates that the words in a sentence can appear in any order.

Phrase Variations	Word Order
1.Борис навестил Ивана	SVO
Boris navestil Ivana	
Boris (nom) visited Ivan (acc)	
"Boris visited Ivan"	
2.Борис Ивана навестил	SOV
3.Ивана навестил Борис	OVS
4.Ивана Борис навестил	OSV
5.Навестил Борис Ивана	VSO
6.Навестил Ивана Борис	VOS

Table 17: Six Accepted Word Orders in Russian

Even though all six orders are accepted, the study conducted in [54] shows that Russian native speakers have a preference for some word orders over the others. For instance, SVO, OVS and SOV are used more frequently and are considered more grammatically correct.

Parsing of order-free languages is more challenging than languages with strict word order such as English. In particular, differentiating between subjects and objects can be specifically hard in Russian and Arabic as they can be put in any order around the verb that connects them.

2.2.4 *Idafa* Construction

The *idafa* construction is a phenomenon specific to the Arabic language. We use the definition given in [42] where they define *idafa* as "a possessive/genetive construction relating two nouns: the first noun grammatically heads and semantically possesses the second". It is an important construct that is widely used in Arabic. Table 18 gives an example of *idafa*.

مدينة الياسمين
madyna$\hbar$u Al+yAsamyn
city the+jasmine
"The city of jasmine"

Table 18: Arabic *idafa* Construct

This construction poses a problem for computational linguistics as it requires special handling because the nouns are connected without the existence of any concrete connectors. Moreover, while the first noun does not contain the definite article *"Al"*, it should not be considered as indefinite because the second noun clearly defines it.

2.3 Phonological Properties

2.3.1 Phonetic Stress

In Russian language, all words have a phonetic stress and sometimes, the same surface forms of a word can have two meanings depending on where the stress goes.The stress is implicit and is not reflected in writing, unlike it is in Spanish, for example. Words with the same spelling but different stress are referred to as homographs [10].

сне́га	снега́
snega (gen.sg.masc)	snega(nom.pl.masc)
"of snow"	"(a lot of) snow"

Table 19: Phonetic Stress in Russian

Table 19 shows an example of homographs, two words with the same spelling, but a different meaning. Russian also has fake homographs. They are occurs because of the letter 'ё', which always has to be stressed in the word that uses the letter. However, in modern literature, the letter is often replaced by 'e'. In these cases, the phonetic stress is still there but it is not reflected in writing, sometimes generating two words with identical spellings but different meanings (refer to section 2.4.2 for word sense ambiguity). The example in Table 20 not only demonstrates the importance of the stress of letter 'ё' but also that two homographs can belong to different parts of speech.

берё́г(verb)	бе́рег(noun)
berjog (past tense.sg)	bereg(nom.sg.masc)
"kept"	"shore"

Table 20: Phonetic Stress in Russian: Fake Homograph

2.3.2 Vocalization

One very interesting problem with Arabic text is the lack of vowels. In traditional Arabic, there are 3 long vowels that are written as separate letters while their 3 short variants are written as diacritics over the preceding letter. However, these diacritics are dropped from most modern Arabic texts and hence the text is left almost vowel-less.

A statistical study done in [64] shows that only 0.5% of words in newspaper articles have some kind of diacritics. One special diacritic, the "Shadda" (which is the consonant gemination marker), accounts for more than 60% of these diacritics. It should be noted however that in some contexts, such as any extracts from the Quran or children's books, the text is usually fully diacritized.

This lack of vocalization in general-purpose Arabic texts clearly leads to an ambiguity problem; one written morpheme can actually be interpreted in many different ways, each with its own pronunciation, meaning and grammatical category. Take for example the sequence "ktb", [64] lists at least 5 different forms in which this sequence can be interpreted, depending on the pronunciation and the missing short vowels. Table 21 lists those forms.

ktb	كتب
kataba	he wrote
kutiba	it was written
kattaba	he made him write
kuttiba	he was made to write
kutub	books

Table 21: Arabic Vocalization Problem

However, this problem, which primarily affects part-of-speech tagging, is made easier by another property of the Arabic language. As some clitics only attach to a specific grammatical category (e.g. nouns and not verbs), we can disambiguate the word by the clitics attached to it. To continue with the same example, the sequence الكتب "Al+ktb" can only be interpreted as "the books" since the definite article "Al" does not attach to verbs.

2.3.3 Diglossia

Arabic exhibits a phenomenon called **Diglossia**. Diglossia is defined by [33] as a phenomenon in which two or more varieties of the same language exist side by side, each being used for specific purposes and in distinct situations. In Arabic, at least two such varieties can be distinguished; Modern Standard Arabic (MSA) and the many regional dialects. While the dialects are acquired by speakers natively at home and used in everyday life, MSA is learned primarily through formal education and used in more formal situations such as university lectures, news broadcasts and parliamentary debates. A more formal definition of diglossia can be found in [64].

This is different from languages that have different dialects, such as English, where the dialects and the standard languages are only slightly different but are generally governed by the same linguistic rules. To further demonstrate, consider Table 22 where the first sentence is written in MSA while the second is in the Levantine Palestinian Dialect (LD) of Arabic [22]:

لا يحب الرجال هذا العمل					الرجال بحبوش الشغل هدا			
lA	yuħibu	Al+rijAl	haðA	Al+σamal	Al+rijAl	bi+ħibw+š	Al+šuγil	hadA
not	like	the+men	this	the+work	the+men	like+not	the+work	this
"Men don't like this work"								

Table 22: Arabic Diglossia

As simple as the example in Table 22 is, the following differences can be noted:

- lexically, the word for "work" is *Al+σamal* in MSA but *Al+šuγil* in LD
- the negation function word is *lA* in MSA but it transforms into the clitic *š* in LD
- syntactically, while Arabic is a free-order language, sentences in MSA tend to follow the VSO order while in most dialects, subject-initial sentences (SVO) are more common
- the demonstrative determiner for "this" precedes the noun in MSA while it follows it in LD

This diglossic situation clearly hampers efforts for automatic processing of Arabic. The difference between MSA and the dialects makes it extremely difficult to develop one NLP tool that can be used on all of them. Moreover, even the dialects among themselves show significant differences that can be sometimes compared to those among Romance languages.

2.4 Semantic Properties

Semantics is the branch of linguistics concerned with meaning. Because the human brain is capable of mapping natural language to meaning with little effort, it may result surprising how difficult it is to automate this task. We attempt to illustrate some of the reasons behind this. In this section we will not particularly focus on russian and arabic but rather on challenges present in the english language, since state-of-the-art semantic parsing in english is still under work and facing these challenges.

2.4.1 Multiword expressions

A multiword expression (MWE) is a lexeme, composed by a set of lexemes that has properties which cannot be predicted from those of its individual components. It is estimated that MWEs occur as frequently in open text as single word expressions[5]. Because of this, recognizing MWEs is important for improving results obtained when retrieving information. Indexing and searching for the individual components of an MWE will most likely result in irrelevant answers, e.g. a search on *pop star* retrieving results relevant to *pop* and *star* separately is most likely not what the user was looking for[1]. Under the same logic, recognizing MWEs is crucial to obtaining the correct logic representation of a phrase when parsing it. In the example *he took his clothes off*, the correct semantic meaning would be *he undressed*, however identifying this logic correctly is not possible unless we are aware of the existence of the MWE *take off [clothes]*. Despite the interest in identifying MWEs and interpreting the meaning of them, these tasks continue to be a great challenge, since MWEs are structurally very diverse.

One of the challenges in handling MWEs is that they vary greatly between languages, and even regionally within a same language. For example if we wish to search for restaurants that prepare meals intended to be eaten else where, we would search for *take out* in American English, but for *take away* in British English, and *carry out* in Scottish English. Another challenge in MWE recognition is that it is common for new MWEs to be created as languages evolve. For example, the expression *carbon footprint* has only become popular since global warming has started concerning humankind. This evolution of MWEs make it difficult to keep MWE lexicons up to date. A major challenge in MWE recognition is that the components of MWEs must not necessarily appear contiguously in a phrase. For example, if we wish to know "Why did Guns n' Roses *kick* Steven Adler *out*?" we are not referring to the literal meaning of the word *kick*. Techniques for identifying MWEs vary for different types of MWEs constructions. Some MWE constructions such as noun compounds e.g. *air conditioner*, are relatively easy to detect, whereas others can be very difficult.

One type of MWEs which lead to important parsing difficulties are **light verb constructions** (LVC). These are constructions made up of a verb and noun complement, in which the contribution of the verb to the meaning of the expression is relatively small in comparison to that of the complement. In some cases the contribution of the verb is so light that the LVC can be paraphrased with the verbal form of the noun complement. For example, if we search the phrase "How to *give* a kiss?" the verb *give* is not actually the subject of interest, but rather the action of kissing. [5].

The state-of-the-art studies in information extraction and question answering that will be discussed in the following sections do not emphasize or provide details on how MWEs recognition is dealt with. Still, it is worthwhile to discuss the challenge of MWE recognition since it has a strong effect on precision of the remaining parsing stages and is still an open research area in Semantic Parsing.

2.4.2 Ambiguity

Ambiguity refers to a word or sentence having more than one possible meaning. As mentioned in previous sections, phonological properties such as phonetic stress and vocalization can lead to ambiguity (section 2.3). Solving ambiguities is deeply related to semantic parsing in that it requires finding the true meaning of text when there is more than one possible meaning.

Polysemes
Polysemes are words that are particularly difficult to dissambiguate because they are written the same and have different, but related meanings. For example, in the question "What country was Napolean *born in*?", if we only consider the relation *born in* we could either be talking about the place of birth or the year of birth. In order to disambiguate we need to look at other parts of the sentence.

Passive Verb Forms
In most English sentences with an action verb, the subject performs the action denoted by the verb. These sentences are said to be in the active voice. For example, in the sentence "John won the gold medal", the subject, John is the person who performed the action won. However, it is possible to change normal order of any active sentence with a direct object so that the subject is no longer active, but is instead, being acted upon by the verb or **passive**. In the phrase "The medal was won by John", the subject is the medal but is not performing the action. It is necessary to recognize these passive verb forms in order to correctly map the meaning of a phrase. This is specially true when both the action maker and the object receiving the action are capable of performing the action. For example, if we consider the active voice "Ben loves Amanda" and the passive voice "Ben is loved by Amanda", though they use similar sets of lexemes in a similar order, they actually have quite different meanings.

3 Language Resources and Knowledge Bases

This chapter is dedicated to available language resources in English, Russian and Arabic. By language resources we refer to datasets of language specific information which support language processing. The simplest form of a language resource is a dictionary, however, as we will discuss, there are much more complex forms.

As previously mentioned, most research in NLP has been focused towards the English language. However, due to the varying properties of world languages, the techniques used to parse English phrases do not always adapt to other languages. However, developing NLP techniques for different languages requires resources which are often unnavailable and costly to generate. As an alternative, researchers sometimes attempt to use resources of a similar language for processing. Language similarity between two different languages or two dialects of a given language could either benefit or hurt when it comes to the different NLP parsing levels. For instance, research shows that some Russian NLP researchers use the Czech language resources to improve their results. On the opposite end, Arabic dialects are so divergent from the standard language that they require additional efforts in NLP development.

3.1 Russian Language Resources

Challenges

Russian is a language spoken by millions of people, however, the research in NLP is not as advanced as one might think. There are several reasons that could account for a slow progress for NLP in the Russian language. Research is done by independent groups separately and there is not much communication between such groups. According to [16], the groups that inherited the Soviet tradition and the modern commercial laboratories work independently instead of defining a common standard or evaluating the state-of-art of the Russian parsers' performance. The first initiative to increase communication among researchers in Russia and to evaluate the methodology used for Information Retrieval of Cyrillic texts (ROMIP) was launched in 2002. Partially due to the ROMIP's initiative and yearly seminars that finally brought together the researchers, the first Gold Standard Corpus for the Russian language was developed in 2004. The initiative continues its work and involves students from the Moscow State University to further unite the research groups and improve the methodology for Russian text IR. In 2013, the following challenges have been identified as a part of a ROMIP seminar [27]:

- Absence of publicly available Russian text collections
- Low interest in creating Russian text collections
- Low participation rate of researches in the evaluation initiatives

The poverty of resources and more importantly the lack of interest in NLP research for Russian are the main drawbacks for the future of IR for Cyrillic texts. Currently the students are being involved in the initiatives to promote the interest, meanwhile, an interesting attempt of sharing resources among similar languages is taking place.

Sharing Resources

Russian and Czech are both Slavic languages and are similar in many grammatical aspects. The Czech language is much more developed in terms of the NLP resources freely available for use. Recently, it has become common practice for Russian researchers to build parsers for Russian based on Czech resources. Such methods will be further discussed in follwing sections of this report, specifically sections 4.2, 4.4, and 4.5. Hana et.al. [49] make use of Czech resources for tagging Russian and give the following arguments in terms of why Czech and Russian can use shared resources:

- both languages are Slavic (Czech - West Slavonic, Russian - East Slavonic)
- both languages have a rich morphology
- in both languages the main verb agrees in person and number with the subject; past tense verbs also agree in gender.
- in both, adjectives and nouns agree in gender, number, and case
- both are order free languages
- in both, the word order is defined by the discourse

Regarding the Russian morphology in comparison with Czech, Russian usually uses less morphological categories because it has six cases, when Czech has seven. In addition, Czech uses formal and colloquial forms of the same word. Table 23 shows the number of different tags used for tagging Russian text using Czech resources (further described in section 4.2). Note that the table has more cases for both languages than those usually described in the language grammar, but those were artificially introduced during research for better automatic morphological analysis. For a full table, refer to [49]

Description	Abbr.	#values(CZ)	#values(RU)
POS	P	12	12
Detailed POS	s	75	32
Gender	g	11	5
Number	n	6	4
Case	c	9	8

Table 23: A snippet from Russian and Czech tag comparison

3.2 Arabic Language Resources

Since the early 2000s, Arabic has seen increasing interest within the field of NLP. Perhaps the first major project to try to improve the quality and quantity of Arabic resources is the **Penn Arabic Treebank (PATB)** developed by the University of Pennsylvania[5]. The first version of this project was released in 2003 and included morphological and syntactic-level annotations for Arabic text.

Another project was launched around the same time at Charles University in Prague, the **Prague Arabic Dependency Treebank (PADT)** [6]. While the annotations at the morphological level are the same as in PATB, PADT uses dependency grammar instead of constituency to represent Arabic parsing trees. Moreover, they are planning to add an extra level of annotations, called the tectogrammatical level, which tries to capture the underlying syntax reflecting the linguistic meaning of a sentence [48].

Other treebank projects include the Columbia Arabic Treebank (CATiB), developed by Columbia University[7] while Bibliotheca Alexandrina in Egypt maintains a large corpus of Arabic documents without any treebank information[8].

Although resources for Arabic are abundant, the diglossic situation of the language creates a problem regarding resources. The dialects are hardly ever written and hence there's a considerable lack of resources when attempting to study them from a computational point of view. In most studies done so far on Arabic dialects, transcribed telephone conversations and TV interviews have been used as the main data source [22] [70].

3.3 Knowledge Bases

At a broad level, a knowledge base (KB) is a technology used to store complex structured and unstructured information used by a computer system. Knowledge bases are meant to represent facts about the world, and in particular, they are interesting for language processing because they can help understand the meaning of words and relations between known entities. In this section we present some of the better known and used KBs in the study of IR.

- **WordNet**
  WordNet is a large freely and publicly available lexical database. Information is organized and represented in sets of synonymous words called synsets, each representing one base concept or meaning. There are about 117,000 linked synsets[9]. The idea behind WordNet is shown in figure 1, where we have two verbs slow and speed and each of these verbs creates a synset with its associated synonyms. The two generated synsets are then linked together with the use of an antonym relation (i.e. the two synsets have the opposite meaning).

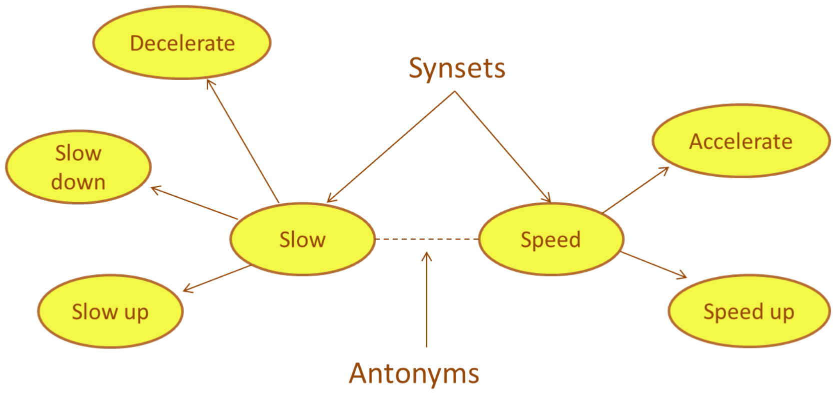


Figure 1: WordNet representation

[5]http://www.ircs.upenn.edu/arabic/
[6]http://ufal.mff.cuni.cz/padt/PADT_1.0/docs/index.html
[7]http://www.ccls.columbia.edu/project/catib
[8]http://www.bibalex.org/unl/frontend/Project.aspx?id=9
[9] http://wordnet.princeton.edu/

WordNet for other languages was also developed. For instance, EuroWordNet includes wordNets for English, Italian, Spanish, German, Dutch, Estonian, French, and Czech [74]. WordNet can be used for word sense disambiguation (i.e. the use of WordNet to archive disambiguation) as well as for automatic query expansion, where the WordNet semantic relations can be used to expand queries so that document searches are not confined to pattern-matching of query terms but also cover synonyms [74].

- **DBPedia**
  DBPedia [10] is a knowledge base that has been constructed by the automatic extraction of structured information from Wikipedia. This is a very powerful resource since Wikipedia has grown into one of the central knowledge sources of mankind, maintained by thousands of contributors collaborating together. DBPedia is publicly availbale and allows you to make sophisticated queries against Wikipedia. The important advantages that DBPedia offers over other knowledge bases is the large coverage it has over many different domains and the fact that it represents knowledge in many languages.

  DBpedia uses the subject-property-object triples (SPO triples) as a flexible data model for representing extracted information. Data from Wikipedia articles is transformed into RDF triples. This data is queried using SPARQL, a language with a syntax similar to that of SQL.

  The English version of the DBpedia knowledge base currently describes 4.0 million things, out of which 3.22 million are classified in a consistent ontology, including 832,000 persons, 639,000 places, 372,000 creative works (music albums, films, video games) and 209,000 organizations (companies, educational institutions). The full DBpedia data set features labels and abstracts for 12.6 million unique things in 119 different languages. The dataset consists of 2.46 billion RDF triples, out of which 470 million were extracted from the English edition of Wikipedia, 1.98 billion were extracted from other language editions, and about 45 million are links to external datasets such as YAGO.

- **Freebase**
  Freebase [11] is a collaborative, freely available knowledge base, licensed by Creative Commons. Freebase is structured as a graph; instead of using tables and keys like in conventional databases to define data structures, Freebase defines its data structure as a set of nodes and links that establish relationships between the nodes. Because its data structure is non-hierarchical, Freebase can model much more complex relationships between individual elements than a conventional database, and is open for users to enter new objects and relationships into the underlying graph.
  Freebase has over 39 million topics about real-world entities like people, places and things. These topics correspond to the nodes in the graph. Some topics are notable because they hold a lot of data (e.g., Wal-Mart), and some are notable because they link to many other topics, potentially in different domains of information. For example, abstract topics like love and poverty don't have many properties associated with them but they appear often as subjects to other entities like books or films.

  Any given topic can be seen from many different perspectives, for example, Bob Dylan can be seen as a song writer, singer, performer, book author, and film actor. In order to capture this multifaceted nature of topics, Freebase introduces the concept of types. Properties define the unique qualities of a given Type. Just as properties and topics are grouped into types, types themselves are grouped into domains. Domains follow the same logic as sections a newspaper like Business, Life Style, Politics, etc [12]. An example of a Freebase schema representing the fact that William Shakespeare wrote the play Hamlet is written as:

  William Shakespeare

is a	type	Author
has a	property	WorksWritten
with a	value	Hamlet

- **Yago2**
  YAGO is a huge semantic knowledge base, derived from Wikipedia, WordNet and GeoNames.

[10]http://dbpedia.org/About
[11]https://www.freebase.com/
[12]https://developers.google.com/freebase/guide/basic_concepts

YAGO currently contains more than 10 million entities (persons, organizations, cities, etc.) and more than 120 million facts about these entities [13]. YAGO tries to infer class memberships from rich Wikipedia category names, and has integrated this information with the taxonomic backbone of WordNet so far assigning entities to more than 350,000 classes[51] Like the RDF model YAGO represents facts as triples of subject (S), predicate (P), and object (O) (SPO).

YAGO2 is the current version of YAGO that combines the entity-relationship-oriented facts with the spatial and temporal dimensions. YAGO2 uses the enhanced representation model SPOTL for SPO, Time and Location. This model which can co-exist with SPO triples, but provides a much more convenient way of browsing and querying the knowledge base. This model is further improved by the inclusion of context forming a full representation model SPOTLX tuples (SPO + Time + Location + context)[51] Using the time and space dimensions, as entities, combined with context data users can for example can ask to identify actors who were born in the vicinity (i.e. at most 10 km away from) Berlin after the "German reunification": Given by the following query:

```
?p isA actor .
?p wasBornIn ?l nearby Berlin 10.0 .
?p wasBornOnDate ?d after "German reunification" .
```

In this example *"wasBornIn"* represents the space dimension entity while *"wasBornOnDate"* represent the time entity.

Having introduced the main language resources we procede to introduce different language processing levels in the following section.

4 Language Processing Levels

Natural Language Processing(NLP) techniques are often used in the Information Retrieval (IR) field. The benefit of using NLP in IR is nevertheless argued among the researchers who consider that document retrieval is not the perfect application for NLP [18]. In discussing different NLP techniques such as tokenization, part-of-speech tagging, morphological analysis, named entity recognition, and syntactic parsing we wish to present the reader with ideas of how these processing levels can support document retrieval. The following section illustrates the different levels of processing present in NLP and the current state of the art in English language processing. We analyze the additional challenges that are present when processing Arabic and Russian.

4.1 Tokenization

Tokenization is a common term in IR and NLP that refers to splitting streams of characters into tokens that are often also called terms [21]. It is the basis of most, if not all, NLP processes as an early step of processing the input document. Tokenization is crucial in bag-of-words approaches to IR since these depend on term matching between queries and documents. If a token in the query matches a token in the document, the document is more likely to be relevant [53].

Normalizing generated tokens into one canonical form is also beneficial for IR systems, despite the risk of decreased precision. The normalization can be achieved through stemming or lemmatization that both achieve the task of normalizing tokens that look similar and hold the same meaning into one canonical form [53] It is especially beneficial for inflectional languages like Russian and Arabic, but also morphologically poor languages like English.

Stemming usually refers to a crude heuristic process that chops off the ends of words and often includes the removal of derivational affixes. one of the most popular algorithms for English stemming is Porter's algorithm, which removes endings based on the longest-match principle [61]. *Lemmatization* refers to token normalization with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings and return to the base dictionary form of a word.[61]

[13]as defined in :http://www.mpi-inf.mpg.de/yago-naga/yago/

4.1.1 Tokenization in Russian

For both English and Russian, as well as most languages that use white spaces to separate words in a sentence, the process of tokenization is rather straightforward. The white spaces are used as delimiters to identify the tokens or separate words. A slightly more sophisticated approach is to use all non-alphanumerical characters as delimiters during the tokenization process [21]. Having said that, tokenization in English and Russian can still be tricky in the case of compounds or words separated by a hyphen, for instance. Compounds are a challenge for automatic NLP systems because they often are not listed in lexical sources, therefore recognizing and splitting them can benefit information retrieval along with other tasks [24].

Even though many languages have compound words, the approach of NLP systems to deal with the issue tends to be language specific. Table 11 shows an example of a compound word in Russian (ветрогенератор), that is formed by 'ветер' and 'генератор'. The problem of simply segmenting the compound word into two is that the two obtained tokens would be 'ветро' and 'генератор', where a commonly recognized token, 'ветер', is different than 'ветро'. Noun stem alternation discussed in 2.1.1 makes tokenization of compounds in Russian a challenge.

An interesting approach of combining statistical features as well as morphological features of a language in order to tokenize compounds was taken by [24]. A baseline experiment of just splitting while matching tokens to a dictionary was enriched with the usage of the corpus, similarity measures, and language-specific rule sets. A list of twelve transformation rules for Russian compounds was created with rules specific for the Russian language morphology. The use of the corpus combined with Levenshtein distance and a large rule set that included inflection treatment resulted in 92.24% of splitting precision of Russian compounds. Table 24 demonstrates how by the combination of recognizing the word from the corpus (магнитный), then applying one of the inflection rules in the rule-set, and comparing the obtained token with an existing token 'магнит', we obtain the correct result after splitting a compound in Russian.

электромагнитный → электро + магнитный
rule 11 магнитный → магнитн
similarity (магнитн, магнит) = 0.86
result: электромагнитный → электро + магнит

Table 24: Splitting Compounds in Russian

Stemming in Russian

Token normalization by applying stemming has proved to have a positive effect on Information Retrieval results. As described in section 2.1.1, Russian inflection is expressed through attaching different inflectional suffixes to stems. The approaches to stemming of morphologically complex languages differs from the English language stemmers. They require either a corresponding language-specific lexical stemmer, or an algorithmic stemmer. Research suggested by [28] describes two new stemming approaches for the Russian language that enhance the result of information retrieval systems. The results are then compared to one of the most popular stemmers, Snowball. The two approaches are called "light" and "aggressive". The "light" approach used a set of 40 rules to address the properties described in section 2.1.1 and a list of stop words. The "aggressive" approach used the same set of rules and the most common derivational suffixes. Both stemmers were tested and compared with other approaches, such as Snowball, 4-gram, and no stemmer at all. All performed better and improved IR results compared to using no stemmer at all. The "light" approach performed best among other stemming strategies even though the difference is not statistically significant [28]. Moreover, the results of the "light" stemmer increased retrieval performance by 90% in Russian compared to using no stemmer at all, which proves once again the importance of stemming Russian texts as part of the NLP for IR.

4.1.2 Tokenization in Arabic

The main problem in tokenizing Arabic text is the agglutinative nature of the language (see Section 2.1.3). While splitting on whitespaces is enough to obtain all tokens in English, the existence of clitics in Arabic requires further treatment of the text in order to obtain all tokens.

Depending on the application, different schemes of tokenization in Arabic can be used. As [42] specifies, a tokenization scheme is defined by the outcome of the tokenization process. They range from the most basic scheme where no decliticization is performed, to the most advanced where all of the groups of clitics are split off. Table 25 illustrates the 4 most common tokenization schemes and the outcome of each (a full list of all schemes can be found in [42]).

	وسينهي الرئيس جولته بزيارة إلى تركيا
	wsynhy Alrŷys jwlth bzyArħ Ǎlý trkyA
	and+will+finish the+president tour+his with+visit to Turkey
	"The president will finish his tour with a visit to Turkey"
D0	wsynhy Alrŷys jwlth bzyArħ Ǎlý trkyA
D1	**w+**synhy Alrŷys jwlth bzyArħ Ǎlý trkyA
D2	w+**s+**ynhy Alrŷys jwlth **b+**zyArħ Ǎlý trkyA
D3	w+s+ynhy **Al+**rŷys jwlt**+h** b+zyArħ Ǎlý trkyA

Table 25: Arabic Tokenization Schemes

We believe that in IR systems the most common tokenization scheme should be *D3* because it splits all clitics and hence gives as much information about the text as possible.

Another level of tokenization is introduced in [64]. The author calls it **segmentation** and distinguishes it from tokenization in that it also includes the splitting of inflectional affixes. For example, consider Table 26. The *At* suffix, which denotes feminine plural, is split off in segmentation but is left as part of the base word in the D3 tokenization scheme.

	وكمهندساتنا
	wkmhndsAtnA
	and + like + engineers(fem.) + our
	"and like our female engineers"
D3	w+k+mhndsAt+nA
Seg.	w+k+mhnds+At+nA

Table 26: Arabic Tokenization Schemes

While this segmentation might not have a clear benefit for IR, [64] argues that it is crucial for lemmatization which helps in grouping different forms of the same word under one form, which in turn is beneficial for languages with inflectional morphology such as Arabic.

E. Mohamed also conducted several experiments in [64] on a corpus extracted from the PATB. Instead of directly doing a *D3* tokenization, he performed a full segmentation of the words and then removed inflectional segment boundaries to revert words back to their *D3* form. He achieved accuracy levels of 99.36% but with a marked decrease on unknown words.

One noteworthy observation in [64] is the effect of automatically vocalizing the text before segmenting it. While his vocalization experiments yielded 93.3% accuracy, segmenting the vocalized text saw a drop of accuracy of more than 1 percentage points. This indicates that while vocalization might reduce the ambiguity of text, automatic vocalization does not necessarily help other NLP tasks.

The best results obtained in [64] are marginally better than the results reached a few years earlier by N. Habash in [44]. While using the same PATB corpora, Habash's tokenization experiments used a machine learning algorithm (SVM) with a vector of 10 features extracted from a morphological analyzer and reached an accuracy of 99.3%.

4.2 Part-of-Speech Tagging

Part-of-Speech (POS) tagging refers to assigning a part-of-speech to each word in a sentence. Some common tags are N (noun), V(verb), ADV (adverb), ADJ (adjective), DET (determinor), etc. However, there are different tagsets that are used among the linguists.

POS tagging is particularly helpful in IR because it brings the knowledge of different senses in which a word is used. For instance, a surface form "object" has two meanings in English: **ob**ject (N) and ob**ject** (V). If "object" is used in a query or found in a document, the IR system will be able to retrieve more relevant documents knowing which part-of-speech the word belongs to, and, therefore, which meaning it holds [75].

Part-of-speech challenges are language specific because different languages, based of their properties, have more or less structural ambiguity. English, for instance, uses articles, which make noun identification much easier, whereas other languages might not have this advantage [66]. Despite the obvious effect that the language structure has on POS tagging, currently a lot of work is done in developing POS taggers for multiple languages. The results for supervised POS tagging in English have reached approximately 97.3% accuracy and now the focus is on unsupervised techniques as well as multilingual and cross-lingual tagging. In order to accomplish such a goal, much research is devoted to developing a universal tagset that can be used for many different languages and achieve state-of-the-art results [69].

The research done by [69] (2012) suggests that having a universal set of tags does not hurt the accuracy of the results specific to the processing level of part of speech tagging. The researchers developed a set of 12 universal tags and mapped them to 25 different treebanks of 22 different languages (for some languages several treebanks were used). Mapping some tags was easier than others and there were many ambiguities in the tags across 25 treebanks, however, the universal tagset simplified and combined many categories into one. Moreover, when testing the performance of the tagset in a dependency parser, the results outperform all studied previously suggested universal tagsets. Table 27 shows the languages studied in this report and their performance with the original tagset from the corresponding treebank, and the performance with the newly developed universal tagset (note, that [69] provides data for all 22 languages).

Language	Source	# Tags	Original(O)	Universal(U)	O/U
Arabic	PADT/CoNLL07 (Hajic et al., 2004)	21	96.1	96.9	97.0
English	PennTreebank (Marcus et al., 1993)	45	96.7	96.8	97.7
Russian	SynTagRus-RNC (Boguslavsky et al., 2002)	11	96.8	96.8	96.8

Table 27: Performance comparison between original and universal tagsets

The 'Original' tab displays results for training and testing using the original tagset, similarly, the 'Universal' shows the results for training and testing using the Universal tagset. The last column of the table is for the results where training was performed using the original tagset, and testing - using the universal tagset. As you can see, the performance for Arabic improves by 0.08%, for Russian - remains the same, and for English only drops 0.06 percent. The minor decrease in the English language performance seems even less sufficient considering that the tagset reduced from original 45 to universal 12. This comparison was conducted by training a supervised tagger based on the trigram Markov model for the 25 treebanks [69]. It is important to note that having a universal tagset is a positive initiative, however simplifying the tagset makes tagging a simpler task, hence the improved tagging perfromance. Moreover, more fine-grained tagsets are more useful for more advanced processing levels.

4.2.1 POS Tagging in Russian

The challenges for part-of-speech tagging in Russian come from the linguistic properties fusion and syncretism (see section 2.1.2 and 2.3.1). These challenges are similar to the ones identified for English, however, the approach to treat them is language-specific: more complex for a language with a complex morphology.

The benefit of successful POS tagging in the IR field is obvious, since identifying the correct part-of-speech of the word could change the meaning of a user-query dramatically. In Russian, if the tagset can only distinguish between the most basic parts of speech without any additional morphological information, there would not be a way to tell apart polysemes, such as the word 'стали' as a noun formed from 'сталь' (steel) or as a verb in the past tense, 'стать' (to become). Two sets of completely different documents could be extracted by IR systems depending on the POS tag.

As a solution to the described problem, Russian taggers have to use some morpho-syntactic features (tense, case, number) in order to identify the part of the speech correctly. This approach is referred to as morpho-syntactic tagging, or extended POS tagging. According to [34], morpho-syntactic tagging is contextually disambiguous morphological analysis. Due to a significant similarity in the process of morpho-syntactic and morphological analysis, both are described together in section 4.4.

Developing a Russian Tagset

One of the biggest challenges of the Russian POS tagging is the resource unavailability as described in section 3, and particularly, the absence of a publicly available extensive tagset and a large annotated corpus. Multiple studies have been done using Czech tagsets and annotated corpora before the first syntactically tagged Russian corpus, SynTagRus was developed in 2002 [52], and the first Gold Standard annotated corpus in 2004.

Research done by [73] proposed an extensive tagset for Russian consisting of 600 tags and resulting in 95 percent accuracy on the disambiguated part of the Russian National Corpus (RNC). One of the objectives of this project was to create an appropriate tagset for Russian to avoid disambiguity in part of speech tagging, but also to be able to use the tagset for other similar Slavonic languages. That is why the morpho-syntactic features were selected using the MULTEXT-East framework that is suitable for many Slavonic languages. Two Russian morphologically annotated corpora had been taken into account when designing the tagset: RNC (137 labels) and HANCO (147 labels). The resulting tagset consists of 12 main categories with zero to ten attributes in each, resulting in 156 value-pairs (for more details see [73]).

Interesting results were obtained after tagging Russian text while using the designed tagset. A disambiguated part of RNC which contains around 5 million words was used to train three statistical taggers, while 10% of the corpus were left apart for testing purposes. The TnT tagger (an implementation of the Viterbi algorithm for second-order Markov models) performed the best with the state-of-the-art result of 95.28 percent. Moreover, some conclusions were made about the tagger's performance separately for verbs, nouns, and adjectives as well as different tasks, such as POS, gender, number, etc. This experiment proves that nouns and adjectives are the most difficult parts of speech to tag and are particularly challenging in gender and case [73].

Resource-light Approach to Tagging Russian
The taggers using Czech resources result in more moderate results, as described in [49], but they are considered resource-light approaches. Resource-light approaches for Russian are interesting since there is a lack of freely available Russian annotated corpora of a large size. Instead of using such an extensive resource, a pre-existing annotated Czech corpus and an unannotated Russian corpus are used by [49]. For the evaluation purposes, a small Russian corpus (1,758 words) was manually annotated and used in training and testing of the tagger. As a result, only a difference of 6 percent overall was noted between labeling Russian text by the use of a Czech morphological analyzer (73.5 percent) and when training a statistical analyzer directly on a small sample of manually annotated Russian corpus (79.7 percent).

4.2.2 POS Tagging in Arabic

The first question to be considered when developing a part of speech tagger is the tagset to be used. Several tagsets have been developed for Arabic including, but not limited to, Buckwalter Tagset, PABT Tagset, PADT Tagset and the Khoja Tagset. They mainly differ in the number of categories they include and hence each is suitable for a different type of application. [42] discusses in detail the most commonly used tagsets.

To demonstrate the effects of the tagset, Table 28 lists a few examples of different studies using different tagsets and the results they obtained. It should be noted that although it is a common understanding that a smaller tagset improves accuracy, the results in Table 28 may not be directly comparable due to the different techniques and corpora used.

Study	**Approach**	**Corpus (# tokens)**	**Tagset**	**# Tags**	**Results**
Aliwy, 2013 [4]	Master - Slaves	29 k	*N/A*	3,000	90.05%
Mohamed, 2010 [65]	Memory-Based	500 k	Habash-Rambow	15	96.41%
El Hadj, 2009 [29]	Hybrid (Statistical / HMM)	20 k	*N/A*	13	96%
Kulick, 2010 [57]	Regular Expressions	340 k	Reduced PATB	40	95.1%
Habash, 2005 [44]	SVM	120 k	Reduced PATB	15	98.1%

Table 28: Arabic POS Studies with Different Tagsets

There have been many attempts to implement part-of-speech taggers for Arabic using **traditional methods**. [65] implemented a Memory-Based POS tagger (MBT) using the PATB tagset and training data and ran experiments on both whole words and automatically segmented ones. The results were similar to what the author reached in [64] for tokenization; performing automatic segmentation of words does not improve the accuracy of tagging. One interesting observation was that nouns, adjectives and proper nouns were the most confused tags and this confusion resulted in almost 25% of erroneous tags. This is due to the nature of the Arabic language in which most proper nouns are directly derived from common nouns and where nouns and adjectives are often interchangeable [65].

Similarly, [9] uses an approach based on genetic algorithms to tag Arabic text. They obtain mediocre results (accuracy of 92%) on a training corpus of over 1 million tokens. Other approaches such as a statistical POS tagger ([29]) and a rule-based tagger ([71]) also failed to give results comparable to English POS taggers.

Hybrid approaches have also been used to tackle the problem of Arabic POS tagging. [4] implements a master-slave architecture that uses the output of a Maximum Match tagger (slave) to modify the probabilities of an HMM tagger (master). The architecture is also extended to include a Brill tagger as another slave and obtains slightly better result using a self-designed tagset (95.7%).

In another study, [47] uses a pipeline of a rule-based tagger followed by an HMM tagger where the erroneous output of the former is passed to the latter for correction. While they obtained a 97.6% accuracy, their experiment was conducted on Quranic scripts using a limited tagset of 33 tags specifically designed for Classical Arabic (CA).

To date, the most successful approaches to Arabic POS tagging have been those that perform the task **simultaneously with tokenization**. The approach was implemented by N. Habash in [44] where an SVM classifier was implemented on a set of 10 morphological features to help choose the "most appropriate" tag from a set obtained from a morphological analyzer. They obtained a score of 97.5% on a corpus extracted from PATB.

[35] further developed the previous system by substituting a Conditional Random Fields (CRF) classifier for the SVM used in [44]. Their aim was to decrease processing time since their version of the system was used as a pre-processing step for a machine translation system. We believe that this consideration (processing time) can also be taken into account for IR systems.

A similar approach was implemented in [57] who attempted to exploit the benefits of performing tokenization and POS tagging together without the use of a morphological analyzer. Instead, they used regular expressions to encode text and POS tag possibilities and used two types of matching, text-match and pos-match, on test data. They obtained an accuracy of 95.1% and also faced erroneous results when disambiguating the noun and adjective categories.

Nowadays, two of the most commonly used tools for Arabic tokenization and part of speech tagging are **MADA** developed at Columbia University [46] and **AMIRA** developed at Stanford University [26]. The main differences between the two tools are those of design. While MADA performs tokenization and POS tagging together based on the output of a morphological analyzer (similar to [44]), AMIRA incorporates much less linguistic knowledge and does tokenization and POS tagging sequentially based on the forms of the words. A detailed description of the tools is available in [46] and [26] respectively while [42] includes a thorough comparison.

4.3 Named Entity Recognition

Named Entity Recognition (NER) is one of the major NLP tasks in IR systems as most user queries include proper names that refer to real-world entities. In most western languages, specifically English, NER has been deeply researched and developed. The current state-of-the-art in English is focused on refining the categorization of the discovered named entities beyond the 4 main categories; person, location, organization and miscellaneous. This is due to the nature of the English language where proper nouns can be relatively easily identified. In other languages however, the task can be troublesome as will be discussed in this chapter.

4.3.1 NER in Russian

NER in Russian has not received much attention in the recent years. The first attempt to address the issue of NER in Russian was developed as an extension to GATE, a general-purpose NLP platform initially created for English and then extend to a number of other languages. It uses six classes: date, person, organization, location, percent, and money. The performance reached by GATE was an F-measure of 71 percent, F-measure being a statistical feature that combines precision and recall. However, there is no system developed that only focuses on NER recognition in Russian. An initiative by [37] introduced two baseline approaches:

- a knowledge-based approach with a set of dictionaries and manually-compiled rules;

- a statistical approach using a machine-learning system (linear-chain conditional random field or CRF);

The knowledge-based approach resulted in the F-measure of 62.17 percent and the statistical approach - 75.05 (above the state-of-art result).

NER in Russian is definitely less studied than in English, however, considering the results from [37], the approaches used for English could be also relevant to Russian NER. This due to the fact that the statistical result, which has not used any specific to Russian grammar rules, performed better than the knowledge-based approach. Despite the relatively low results in Russian NER, it is still less challenging than Arabic, where proper names are not capitalized in text.

4.3.2 NER in Arabic

As for NER in Arabic, it faces several problems due mainly to language-specific properties: the absence of capital letters and the inflectional nature of Arabic (which also applies to proper nouns) [8]. Moreover, given that Arabic is written with its own script, many transliterations of the same foreign noun can be found in texts [72]. Table 29 shows 4 different ways to write "Los Angeles" in Arabic, all extracted from online newswire articles [72].

Los Angeles		
لوس أنجليس	lws	Ânjlys
لوس أنجلوس	lws	Ânjlws
لوس أنجيلس	lws	Ânjyls
لوس أنجيليس	lws	Ânjylys

Table 29: Different Arabic Transliterations of "Los Angeles"

To the best of our knowledge, the most advanced system to tackle the problem of NER in Arabic is **ANERSys** which was developed in the Universidad Politécnica de Valencia in 2007. The first version [8] used an approach based on Maximum Entropy and included the building of a corpus (ANERcorp) and several gazetteers (ANERgazet) compiled from several online resources. With a precision of 63.21% and a recall of 49%, the first experiments on the system were not satisfactory.

The same team followed up with a second version in the same year [6]. They recognized the main source of errors of the first version as being multi-word NEs and proposed to tackle this problem by performing NER in two steps. The first step included recognizing the beginning and ending boundaries of the NEs separately from each other and using POS information to merge the tokens of the multi-word NE. After completing that, the second step, which is based on Maximum Entropy as well, would classify the detected NE into one of the main categories (person, location, organization and other). The results in these experiments [6] showed a noticeable improvement reaching an F-measure of 65.9%, more than 10 points greater than the first version.

Further improvement on the system was done in [7] when the Maximum Entropy model was replaced by a Conditional Random Fields (CRF) model. The experiments, conducted on ANERcorp (which by then included more than 150,000 annotated tokens), showed further improvement with the F-measure reaching 79.2%. The most relevant feature in training the model was found to be the POS tag, followed by the Base-Phrase Chunk (BPC).

Another system that we would like to overview here is **NERA**, developed in 2009 by the British University in Dubai [72]. The system follows a rule-based approach using NE indicators (triggers) to compensate for the lack of capital letters in Arabic script. It also includes a gazetteer of NEs, "whitelist", and a "blacklist" to filter matches that appear after NE indicators but that are not valid NEs.

Moreover, the system was developed for the aim of being used in information extraction and hence it divides NEs into 10 different categories (e.g. date, phone number, price) [72]. Their experiments reached F-measures between 85% and 98%, depending on the NE category with the weakest categories detected being company, location and person.

They followed up their research in 2012 by including a Machine Learning (ML) component into the system [68]. In this new version, the output of the rule-based component is now fed to an ML component for refinement. This ML component included different sets of features for different NE categories with the most common features used being POS tag, the decision of the rule-based component, and several morphological features extracted from MADA [46]. Three classifiers were trained in order to make a comparison, Decision Trees, SVM and Logistic Regression.

The experiments were done on several corpora, including ANERCorp developed in [8]. The obtained results greatly surpassed those of the experiments conducted in [7], reaching an F-measure of 94.4% for the person category. Future research is planned on increasing the size of the gazetteers, refining the rules of the rule-based system and experimenting with other classifiers [68].

4.4 Morphological Analysis

Morphological Analysis is done as part of the pipeline of NLP tasks in several applications such as information retrieval and machine translation. In an IR context, the most commonly used morphological analysis is lemmatization of words in order to extract roots. Moreover, the output of a morphological analyzer can be used to facilitate other NLP tasks such as POS tagging, as explained in Section 4.2.2.

While this task is fairly complicated in inflectionally poor languages such as English, it is even more so in inflectional and agglutinative languages. More precisely, inflectional languages tend to have several stem alternations while the main problem in agglutinative languages is splitting the agglutinated form into its components and lemmatizing each of them.

4.4.1 Morphological Analysis in Russian

Russian NLP is at the stage of developing automatic systems for morphological analysis and detailed POS tagging. Developing such systems is difficult due to the rich morphology, high inflection, wordform ambiguity, and stem alternations.

Due to above listed reason morpho-syntactic tagging uses a tagset of about 600-900 tags (different number in different studies) in order to express appropriate morphological features, compared to the 50 tags that are enough to describe English morphology [34]. Full morphological analysis requires even more resources than morpho-syntactic features because of the stem alternations: many rules have to be developed in order to be able to predict all possible stem alternations for a word. Such rules often have poor documented linguistic interpretation, which leaves a lot of ambiguities in a language's grammatical paradigms [39].

These rule sets are necessary for a direct approach to morphological analysis. The rules are artificial and non-intuitive for native speakers and create a new class for every paradigm in the language: 1000 in Russian and 1500 in Czech [40]. Due to the large rule set necessary for traditional or direct morphological analysis, a new type of analysis through generation was proposed by [39].

Morphological analysis through generation is a simpler approach because it uses the morphological models that already exist as part of natural language grammar. The number of such for Russian is 40 classes, compared to 1000 artificial classes (a class is refered to a set of flexions that uniquely identify a stem). A static method of processing stem morphemes is used in the method. Even though stem alternation is a challenge, only about 30 percent of Russian words have alternation, and the maximum alternations a word could have is four. Hence, the static approach of saving all possible stem alternations in a dictionary is not going to increase its size significantly [40]. Analysis through generation is much simpler because instead of coming across with молотк- and having to come up with молоток- it does the inverse. In order to identify that молотк- is a form of молоток- several complex rules need to be applied, which are highly unintuitive to the native speaker, and have many exceptions. On the other hand, when we have молоток-, by applying a single rule, we know that in genitive singular it changes the stem to молотк-, which makes the task rather straightforward. The method works this way:

- an input wordform is detached from all the known suffixes and flexions;
- the remaining part of the word is considered a stem and checked against the dictionary;
- analysis is successful if the stem is found in the dictionary and the detached suffix(es) correspond to the saved grammatical information.

Since the number of morphological classes is low, the algorithm is much easier to implement than the direct morphological approach [39].

Detailed POS tagging and Morphological Analysis are important steps for processing of any morphologically rich language, such as Russian. These NLP levels also proved to be beneficial for information retrieval. Along with stemming, POS tagging and morphological analysis are used for a more efficient way to index documents. It is currently difficult to find the state-of-the-art results for morphological analysis effectiveness in IR, however, some results were found for a domain specific study for indexing medical texts [23]. Medical terminology in Russian is said to be morphologically complex, hence stemming techniques discussed in 4.1.1 are not sufficient. Morphological analysis is used to analyze the terms and helps the indexing process for IR. The study [23] suggests a 10% improvement in mean average precision of the IR systems that use morphological analysis. The study suggests that the improvement in precision depends on the domain of the language, the query length, and the document length, however, can bring great results for specific terminology texts.

Similar to how the POS tagging is often combined with morphological analysis due to the rich morphology of the Russian language, some researchers propose combining the two with syntactic parsing. Such combination of the three NLP levels demonstrates a better performance than the state-of-the-art results for all studied languages, including Russian [15].

4.4.2 Morphological Analysis in Arabic

For many years, morphological analysis in Arabic has been extensively studied since Arabic is a morphologically rich language with a high degree of inflection. Several approaches have been used since the early 2000s to build morphological analyzers for Arabic. A comprehensive survey of these approaches, categorized by the main technique used, is done in [2].

In this section, we will review three of the most used morphological analyzers for Arabic, highlighting their key implementation details as well as their strong points and drawbacks. It should be noted that while these systems also have morphological generation capabilities, we will not mention them here as it is of little interest to IR systems.

1. **BAMA: Buckwalter Arabic Morphological Analyzer**
   BAMA was developed in 2002 by Tim Buckwalter [14]. While the two first releases are publicly available, the third is available through the Linguistic Data Consortium (LDC) [42]. BAMA is used in both PATB and PADT and is still to date one of the most popular morphological analyzers.

   BAMA is based on a simple assumption that Arabic words are composed of three parts: *a*) a prefix (0-4 characters), *b*) a stem (1-infinite characters), and *c*) a suffix (0-6 characters)

   It uses this assumption to generate all possible segmentations of an input token. Afterwards, it uses dictionary lookup to check if the segmented prefix, stem and suffix are valid entries. If all three are found to be valid, their morphological categories are used to check if they are compatible together or not.

   BAMA uses a form-based approach, as it depends on the form of the word in the analysis. This approach also falls under the *Table Lookup Approaches* in the categorization proposed by [2]. Its main advantage is its simplicity of use but it has several drawbacks.

   Most notably, [3] notes that BAMA does not recognize the Arabic broken plural (instead categorizing the words as singular feminine) and confuses the starting *"Al"* in some words with the Arabic definite article (e.g. *AltzAm* التزام). BAMA also only performs stemming without extracting roots as its lookup dictionary contains many forms of the same lemma.

2. **MAGEAD: Morphological Analysis and Generation for Arabic and its Dialects**
   N. Habash and O. Rambow proposed in [45] MAGEAD, the first morphological analyzer and generator that addresses the Arabic dialects explicitly as well as the first system to use a linguistic knowledge representation to process multiple variants of the same language family. The system was implemented using FSA technology and as such can be used for both analysis and generation.

   In order to achieve this goal, a dictionary lookup cannot be used since including the lexicon of all dialects is virtually impossible. Moreover, there is not standard writing convention for the dialects and hence the system cannot depend on the surface form of the words. Hence, an abstract lexeme-and-features representation was used in the system.

   In general, the system includes five tiers of representation: *a*) pattern and affixes, *b*) root, *c*) vocalization, *d*) phonological representation, and *e*) orthographic representation. When analyzing a surface form token, the orthographic representation is converted to a phonological one using a set

[14]http://www.qamus.org/morphology.htm

of rules (53 for MSA). This phonological representation is then vocalized with all possible combinations (a very limited set since Arabic only has 3 short vowels). The root is extracted by using another set of rules (69 for MSA) and finally the pattern and affixes are extracted.

As can be concluded from the above description, adapting this system for use with different Arabic dialects can be achieved easily. As a matter of fact, the authors adapted the system for use with Levantine Arabic (LA) to demonstrate its capabilities. The main necessary changes were rewriting some (only 2) conversion rules between phonological and orthographic representations and adding two features to the lexeme-and-features representation. As reported in [45], the whole conversion was done by a native speaker linguist in 6 hours.

3. **Sakhr:**
This morphological analyzer and generator is developed by Sakhr software company [15]. It performs single word analysis extracting the root, affixes, morphological features and part of speech tag of the surface word [3]. While it is very popular for use in Arabic NLP solutions, no details about the implementation of their software have been published. This product was only included for purposes of demonstrating the increased interest in Arabic NLP.

4.5 Syntactic Parsing

While syntactic parsing is not traditionally thought of as an important NLP task for IR systems, it is increasingly becoming so with the rapid development of parsing engines. Syntactic parsing can help improve the accuracy of an IR system by giving information not present in the previously mentioned NLP tasks because understanding the structure of a sentence is the first step towards understanding its meaning.

Syntactic parsing often combines multiple NLP tasks, such as tokenization, POS taggings, and morphological analysis and provides information about how the words are linked within a sentence. As with other sub-fields of NLP, syntactic parsing is generally more advanced and researched in English than in other languages. In this section we will discuss the language-specific problems of syntactic parsing and the proposed solutions to overcome them.

4.5.1 Syntactic Parsing in Russian

Syntactic parsing has been addressed by many researchers in the last twenty years. Most of the approaches were statistical and when applied multilingually performed rather poorly for morphologically rich languages with a relatively free word order (section 2.2.3), such as Russian. In 2006 and 2007, CoNLL shared tasks on multilingual dependency parsing were organized, where the group of morphologically rich languages showed the lowest parsing accuracy among all [15]. Ever since, much attention has been devoted to studying the reasons of such poor performance and finding ways to improve parsing accuracy. According to [15], one possible reason for low parsing accuracy on highly inflective languages is the separation of morphological analysis and syntactic parsing, hence the authors proposed a joint method for morphological analysis and dependency parsing to evaluate parsing accuracy. References to other papers that tried similar approaches can be found in [15], like the joint method for part-of-speech tagging and dependency parsing or the morpho-syntactic joint method used in constituency-based statistical parsing. The results show that the joint morphological analysis and syntactic parsing, a rule-based morphological analyzer, and word clusters all contribute to an increase of parsing accuracy of all studied languages, including Russian, Hungarian and Finnish (the last two being agglutinative languages like Arabic) [15]. Russian, specifically, reached an accuracy above the state-of-art at that time in POS tagging, morphological analysis, and lemmatization: 98.9, 95.7, and 96.6 percent respectively [15].

An evaluation of Russian syntactic parsers took place as a part of RU-EVAL forum in 2012 where eight teams (companies or educational institutions) submitted their results for evaluation. The evaluation committee identified the following as the main features in Russian parsers [38]:

- they are mostly based on dependency tree representations;
- they are rule-based;
- they have no uniform annotation scheme between the systems;

[15] http://www.sakhr.com/index.php/en/

- many formalisms are used to recognize semantic-syntactic relations (in order to reduce the free word order issue);

As the result of the evaluation, the two best performing parsers were developed using a manual rule-based approach and a semantic component developed by a team of linguists: ABBYY (precision of 0.952) and ETAP-3 (precision of 0.933). The average of all eight parsers was 0.888, which is a relatively high result considering the complexity of the Russian language.

Importantly, the main challenge of parsing the Russian language: relatively free word order, has been overcome by a combinations of including semantic components and integrating statistical approaches into the rule-based systems [38].

4.5.2 Syntactic Parsing in Arabic

Syntactic parsing in Arabic has seen increased interest from the NLP community in the last few years. As with other NLP tasks, it suffers from major performance shortages due to the lack of sufficient resources as well as the nature of the Arabic language. More precisely, the main problems in parsing Arabic are [41]:

- lack of vocalization,
- difficulty in distinguishing between nouns and adjectives,
- the *idafa* construct (see Section 2.2.4),
- the length of Arabic sentences, which usually exceeds that of English and other European languages.

[42] names three parsers as the "state-of-art-parsers in parsing Arabic". We will review each of them next.

1. **Stanford Arabic Parser** [16]
   Stanford parser was implemented in 2002 at the Department of Computer Science at Stanford University. The principal model was proposed by D. Klein and C.D. Manning in [55] and is based on the A* parsing algorithm.

   While the first model was based on the constituency grammar developed for the English version of the parser, this proved to be insufficient for Arabic and gave poor results. S. Green and C.D. Manning try to improve the performance of the Arabic parser in [41]. They added language-specific features of Arabic to the parser in order to better parse constructs that do not exist in English. Namely, they handled equational sentences and the *idafa* construct by explicitly tagging sentences as such.

   They ran their experiments on PATB and while their results did not surpass those of Berkeley's parser, they saw an improvement of more than 5 percentage points over the previous version [41].

2. **Bikel's Parser**
   Daniel Bikel presented this parser in his doctoral dissertation in 2004 [13]. His main motivation behind building a new parser was that most existing parsers were too dependent on English and on PTB. His model is based on constituency grammar with each phrase being divided into a head and a body and offers complete language independence with "language packages" that include language-specific dependency rules. His dissertation included experiments on Chinese, Arabic and Portuguese and his parsing of Arabic achieved an accuracy of 72.5% [13].

   His work was followed in 2006 by S. Kulick et. al. [58]. They ran experiments on English using a training set of a similar size to the PATB and still obtained better results; eliminating the argument that the unavailability of Arabic resources was damaging the parser's performance.

   Moreover, they tried to improve Bikel's results by handling linguistic phenomena specific to Arabic by improving the tagset mapping from English and handling the *idafa* construct and equational sentences. They managed to improve the results obtained in [13] by almost 6 percentage points using the PATB corpus.

[16] http://nlp.stanford.edu/projects/arabic.shtml

3. **MaltParser** [17]

   Another approach to language-independent parsing came in 2006 by J. Nivre et. al. [67]. Unlike the first two parsers, this parser is based on dependency grammar. It implements a data-driven approach in which the system uses a set of lexical and morphological features to learn a grammar from a treebank and does not need any pre-written grammar rules.

   While the first experiments with MaltParser did not include Arabic, it has since been used in several studies on Arabic parsing. [64] implemented a pipeline for processing Arabic text starting from tokenization and ending with a dependency parsing using MaltParser. He obtained a parsing accuracy of 73% noting that this can be improved using gold-standard POS tagging.

   It is worth mentioning that [64] names the advantages of using dependency rather than constituency parsing to be that dependency relations are closer to semantic relations and are more suitable to languages with variable word orders such as Arabic.

   Y. Marton et. al. also conducted an extensive experiment with MaltParser in [62]. They configured the parser to use the Columbia Arabic Tree Bank (CATiB) as the treebank source, SVM as the learning algorithm and a set of lexical and morphological features extracted from MADA as the feature model.

   The best obtained results in [62] reached an accuracy of 80.45% and the most beneficial features were determined to be the determiner, number, gender and person features. It also found that while the case feature was the most important on gold-standard POS tagging, it actually harmed the performance of the parser with predicted POS tags (since singular nouns inflect for case by changing vocalization only. See Section 2.1.1)

   It should be noted that while MaltParser's ability to operate on treebanks without hand-written grammars is generally considered as an advantage, it would render it virtually impossible to extend its use on the rarely-written Arabic dialects which suffer from a severe lack of resources.

4.6 Semantic Parsing

Semantic parsing is understanding the logical meaning of natural language phrases. This subject is becoming increasingly interesting in the context of question answering and information extraction. For these applications, a surface level, bag-of-words oriented NLP, does not suffice. It is necessary to derive enough meaning from natural language input in order to obtain precise answers [36].

First approaches of semantic parsing used highly supervised learning techniques over specific domains. In these approaches, large amounts of domain specific labeled data are used to train models capable of parsing input related to the same domain. However, these models suffer from two shortcomings: (1) labeled data for training are costly to obtain and (2) modern applications require extracting information from open domains and vast data repositories [31]. Because of this, research in semantic parsing is now evolving towards less supervised models, capable of parsing open domain phrases.

4.6.1 Shallow vs. Deep Semantic Parsing

In semantic parsing we try to understand the meaning of free text. Though syntactic levels of processing help to understand the surface of a phrase, it is only through semantic analysis that we can fully interpret what is being stated or asked for. Shallow semantic parsing involves a surface level interpretation of meaning, mostly relating meaning to words or sets of words. There are two main tasks associated to semantic parsing:

- Text chunking consists of dividing a text in syntactically correlated groups of words. Though this task is more related to syntactic parsing, we will discuss it in this section, as it is generally a step towards full semantic parsing. The idea is to subdivide sentences into clusters called chunks according to prosodic patterns and pauses in reading. The following step is to construct a parse tree where the content words are usually the major heads of phrases. The structure of chunks can best be described by a context-free grammar. In figure 2 we see an example of chunking of the phrase *i.e. [the bold man] [was sitting] [on his suitcase].*

- Semantic Role Labeling aims at assigning a semantic role to constituents of a sentences. It involves the detection of the semantic arguments associated with the predicate or verb of a sentence and

[17] http://www.maltparser.org/

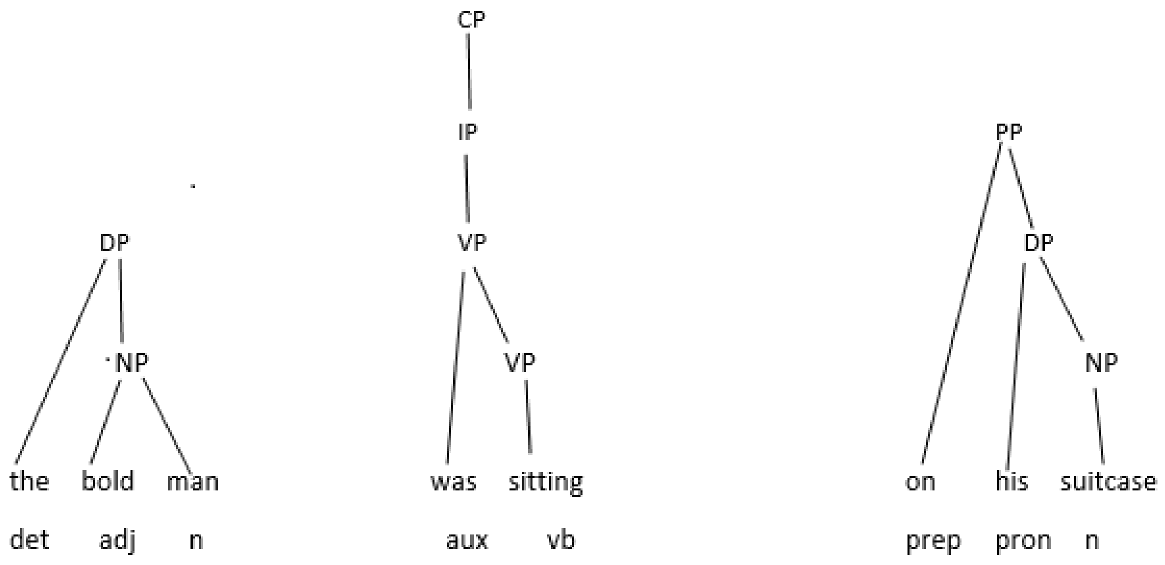


Figure 2: Shallow parsing: chunking (Extracted from: http://kontext.fraunhofer.de)

their classification into their specific roles. Recognition of semantic roles for the English language, based on PropBank [50] predicate-argument structures, where given a sentence we analyse the propositions expressed by some target verbs of the sentence.

Example Semantic Role Labeling

"The cat ate the mouse."
The meaning representation of the phrase is given as
Ate (cat mouse)
Result
Ate: verb (action)
Cat: Agent (entity performing the action)
Mouse: Patient (entity affected by action)

In a novel approach to shallow parsing, Collobert et. al. propose a technique which combines four NLP tasks that had previously only been considered separately: POS, chunking, Named Entity Recognition and Semantic Role Labeling [25]. One of the main drawbacks in using parsing techniques in real time search engines is that they tend to be computationally expensive, affecting response times. Instead of first extracting a rich set of features from a sentence and then feeding them to a classical shallow classification algorithm, they propose a deep neural network architecture which receives an input sentence and extracts features for words in a first layer and then features for windows of words. The features in the layers of the network are automatically trained by backpropagation. In their approach, Collobert et. al. compare the performance of their so called "almost from scratch" NLP tagger with state-of-the-art techniques for each of the four tagging processes, finding satisfying performance with respect to each of them.

Deep parsing seeks to provide meaning to a whole sentence, rather than only to parts of it. This task is more computationally expensive than shallow parsing, since the second can generally be approached using statistical methods. Deep parsing generally makes use of linguistic grammars or sets of syntactic rules which govern the composition of lexical items (words or phrases). Linguistic grammars rely basically on two main methods: the unification-based approaches and the lambda-calculus approaches [79].

4.6.2 Unification-based parsing approaches

These approaches rely on representations called feature structures that express valid phrase structural rules between syntactic categories and use lexical entries as their content. These rules guarantee the correct unification of all the semantic features with instances that respect the syntactic features. Phrase structure rules commonly operate according to the constituency relation, and hence govern constituency grammars, however this is not always the case. Unification takes in two feature structures as arguments and returns a merged feature structure if they are compatible, otherwise the return a failure [74].

Example Unification operator

Simple unification operator for performing an equality check.

[NUMBER PL] $\sqcup$ [NUMBER PL] = [NUMBER PL]

Success due to same value for NUMBER feature

[NUMBER PL] $\sqcup$ [NUMBER []] = [NUMBER PL]

Success due to compatibility of the structures (i.e. unspecified value in one structure, can successfully be matched with any value in the corresponding feature in another structure).

[NUMBER PL] $\sqcup$ [NUMBER SG] = [NUMBER PL]

Failure due the NUMBER features of the first and the second structures having incompatible values.

A particular and commonly used grammar within these approaches is Head-Driven Phrase Structure Grammar (HPSG) in which the formalism is based on lexicalism. That means the lexicon is more than just a list of entries; it is in itself richly structured and individual entries are marked with types. The IBM Watson project team developed a parser which combines shallow and deep parsing using an HPSG-style grammar. This project has received a lot of attention because they have successfully built a parser capable of playing and beating the best human players at Jeopardy!, a quiz competition in which contestants are presented with general knowledge clues in the form of answers, and must phrase their responses in question form. This problem requires deep parsing and understanding the subject of the input clues which may not even be obvious for a human reader. An example of a Jeopardy! clue is "Chandeliers look great but nowadays do not usually use these items from which their name is derived"[63].

For deep semantic parsing in Watson, a Slot Grammar was used, in which a lexicon associates lexemes with so called *complement slot frames* which can be complements or adjuncts to the lexeme [63]. When parsing an input sentence, each word is associated to a list of complement slots, filled in by other words of the sentence. This association generates a dependency structure which is interesting since HPSG is generally used to define constituency relations. In particular, for verbs, the first slot is filled by the logical subject of the verb, that is, the subject if the verb were in the active form, even if the input grammar is in the passive form. This means Watson is able to overcome this aforementioned challenge (see section 2.4.2).

4.6.3 Lambda-calculus based parsing

These approaches use lambda-calculus as the glue to combine semantic representations. Lambda-calculus consists of a single transformation rule (variable substitution) and a single function definition scheme. The advantage of using lambda-calculus lies in its generality. The meanings of individual words and phrases can be arbitrary lambda-expressions, while the final meaning for a sentence can take different forms.

One of the most frequent grammars used to map natural language to lambda-calculus is Combinatory Categorical Grammar (CCG). A CCG grammar includes a lexicon Λ with entries like the following:

$$\begin{aligned}
&NewYork \vdash NP\, ny \\
&borders \vdash (S\backslash NP)/NP : \lambda x \lambda y.next\_to(y,x) \\
&Vermont \vdash NP\, vt
\end{aligned}$$

where each lexical entry $w \vdash X : h$ has words w, a syntactic category X, and a logical form h expressed as a lambda-calculus expression. CCG combines categories using a set of combinatory rules, for example forward(>) and backward(<) *application* rules:

$$X/Y : f \quad Y : g \rightarrow X : f(g) \qquad (>)$$
$$Y : g \quad X\backslash Y : f \rightarrow X : f(g) \qquad (<)$$

These rules provide a relaxed notion of constituency. Each step in the parse applies a combinatory rule, until one logical representation of the input is created. Figure 3 is an example of a CCG parse using the previous examples for lexicon and combinatory rules [60].

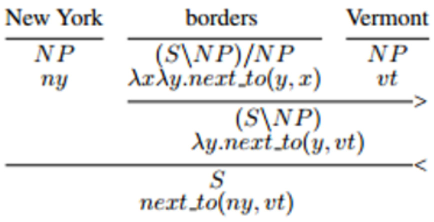


Figure 3: An example CCG parse obtained from [60]

Due to the generality of lambda-calculus approaches, parsing tends to over-generate logical representations given one input sentence. Once a set of logical expressions is generated, feature models can be used to select the most accurate representation. The parameters of these optimization functions are usually learned through labeled training data.

There is increasing convergence in methods and objectives between deep, grammar-based parsing and shallow, statistics oriented parsing. On the one hand, as data-driven, statistical methods become more sophisticated, they become capable of learning structured representations and multiple layers of annotations more accurately [25]. On the other hand, many of the current deep systems have statistical components: either as pre- or post-processing to control ambiguity, or as means of acquiring and extending lexical resources, or they even use machine learning techniques to acquire deep grammars automatically[19].

5 Open Domain Information Extraction

In this section we will cover state-of-the-art Information Extraction techniques and their relation to Semantic Parsing. We describe Information Extraction as the automatic extraction of structured information from unstructured machine-readable documents. It is different from Information Retrieval in that the latter retrieves unstructured data satisfying a given need. The following citation borrowed from [31] intends to ilustrate the motivation behind Information Extraction:

> "Ever since its invention, text has been the fundamental repository of human knowledge and understanding. With the invention of the printing press, the computer, and the explosive growth of the Web, the amount of readily accessible text has long surpassed the ability of humans to read it."

We have vast amounts of textual information available, but it is in the form of natural language. In order to take advantage of this data, we need to find a way to automatically structure it so that we can actually use it to answer precise queries. For example, if a search engine is requested to answer a question of the form: "Who was the winner of the 1998 Worldcup", we need to know that a *World Cup* is a *yearly event*, that *winners* of a *World Cup* are teams represented by *countries*, and that *France* was the *country* of the *1998 World Cup* winning team.

Typically, Information Extraction (IE) systems are trained through labeled examples which map open text to pre-defined relations. However, this approach does not scale to corpora where the number of target relations is very large and cannot be specified in advance. **Open Information Extraction** (OIE) aims at developing unlexicalized, domain-independant extractors, capable of scaling to Web corpus. In a nutshell, OIE systems should be capable of extracting relational tuples of the form $(Arg1, Pred, Arg2)$, without requiring any relation-specific training data. For example, given the sentence "Mc-Cain fought hard against Obama, but finally lost the election", an Open IE system should extract two tuples, (Mc-Cain, fought against, Obama), and (McCain, lost, the election)[31]. The result of OIE systems are knowledge bases or repositories of structured information such as those presented in section 3.3.

5.1 Three approaches to Open Information Extraction

In this section we will present three state-of-the-art approaches to Open Information Extraction and discuss their contributions and shortcomings.

5.1.1 Reverb: a constraint based approach to identifying relations [32]

Reverb is the name of a program, developed at the University of Washington, that automatically identifies and extracts binary relationships from English sentences [18]. In particular, Reverb extracts triples of the form $(Arg1, Pred, Arg2)$

At the moment Reverb was developed, previous OIE systems developed at the University of Washington, TEXTRUNNER[30] and WOE, proceded in three steps:

1. **Labeling**: Heuristics or distant supervision was used over small corpus to learn a classifier which labeled extractions as "trustworthy" or not.

2. **Identifying relations**: A relation phrase extractor was learned to generate one or more candidate tuples from each sentence, and send each candidate to the classifier, retaining the ones labeled as trustworthy. The extractor uses most probable POS, and noun phrase chunking as a pre-processing step. Relations are found by examining the text between the noun phrases and heuristically eliminating non-essential phrases, such as prepositional phrases that overspecify an entity (e.g.“Scientists from many universities are studying ...” is analyzed as “Scientists are studying...”).

3. **Extracting**: The system takes a sentence, identifies a pair of NP arguments $(Arg1, Arg2)$ and uses the learned Extractor to label the words between the two arguments as part of the relation phrase (or not).

Though these systems performed well, they suffered from incoherent extractions and uninformative extractions.

- **Incoherent extractions**: are cases where the extracted relation phrase has no meaningful interpretation. In an example from [32] the sentence "The guide *contains* dead links and *omits* site" lead to extracting the incoherent relation "contains omits". Incoherent extractions occur because the learned extractor makes a sequence of decisions about whether to include each word in the relation phrase, often resulting in uncomprehensible relations.

- **Uninformative extractions**: these are extractions that omit critical information. In another example from [32], the sentence “Faust made a deal with the devil” was returning the uninformative relation $(Faust, made, a\ deal)$. This type of error is caused by improper handling of relation phrases that are expressed by a combination of a verb with a noun, such as light verb constructions (see section 2.4.1).

To overcome these problems, Fader et. al. [32] introduced a syntactic constraint. They defined a POS tag pattern which the relation phrase is required to match. The pattern limits relation phrases to be either a verb (e.g., created), a verb followed immediately by a preposition (e.g., made in), or a verb followed by nouns, adjectives, or adverbs ending in a preposition (e.g., has atomic weight of). If there are multiple possible matches in a sentence for a single verb, the longest possible match is chosen. If the pattern matches multiple adjacent sequences, we merge them into a single relation phrase (e.g., wants to extend). This refinement enabled the model to handle relation phrases containing multiple verbs, but requires relation phrases to be a contiguous sequence of words in the sentence.

The syntactic constraint implemented in REVERB proved to sometimes match relation phrases which were so specific that they had only a few possible instances. One of the examples presented by the authors of REVERB is the sentence "The Obama administration is offering only modest greenhouse gas reduction targets at the conference" in which the relation extracted was "offering only modest greenhouse gas reduction targets at". To avoid extraction of over specific relations, a lexical constraint was introduced to separate valid relation phrases from over-specified relation phrases.

Lexical constraints were determined statistically and enforced using a large dictionary of relation phrases. The intuition behind this constraint is that relations that appear often in a corpus with different arguments, are more likely to be correct relations than those that appear very unfrequently. They construct the dictionary of valid relations offline by finding all matches of the POS syntactic pattern in a

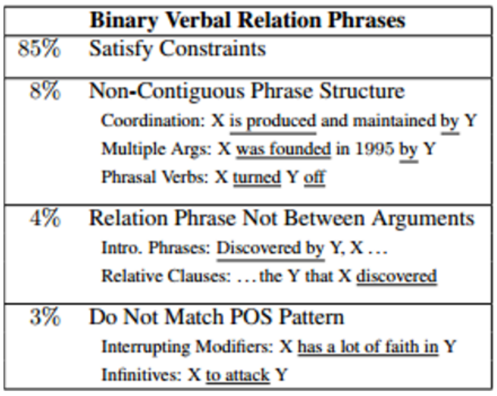
[18]Downloadable at :http://reverb.cs.washington.edu/

corpus of 500 million Web sentences. For each relation phrase, its arguments were heuristically identified and finally, the dictionary was constructed as the set of all relation phrases that take at least k distinct argument pairs. In order to allow for minor variations in relation phrases, the extracted relations were normalized by removing inflection, auxiliary verbs, adjectives, and adverbs. The result was a dictionary of around 1.7 million distinct normalized relation phrases, which is stored in memory at extraction time.

To summarize, REVERB uses POS tagging and NP-chunking as a pre-procesing step. It then uses semantic and lexical constraints to identify binary, verb-based relations composed of contiguous words inside of sentences. Finally it procedes to heuristically identifying the arguments related to the relations, by finding the nearest noun phrases. This type of techniques work well for English since it is not an order-free language (section 2.2.3). However, languages where subject, verb and object can appear in any order, these techniques which rely on sequences of POS tags may not be very efficient.

Fader et.al. acknowledge that their constraints represent an idealized model of relation phrases. To analyze the type of relations that they fail to recognize using the described constraints, they used a test set of 300 sentences in which all verb-based relationships were manually identified. For each relation phrase they checked whether it satisfied the constraints defined by them, and found the results shown in figure 4.

They found that 15% of the relations violated the constraints. Most of the failed cases involved long-range dependencies between words in a sentence. Those types of dependencies are not easily representable using patterns over POS tags and would require a deeper syntactic analysis. Fader et. al. also analyzed the incorrect extractions made by Reverb and noticed that 65% of incorrect extractions were due to incorrect argument identification, rather than relation extraction. One common mistake they found (16% of errors) was when trying to extract a relation phrase expressing an n-ary relationship via a ditransitive verb. For example, in the sentence "I gave him 15 photographs" REVERB would extract $(I, gave, him)$. This is due to the fact that REVERB only models binary relations.



	Binary Verbal Relation Phrases
85%	Satisfy Constraints
8%	Non-Contiguous Phrase Structure
	Coordination: X is produced and maintained by Y
	Multiple Args: X was founded in 1995 by Y
	Phrasal Verbs: X turned Y off
4%	Relation Phrase Not Between Arguments
	Intro. Phrases: Discovered by Y, X …
	Relative Clauses: … the Y that X discovered
3%	Do Not Match POS Pattern
	Interrupting Modifiers: X has a lot of faith in Y
	Infinitives: X to attack Y

Figure 4: Relation phrase compliance with semantic/lexical constraints [32]

5.1.2 DepOE: Using Dependencies to Identify Non-obvious Clauses

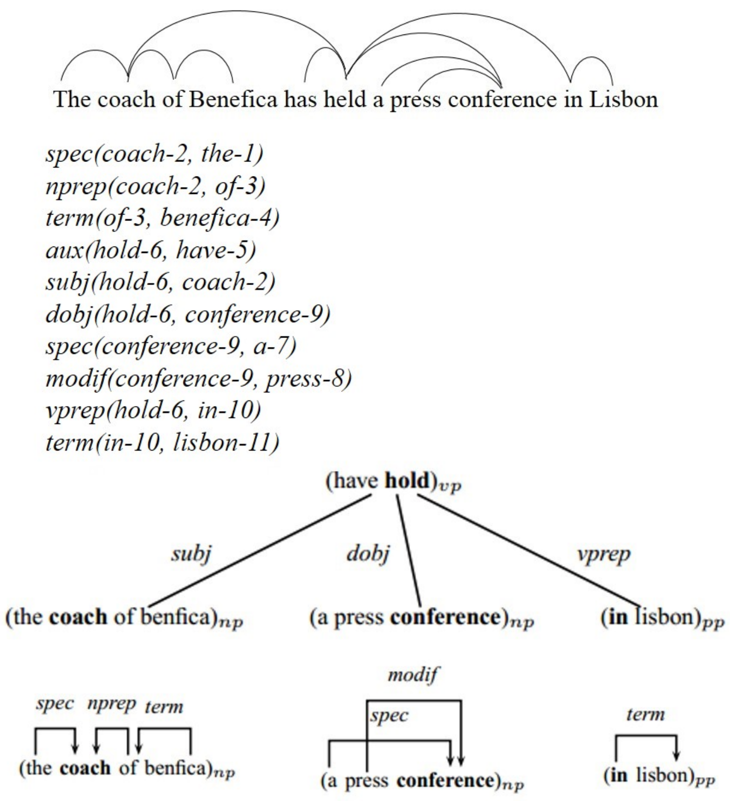
In an attempt to overcome the shortcomings of the approach taken by Fader et.al. in their system REVERB, Gamallo et.al. propose a dependecy-based approach for extracting verb-based relations[36]. Moreover, whereas Reverb is an Open Information Extractor for the English language, Gamallo et.al. claim their approach to be multilingual, provided there is access to a dependency grammar for parsing the desired language.

To further illustrate the advantage of using dependency parsing we will present an example from [36]. In the sentence “The first commercial airline flight was from St. Petersburg to Tampa in 1914” two or three different relational triples could be extracted, namely

$$(the\ first\ commercial\ airline\ flight, was\ from, St.Petersburg)$$
$$(the\ first\ commercial\ airline\ flight, was\ to, Tampa)$$
$$(the\ first\ commercial\ airline\ flight, was\ in, 1914)$$

Performing this multiple extraction is quite challenging if we cannot identify that there is some connection between non-contiguous elements in the sentence.
The OEI method proposed by Gamallo et.al. consists of three steps:

- **Dependency parsing**: This first step is done using DepPattern, a multilingual, robust and fast dependency parser which receives POS tags as input [19].
- **Clause constituents**: For each parsed sentence, the verb clauses are discovered, and then, for each clause, their participants are identified, as well as the participant functions, i.e. subject, direct object, attribute, and prepositional complements. The objective is to transform the dependency path built in the first step into a partial constituency tree, where only the constituents of the clause are selected. This is performed as follows:
  - Given a verb dependency (subj, dobj, vprep, attrib) the dependent lemma of the clause verb is selected, and then all dependent lemmas linked to the target lemma (as a head) are listed using the syntactic dependency path.
  - The verb phrase is built similarly, it contains all dependent lemmas of the verb that are not part of the clause constituents identified before.



```
spec(coach-2, the-1)
nprep(coach-2, of-3)
term(of-3, benefica-4)
aux(hold-6, have-5)
subj(hold-6, coach-2)
dobj(hold-6, conference-9)
spec(conference-9, a-7)
modif(conference-9, press-8)
vprep(hold-6, in-10)
term(in-10, lisbon-11)
```

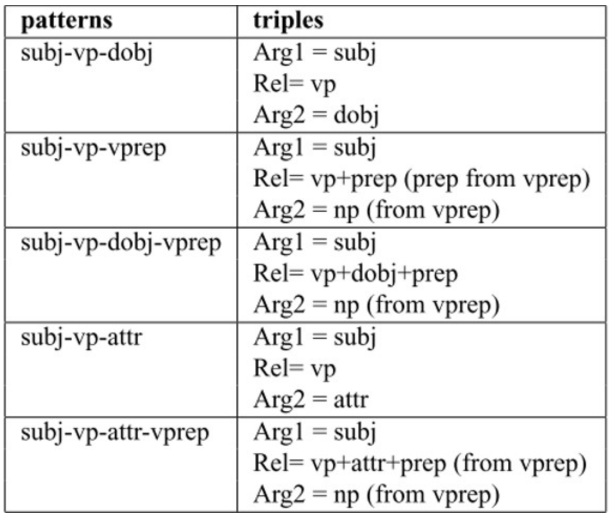
The coach of Benefica has held a press conference in Lisbon
(have **hold**)$_{vp}$
subj
dobj
vprep
(the **coach** of benfica)$_{np}$
(a press **conference**)$_{np}$
(**in** lisbon)$_{pp}$
spec
nprep
term
modif
spec
term

Figure 5: Dependency parsing and clause constituents

The function of a constituent inherits the name of the dependent relation linking the clause verb to the head of the constituent. Figure 5 shows the dependency parse and constituent clauses obtained for the phrase "The coach of Benefica has held a press conference in Lisbon".

- **Extraction**: The final step of the method is to extract the target triples, by applying simple, grammar-based rules on the clause constituents. Figure 6 shows a summary of the main rules applied for extraction. The rules applied by Gamallo et.al. only consider verb-based clause triples and only extract one triple per clause. However, it is possible to write extraction rules to generate several triples from one clause with many arguments, or to extract triples from other patterns of constituents.

[19]Downloadable at: http://gramatica.usc.es/pln/tools/deppattern.html



patterns	triples

subj-vp-dobj	Arg1 = subj
	Rel= vp
	Arg2 = dobj

subj-vp-vprep	Arg1 = subj
	Rel= vp+prep (prep from vprep)
	Arg2 = np (from vprep)

subj-vp-dobj-vprep	Arg1 = subj
	Rel= vp+dobj+prep
	Arg2 = np (from vprep)

subj-vp-attr	Arg1 = subj
	Rel= vp
	Arg2 = attr

subj-vp-attr-vprep	Arg1 = subj
	Rel= vp+attr+prep (from vprep)
	Arg2 = np (from vprep)

Figure 6: Relation extraction rules used by Gamallo et.al.[36]

When compared to Reverb, the method proposed by Gamallo et.al. did not extract as many triples. They attribute this to the fact that that the DepPattern grammars are not complete. In particular, the grammars do not consider all types of coordination and do not deal with significant linguistic clausal phenomena such as interrogative, conditional, causal, or adversative clauses. DepOE showed to have more precise extractions of the two arguments in triples, in particular of $Arg1$, since the parser is able to correctly identify the subject. Nevertheless, as in Reverb, it produces many truncated $Arg2$ arguments. An example of a truncated $Arg2$ is in the extracted triple ($Cities\ and\ towns\ in\ Romania, can\ have, the\ status$) from the sentence “Cities and towns in Romania can have the status either of municipiu or oras”.

The authors of DepOE conclude that the improvement of the system depends on improving the grammars it is based on. Additionally, they found problems concerning the correct identification of arguments due to unefficient Named Entity Recognition. Unfortunately, improving grammars can be very labor-intensive and thus, a costly task. The same applies for the design of rules for the extraction phase.

5.1.3 Mapping Natural Language to Knowledge Bases

This third approach to OIE, takes advantage of existing knowledge bases to identify known relations or word entities in sentences. Bordes et.al.[17] use WordNet (see section 3.3), among other data sources, to map sentences to known WordNet relations and arguments. They generate tuples of the form ($arg1, relation, arg2$) where the arguments correspond to WordNet synsets and the relation corresponds to egdes between synsets in the WordNet graph. In a similar mapping approach, Weston et.al. [76] assume that a previous language processing step will recognize the arguments in sentences, and then map the possible "relation mentions" between the identified arguments, to relations from Freebase.

Bordes et.al. use the efficient, statistics-based Semantic Role Labeling method introduced in section 4.6.1 which allows identifying the role of each set of words in an input sentence. They then procede to construct tuples of the form ($subject, verb, direct\_object$) where each element is a lemma.

Both approaches use an energy-based model which projects words in a low-dimensional vector space and can compare triples based on their similarity in this space. Bordes et.al. map their extracted tuples to the generated tuples of WordNet synsets. Weston et.al. map possible tuples, formed by conjunctions of words appearing in between two arguments, to known Freebase (see section 3.3) relations with the same two arguments.

6 Question Answering

In this section we will cover another interesting research topic related to semantic parsing, namely Question Answering (QA). We define Question Answering as the automatic search for the answer to a question posed in natural language. QA is becoming more and more interesting in our data-driven world, where query engines are now not only expected to propose relevant documents, but also answer specific questions. We can observe this tendency in the Google Search Engine, which now includes QA for general questions. For example, when querying "who was the first person to climb Mount Everest?", Google provides an answer showing Mount Everest's first ascenders, followed by the usual list of relevant URLs, retrieved using IR techniques. However, if this question is further specified to "who was the first person to climb Mount Everest without oxygen?", the search engine is uncapable of mapping it to the correct answer, and only shows the list of related URLs. Similarly, if we change the verb *climb* with the verb *ascend*, even though they are synonyms, the Google QA engine is uncapable of finding the answer.

QA is highly related to OIE since most approaches to QA involve applying NLP to an input and mapping it to some repository of structured information. As we saw in the previous section, these repositories of structured information are often constructed using Information Extraction techniques. Moreover, OIE methods which involve mapping natural language to relations in knowledge bases, like the one presented in section 5.1.3, can also be applied to question answering. If we are capable of mapping entities of an input question to a KB relation tuple, and identify a missing argument from the input, that missing argument could be the subject of the question being asked.

6.1 Main challenges in Question Answering

Two main challenges arise in the problem of QA of open domain questions:

1. The first challenge is related to the very large number of ways that one relation can be refered to in natural language. On one hand it is impossible to have every possible relation represented in a knowledge base, but on the other hand, mapping two relations that are logically the same but are expressed differently can be very challenging. For example in the question: "Who starred in Pulp Fiction?", it is likely that a knowledge base does not contain the relation $stars\_in$ but rather $acts\_in$. Since the end goal of QA systems is to be able to answer any type of questions made in natural language, solving this mapping is crucial.

2. The second challenge has to do with the fact that users may often pose questions that cannot be answered by a single relation in the knowledge base, but rather a composition or an aggregation of a set of relations. For example, in the specific question: "Who was the coach when Brasil won the World Cup?" the knowledge base will most likely not contain the relation $coach\_of\_world\_cup\_winner$ but rather the relations $coach\_of$ and won_world_cup.

6.2 Approaches to Question Answering

In this section we will cover state-of-the-art approaches to question answering, discussing their performance and shortcomings. We will address QA in two main trends, one based on semantic parsing and the other on knowledge base exploration. All the studied state-of-the-art QA methods are constructed using the knowledge base Freebase, presented in section 3.3.

6.2.1 QA by Training of a Semantic Parser

These methods use knowledge from Freebase, combined with large text corpus to train semantic parsers capable of mapping natural language to logical predicates from the knowledge base. There are two tendencies in representing natural language as logic:

- **CCG parsing** CCG parsing approaches tackle the challenge of mapping a sentence to a composition of logical predicates in a knowledge base. Since CCGs represent language in a logical form, the mapping from a CCG to predicates in a Knowledge base is rather straightforward.

  1. The approach taken by Krishnamurthy et.al. [56] builds a parser using a knowledge base KB, a corpus of dependency-parsed sentences S, a CCG lexicon capable of producing logical forms with predicates from KB and a procedure for identifying mentions of entities from KB in the corpus. In the parser, the CCG logical forms are constructed by combining categories, relations and entities from the knowledge base with logical connectives. The semantic parser is trained to learn parameters for the CCG that produce correct semantic parses for sentences in the corpus.

     To construct the CCG lexicon which allows producing logical forms with predicates from KB, Krishnamurthy et.al. use the corpus of dependency-parsed sentences and identify entities in those sentences. Tuples of the form $(e1, e2, s)$ are extracted where $e1$ and $e2$ are entities and s is the sentence where they are found. A mapping is then done from the dependency path between $e1$ and $e2$, to a set of specified dependency parse patterns. Finally these patterns are matched to relations $r(e1, e2)$ known to exist in the knowledge base. In this sence, this approach to semantic parsing is similar to the dependency based approach to OIE presented in section 5.1.2.

The parser is trained to learn the parameters for the CCG using weak supervision. On one hand it is assumed that every relation instance $r(e_1, e_2)$ from KB is expressed by at least one sentence in the corpus. On the other hand it is also assumed that the correct semantic parse of a sentence contains a subset of the syntactic dependencies contained in a dependency parse of the corpus S. These two assumptions allow for training a parser capable of correctly mapping the logical meaning of an utterance, without ever having observed labeled data mapping input to logical form.

Krishnamurthy et.al. were able to train a parser to accurately map the logical meaning of most binary relations and even around 50% of more complex logical forms, involving conjunctions of multiple relations. The main limitation of this approach is that they rely on hand-built dependency parse patterns which require costly, manual engineering.

2. Cai & Yates build the semantic parser FreeParser which takes an approach similar to that of Krishnamurthy et.al. but overcome the need for manual specification of rules that construct CCG lexical entries from dependency parses [20]. They automate the construction of the CCG lexical entries using a self-supervised architecture. The way this task is approached is through a sentence retrieval engine which constructs key-word queries to obtain sentences over a large text corpus, that are likely to express the same relationships as those of the object knowledge base (Freebase). The way these sentences are identified is by the mapping of entities in the sentence as well as in the Freebase relation. These sentences are then mapped to a simple logical form, using only the Freebase relation and the key-word query. From the retrieved sentences, those that are too long or that appear to be labeled incorrectly are filtered out, and the remaining are used to train the semantic parser. By mapping "relation mentions" to relations from Freebase, based on how often they appear with the same entities, they tackle the challenge of recognizing different representations of a same relations (section 6.1). This approach relates to the lexical constraint introduced in Reverb (section 5.1.1).

3. Kwiatkowski et.al. [59] take a different approach than the previous two and train their parser using question-answer pairs rather than weak supervision based on constraints. Their parser over-generates possible logical forms to a question and chooses the one with the highest score as the most correct one. The features for this score calculation are derived from the training set.

   Generating QA answer pairs requires manual labeling which can be costly. In their research, Kwiatkowski et.al. used a set of QA pairs mapping NL questions to Freebase relations which was created and made publicly available by Berant et.al. [11] in their QA studies. It was created using the Google Suggest API to obtain one million questions of which a random sample of one hundred thousand were submitted to Amazon Mechanical Turk. In a collaborative, manual effort, these questions were mapped to Freebase entities.

   Kwiatskowski et.al. try to overcome the challenge of the very large number of ways that a relation can be expressed by natural language, by introducing an ontology mapping stage in the parsing. The parser first constructs a linguistically motivated domain-independent meaning representation using CCG. It then uses a learned ontology matching model to transform this representation for the target domain. As opposed to the previous two approaches, Kwiatkowsi et.al. do not induce a CCG lexicon, the lexicon is open domain, using no symbols from the ontology. The burden of learning word meaning is shifted to the second, ontology matching, stage of parsing.

   The ontology mapping phase not only matches constants (e.g. *stars in* to *acts in*), but it also does structure matching. Structure matching is done by applying rules that collapse or expand literals to obtain new logical forms. An example of collapsing would be mapping $z.Public(z) \bigwedge Library(z) \bigwedge Of(z, NewYork)$ to the more specific form $PublicLibraryOfNewYork$. Performing collapses on the underspecified logical form allows non-contiguous phrases to be represented in a collapsed form. On the other hand, expanding a logical predicate can allow finding relations that can be represented as a conjunction of relations of the ontology (knowledge base). For example Freebase does not contain a relation 'daugther' but this can be represented by expanding to the relations 'female' and 'child'.

- **Dependency-based compositional semantics**

1. Berant et. at. [11] develop SEMPRE[20], a semantic parser for Freebase which is trained using question-answer pairs, like Kwiatkowsi et.al. [59]. They build upon the idea of Cai & Yates [20] to align sentences to relations based on appearance of same entities. In particular, they build a lexicon which maps natural language relations to Freebase relations. Instead of using CCG to map sentences to logical representations, they use their constructed lexicon to detect possible mappings of a sentence to a relation. To improve the quality of these mappings, they introduce a bridging operation which is capable of deriving new relations from found ones. This is important to cover cases where predicates are expressed weakly or implicitly (section 6.1). We will use the examples presented by the authors to illustrate this process. In the question "What government does Chile have?", the predicate is expressed by the light verb *have*. In the question "What actors are in Top Gun?" the predicate is expressed in the highly ambiguous preposition *in*. The bridging relation takes mapped relations and is capable of creating logical expressions that may not appear in the lexicon, or that are defined as a conjunction of relations of the knowledge base. This operation greatly over-generates the possible representations for a given question, and like Krishnamurthy et.al.[56], they train their model using learned features to select the most appropriate representation. To guide the composition of predicates in the bridging stage, they use POS tag features. These features can penalize skipping of important elements such as proper nouns, and can promote the conjunction of elements likely to depend on each other. For example, the phrase "located" is aligned to the predicate ContainedBy. POS features can detect that if "located" precedes a noun phrase ("What is located in Beijing?"), then the noun phrase is the object of the predicate, and if it follows the noun phrase ("Where is Beijing located?"), then it is in subject position.
2. Berant & Liang [12] develop PARASEMPRE, a system built using paraphrasing to generate possible meanings for a given input. Given an input utterance, they first use a simple method to deterministically generate a set of candidate logical forms present in a knowledge base, with a canonical realization in natural language for each. Then, a paraphrase model is used to choose the realization that best paraphrases the input. Paraphrases are generated using an association model and a vector space model and the models are trained using QA pairs. Figure 7 shows a high level comparison between the semantic parsing QA models we have discussed so far. Whereas Krishnamurthy et.al. [56], Cai & Yates [20] and Berant et.al. [11] take a direct approach to mapping utterances to logical predicates, Kwiatkowski et.al. [59] and Berant & Liang [12] rely on an intermediate layer to improve mappings between natural language associations and relations explicitly existing in a knowledge base.

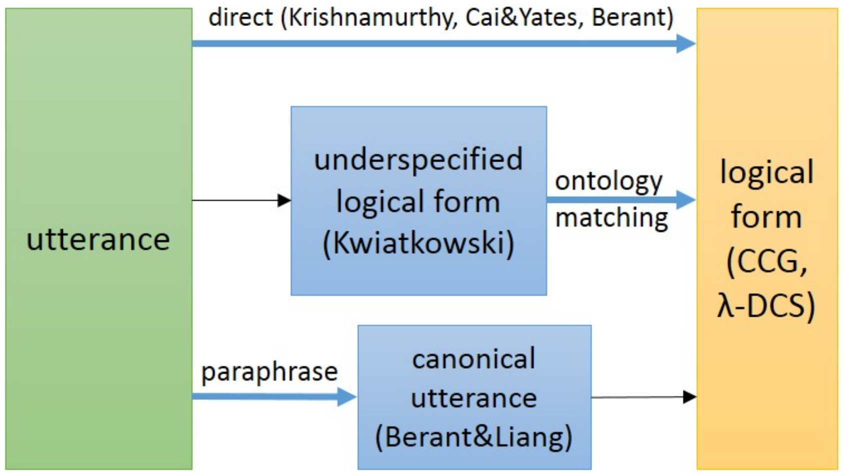

direct (Krishnamurthy, Cai&Yates, Berant)
utterance
underspecified logical form (Kwiatkowski)
ontology matching
logical form (CCG, λ-DCS)
paraphrase
canonical utterance (Berant&Liang)

Figure 7: A comparison between QA semantic parsing approaches[12]

To construct candidate logical forms for a given utterance the strategy is to find an entity from the knowledge base in the utterence and grow the logical form from that entity. This is done following a simple set of templates. Then each candidate logical form is mapped to

[20]Source code can be downloaded at: http://nlp.stanford.edu/software/sempre/

a canonical representation in natural language. The mapping of logical predicate to natural language turned out to be fairly simple using the alignment lexicon released by Berant et.al [11].

The paraphrasing step aims at selecting the canonical representation, generated from the logical forms, which best paraphrases the input utterance. The association model aims at determining whether two natural language phrases contain subphrases that are likely to be paraphrases. For each pair of NL phrases, each pair of subsets of words is a possible association. Associations are primarily detected if they can be found in the PARALEX corpus, a resource containing 18 million pairs of question paraphrases from wikianswers.com, which were tagged as having the same meaning by users. Candidate associations were also generated between words sharing the same lemma, the same POS tag, or that are linked through a derivation link on WordNet (see section 3.3).

The association model relies on having a good set of candidate associations, but mining associations suffers from coverage issues. The vector space model tries to overcome this by assigning a vector representation for each utterance, and learning a scoring function that ranks paraphrase candidates. They start by constructing vector representations of words. Next, they construct a vector for each utterance by averaging the vectors of all content words (nouns, verbs, and adjectives). A paraphrase score is the weighted combination of the components of two phrase vectors. This approach of using similarity between utterances through representations that are based on the word components relates to the OIE approach proposed by Weston et.al. (see section **??**) in which an energy based model compares relations by representing them with their component words in a low dimensional space.

The vector space model can identify correct paraphrases in cases where it is hard to directly associate phrases from one phrase to another. For example, the answer to “Where is made Kia car?”, is given by the canonical utterance “What city is Kia motors a headquarters of?”. The association model does not associate “made” and “headquarters”, but the vector space model is able to determine that these utterances are semantically related[12].

This paraphrasing approach tackles both challenges presented in section 6.1 since the natural language utterances are produced from the logical form, instead of the other way around.

6.2.2 QA by Exploration of a Knowledge Base Graph

This approach differs from the one presented in the previous section, in that it’s goal is not to understand the whole logic of a given question, but simply to find the answer. It views a KB as an interlinked collection of topics. For a given question regarding one or several topics, a view over the knowledge base is considered, concerning only the involved topics. This view is then explored to inspect all elements within a few hops from the topics detected in the input [78]. In a comparison done between both approaches [77], it was shown that that they behaved similarly in terms of accurate question answering.

This approach does not completely ignore parsing. They train a dependency parser using QA pairs, that learns to identify a question word, a question focus a question verb, and a question topic. Figure 8a shows the parsing of the question "What is the name of Justin Bieber’s brother?"

In their approach, Yao & Durme [78] aim at finding the most likely relation a question maps to, i.e. $P(relation|question)$. This probability is calculated using Naive Bayes, under the assumption of conditional independence between words. Although this is a strong assumption which normally does not hold, the approach proved to behave as well as other state-of-the-art involving language processing. Figure 8b shows a view over Freebase entities associated to Justin Bieber, to answer the same question regarding his brothers name.

This approach simplifies language related challenges mentioned in section 6.1 since it only deals with parsing the structure of an input question. They assume that input questions for a search engine will generally have a simple, predictable structure and so will not require deep parsing.

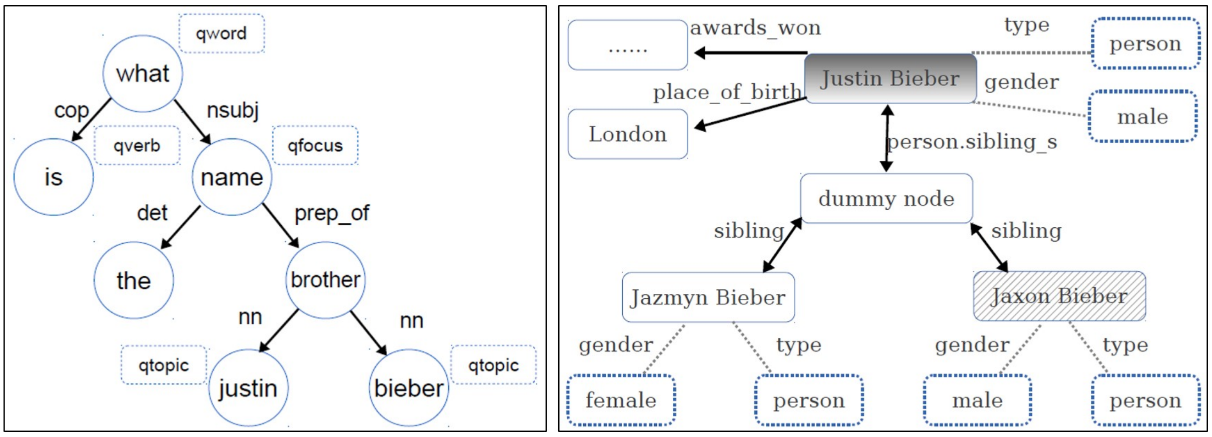


Figure 8: a) Parsing of input sentence[78] b) Freebase graph[78] representation

7 Conclusion and Final Remarks

Most research in IR has been oriented towards bag-of-words approaches. These approaches are efficient for English content because the English language is morphologically poor. However, non-English content is becoming more frequent online, which brings the need for multilingual NLP.

Developing language-specific systems poses new challenges not faced when dealing with English, since different languages have very different linguistic properties. Nonetheless, it is interesting to notice that languages that appear very different, such as Arabic and Russian, can sometimes take advantage of the same types of natural language processing tehniques.

Moreover, IR systems are tending towards QA systems, that is, users do not want relevant documents but answers to questions. The approaches taken to construct QA engines have so far been focused on English, due to the availability of knowledge bases in this language.

The most popular knowledge base currently used in QA (Freebase) is manually constructed. Another approach would be using semantic parsing to automatically extract structured information from the Web; this will make it possible to construct rich knowledge bases for languages other than English.

While at shallow levels of NLP processing a rich morphology might be an additional challenge, at a semantic level of language processing it can actually help understand the meaning of a sentence. For example, the use of cases in Russian can help clearly identify the role of a noun in a sentence, which is an important part of information extraction.

References

[1] Otavio Costa Acosta, Aline Villavicencio, and Viviane P. Moreira. Identification and treatment of multiword expressions applied to information retrieval. In *Proceedings of the Workshop on Multiword Expressions: From Parsing and Generation to the Real World*, MWE '11, pages 101–109, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[2] Imad a. Al-Sughaiyer and Ibrahim a. Al-Kharashi. Arabic Morphological Analysis Techniques: A Comprehensive Survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213, February 2004.

[3] Sameh Alansary, Magdy Nagi, and Noha Adly. Towards Analyzing the International Corpus of Arabic (ICA): Progress of Morphological Stage. *International Conference on Language Resources and Evaluation*, 2008.

[4] AH Aliwy. Arabic Morphosyntactic Raw Text Part of Speech Tagging System. *PhD Dissertation, University of Warsaw*, (January), 2013.

[5] Timothy Baldwin and Su Nam Kim. Multiword Expressions. In Nitin Indurkhya and Fred J Damerau, editors, *Handbook of Natural Language Processing, Second Edition.* CRC Press, Taylor and Francis Group, Boca Raton, FL, 2010.

[6] Yassine Benajiba and Paolo Rosso. ANERsys 2.0: Conquering the NER Task for the Arabic Language by Combining the Maximum Entropy with POS-tag Information. *IICAI*, pages 1814–1823, 2007.

[7] Yassine Benajiba and Paolo Rosso. Arabic Named Entity Recognition Using Conditional Random Fields. *HLT & NLP within the Arabic world: Arabic Language and Local Languages Processing*, 2008.

[8] Yassine Benajiba, Paolo Rosso, and JM Benedíruiz. ANERsys: An Arabic Named Entity Recognition System Based on Maximum Entropy. *CICLing*, pages 143–153, 2007.

[9] Bilel Benali and Fethi Jarray. Genetic Approach for Arabic Part of Speech Tagging. *International Journal on Natural Language Computing*, 2(3):1–12, 2013.

[10] E.M. Bender. *Linguistic Fundamentals for Natural Language Processing.* Morgan & Claypool Publishers, 2013.

[11] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. *Proceedings of EMNLP*, (October):1533–1544, 2013.

[12] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. *Proceedings of ACL*, (Figure 1), 2014.

[13] D.M. Bikel. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* University of Pennsylvania, 2004.

[14] I. Boguslavsky, I. Chardin, and S. Grigorieva. Development of a Dependency Treebank for Russian and its Possible Applications in NLP. Technical report, Russian Academy of Sciences, Moscow, Russia, 2002.

[15] Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Filip Ginter, and Jan Hajic. Joint Morphological and Syntactic Analysis for Richly Inflected Languages. *Association for Computational Linguistics*, 1(2012):415–428, 2013.

[16] A. Bonch-Osmolovskaya, O. Lyashevskaya, and S. Toldova. Learning Computational Linguistics through NLP Evaluation Events : The Experience of Russian Evaluation Initiative. Technical report, Association for Computational Linguistics, Sofia, Bulgaria, 2013.

[17] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengi. Joint learning of words and meaning representations for open-text semantic parsing. *In Proceedings of 15th International Conference on Artificial Intelligence and Statistics*, 22:127–135, 2012.

[18] Thorsten Brants and Google Inc. Natural language processing in information retrieval. In *In Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands*, pages 1–13, 2004.

[19] H C Bunt, P Merlo, and J Nivre. *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing.* Text, Speech and Language Technology. Springer, 2010.

[20] Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*, pages 423–433. The Association for Computer Linguistics, 2013.

[21] Cambridge. Online edition (c) 2009 Cambridge UP. *Cambridge University Press*, (c):19–47, 2009.

[22] D Chiang, M Diab, N Habash, O Rambow, and S Shareef. Parsing Arabic Dialects. *EACL*, 2006.

[23] Vincent Claveau. Unsupervised and semi-supervised morphological analysis for Information Retrieval in the biomedical domain, 2012.

[24] Elizaveta Loginova Clouet, Béatrice Daille, et al. Multilingual compound splitting combining language dependent and independent features. *Computational Linguistics and Intellectual Technologies*, 1(12):455–463, 2013.

[25] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[26] M Diab. Second generation AMIRA tools for Arabic processing: Fast and robust tokenization, POS tagging, and base phrase chunking. pages 285–288, 2009.

[27] B.V. Dobrov and I. Kuralenok. Russian Information Retrieval Evaluation Seminar. Technical report, Moscow State University, St.Petersburg University, 2004.

[28] Ljiljana Dolamic. *Influence of Language Morphological Complexity on Information Retrieval.* PhD thesis, Universite de Neuchatel, 2010.

[29] YOM Elhadj. Statistical Part-of-Speech Tagger for Traditional Arabic Texts. *Journal of Computer Science*, 5(11):794–800, 2009.

[30] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008.

[31] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: The second generation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, pages 3–10. AAAI Press, 2011.

[32] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[33] A Farghaly and K Shaalan. Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing, Vol. 8, No. 4, Article 14*, 8(4):1–22, 2009.

[34] Anna Feldman and Jirka Hana. A Resource-Light Approach to Morpho-Syntactic Tagging. *Computational Linguistics*, 37(1):253–254, 2010.

[35] S Gahbiche-Braham and H Bonneau-Maynard. Joint Segmentation and POS Tagging for Arabic Using a CRF-based Classifier. *LREC*, pages 2107–2113, 2012.

[36] Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, ROBUS-UNSUP '12, pages 10–18, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[37] Rinat Gareev, Maksim Tkachenko, Valery Solovyev, Andrey Simanovsky, and Vladimir Ivanov. Introducing baselines for russian named entity recognition. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7816 of *Lecture Notes in Computer Science*, pages 329–342. Springer Berlin Heidelberg, 2013.

[38] Anastasia Gareyshina, Maxim Ionov, Olga Lyashevskaya, and Dmitry Privoznov. RU-EVAL-2012 : Evaluating dependency parsers for Russian. In *Proceedings of COLING 2012:Posters*, volume 3, pages 349–360, Mumbai, 2012.

[39] A. Gelbukh and G. Sidorov. Morphological Analysis of Inflective Languages through Generation. *Procesamiento de Lenguaje Natural*, 2(1983), 2002.

[40] A. Gelbukh and G. Sidorov. Approach to Construction of Automatic Morphological Analysis Systems for Inflective Languages with Little Effort. Technical Report Cic, National Polytechnic Institute, Mexico, 2003.

[41] Spence Green and CD Manning. Better Arabic parsing: Baselines, evaluations, and analysis. *COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics*, pages 394–402, 2010.

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

[43] N Habash, A Soudi, and T Buckwalter. On Arabic Transliteration. *Arabic Computational Morphology*, 2007.

[44] Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. *Proceedings of the 43rd Annual Meeting of the ACL*, (June):573–580, 2005.

[45] Nizar Habash and Owen Rambow. MAGEAD: Morphological Analysis and Generation for Arabic and its Dialects. In *International Conference on Computational Linguistics*, pages 681–688, 2006.

[46] Nizar Habash, Owen Rambow, and Ryan Roth. MADA + TOKAN : A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. *2nd International Conference on Arabic Language Resources and Tools*, pages 102–109, 2009.

[47] Meryeme Hadni, Said Alaoui Ouatik, Abdelmonaime Lachkar, and Mohammed Meknassi. Hybrid Part-Of-Speech Tagger for Non-Vocalized Arabic Text. *International Journal on Natural Language Computing*, 2(6):1–15, December 2013.

[48] J Hajic, O Smrz, and P Zemánek. Prague Arabic Dependency Treebank: Development in Data and Tools. *International Conference on Arabic Language Resources and Tools*, 2004.

[49] J. Hana, A. Feldman, and C. Brew. A Resource- Approach to Russian Morphology: Tagging Russian Using Czech Resources. *EMNLP*, 2004.

[50] Sanda Harabagiu, Cosmin Adrian Bejan, and Paul Morarescu. Shallow semantics for relation extraction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1061–1066, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[51] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, January 2013.

[52] L Iomdin, V Petrovchenko, V Sizov, and L Tsinman. ETAP parser : State of the Art, 2011.

[53] Jing Jiang and Chengxiang Zhai. An empirical study of tokenization strategies for biomedical information retrieval. *Inf. Retr.*, 10(4-5):341–363, October 2007.

[54] ED Kallestinova. *Aspects of Word Order in Russian.* Phd dissertation, University of Iowa, 2007.

[55] Dan Klein and Christopher D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

[56] Jayant Krishnamurthy and Tom M. Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 754–765, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[57] Seth Kulick. Simultaneous tokenization and part-of-speech tagging for Arabic without a morphological analyzer. *Proceedings of the ACL 2010 Conference Short Papers*, (July):342–347, 2010.

[58] Seth Kulick, Ryan Gabbard, and M Marcus. Parsing the Arabic treebank: Analysis and improvements. *Proceedings of the TLT*, pages 31–42, 2006.

[59] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling Semantic Parsers with On-the-fly Ontology Matching. *Emnlp*, (October):1545–1556, 2013.

[60] Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Inducing Probabilistic CCG Grammars from Logical Form with Higher-order Unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1223–1233, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[61] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, Cambridge, UK, 2008.

[62] Yuval Marton, Nizar Habash, and Owen Rambow. Improving Arabic dependency parsing with lexical and inflectional morphological features. *Proceedings of the NAACL HLT 2010. First Workshop on Statistical Parsing of Morphologically-Rich Languages*, (June):13–21, 2010.

[63] M. C. McCord, J. W. Murdock, and B.K. Boguraev. Deep parsing in Watson. *IBM Journal of Research and Development*, 56(3.4):3:1–3:15, May 2012.

[64] Emad Mohamed. *Orthographic Enrichement for Arabic Grammatical Analysis.* PhD thesis, Indiana University, 2010.

[65] Emad Mohamed and S Kübler. Arabic Part of Speech Tagging. *LREC*, 2010.

[66] Tahira Naseem, Benjamin Snyder, Jacob Eisenstein, and Regina Barzilay. Multilingual part-of-speech tagging: Two unsupervised approaches. *Journal of Artificial Intelligence Research*, pages 1076–9757.

[67] Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, 2006.

[68] M Oudah and KF Shaalan. A Pipeline Arabic Named Entity Recognition using a Hybrid Approach. *COLING*, 2(December 2012):2159–2176, 2012.

[69] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[70] A Rozovskaya, R Sproat, and E Benmamoun. Challenges in Processing Colloquial Arabic. *Challenges to Arabic NLP*, pages 4–14.

[71] Khaled Shaalan. Rule-based approach in Arabic natural language processing. *The International Journal on Information and Communication Technologies*, 3(3), 2010.

[72] Khaled Shaalan and Hafsa Raza. NERA: Named entity recognition for Arabic. *Journal of the American Society for Information Science and Technology*, 60(8):1652–1663, 2009.

[73] S. Sharoff, M. Kopotev, and T. Erjavec. Designing and Evaluating a Russian Tagset. *LREC*, 2008.

[74] U. S. Tiwary and Tanveer Siddiqui. *Natural Language Processing and Information Retrieval.* Oxford University Press, Inc., New York, NY, USA, 2008.

[75] University of Maryland. Part-of-Speech Tagging.

[76] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. *CoRR*, abs/1307.7973, 2013.

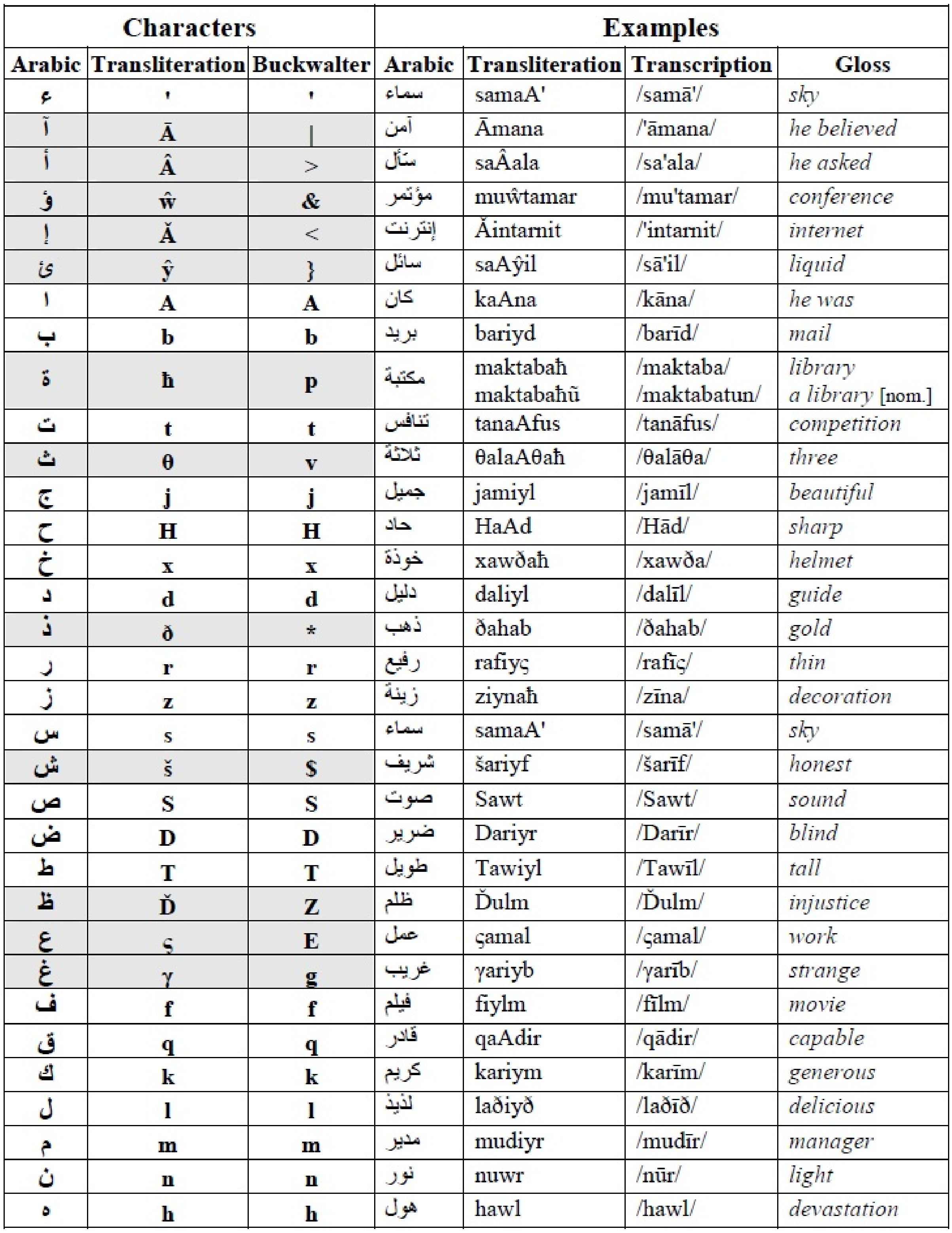
[77] Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. Freebase QA: Information Extraction or Semantic Parsing? *allenai.org*, 2014.

[78] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*, 2014.

[79] A. Zouaq. An overview of shallow and deep natural language processing for ontology learning. In W. Wong, W. Liu, and M. Bennamoun, editors, *Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances*. IGI Global, 2011.

A Habash-Soudi-Buckwalter Arabic Transliteration Scheme

This table as well as further explanation of the transliteration scheme and Arabic script properties can be found in [43].



Characters			Examples			
Arabic	**Transliteration**	**Buckwalter**	**Arabic**	**Transliteration**	**Transcription**	**Gloss**
ء	'	'	سماء	samaA'	/samā'/	*sky*
آ	Ā	|	آمن	Āmana	/'āmana/	*he believed*
أ	Â	>	سَأل	saÂala	/sa'ala/	*he asked*
ؤ	ŵ	&	مؤتمر	muŵtamar	/mu'tamar/	*conference*
إ	Ǎ	<	إنترنت	Ǎintarnit	/'intarnit/	*internet*
ئ	ŷ	}	سائل	saAŷil	/sā'il/	*liquid*
ا	A	A	كان	kaAna	/kāna/	*he was*
ب	b	b	بريد	bariyd	/barīd/	*mail*
ة	ħ	p	مكتبة	maktabaħ maktabaħũ	/maktaba/ /maktabatun/	*library* *a library* [nom.]
ت	t	t	تنافس	tanaAfus	/tanāfus/	*competition*
ث	θ	v	ثلاثة	θalaAθaħ	/θalāθa/	*three*
ج	j	j	جميل	jamiyl	/jamīl/	*beautiful*
ح	H	H	حاد	HaAd	/Hād/	*sharp*
خ	x	x	خوذة	xawðaħ	/xawða/	*helmet*
د	d	d	دليل	daliyl	/dalīl/	*guide*
ذ	ð	*	ذهب	ðahab	/ðahab/	*gold*
ر	r	r	رفيع	rafiyς	/rafīς/	*thin*
ز	z	z	زينة	ziynaħ	/zīna/	*decoration*
س	s	s	سماء	samaA'	/samā'/	*sky*
ش	š	$	شريف	šariyf	/šarīf/	*honest*
ص	S	S	صوت	Sawt	/Sawt/	*sound*
ض	D	D	ضرير	Dariyr	/Darīr/	*blind*
ط	T	T	طويل	Tawiyl	/Tawīl/	*tall*
ظ	Ď	Z	ظلم	Ďulm	/Ďulm/	*injustice*
ع	ς	E	عمل	ςamal	/ςamal/	*work*
غ	γ	g	غريب	γariyb	/γarīb/	*strange*
ف	f	f	فيلم	fiylm	/fīlm/	*movie*
ق	q	q	قادر	qaAdir	/qādir/	*capable*
ك	k	k	كريم	kariym	/karīm/	*generous*
ل	l	l	لذيذ	laðiyð	/laðīð/	*delicious*
م	m	m	مدير	mudiyr	/mudīr/	*manager*
ن	n	n	نور	nuwr	/nūr/	*light*
ه	h	h	هول	hawl	/hawl/	*devastation*

و	w	w	وصل	waSl	/waSl/	*receipt*
ى	ý	Y	على	çalaý	/çala/	*on*
ي	y	y	تين	tiyn	/tīn/	*figs*
َ	a	a	دَهَنَ	dahana	/dahana/	*he painted*
ُ	u	u	دُهِنَ	duhina	/duhina/	*it was painted*
ِ	i	i	دُهِنَ	duhina	/duhina/	*it was painted*
ً	ã	F	كتاباً	kitaAbAã	/kitāban/	*a book* [nom.]
ٌ	ũ	N	كتابٌ	kitaAbũ	/kitābun/	*a book* [acc.]
ٍ	ĩ	K	كتابٍ	kitaAbĩ	/kitābin/	*a book* [gen.]
ّ †	~	~	كَسَّرَ	kas~ara	/kassara/	*he smashed*
ْ ‡	.	o	مَسْجِد	mas.jid *or* masjid	/masjid/	*mosque*
ـ §	–	–	مَسْــجِد	mas.____jid	/masjid/	*mosque*

† Shadda (شدة šad~aħ) is a symbol marking consonant doubling.
‡ Sukun (سكون sukuwn) is a symbol marking lack of vowel. It can be used for contrastive purposes in the transliteration. However, it is not required in this book for the purpose of improving readability.
§ Tatweel (تطويل taTwiyl) or Kashida (كشيدة kašiydaħ) is an orthographic elongation symbol with no phonetic value.

Figure 9: Transliteration Scheme