Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

Ahmed Draia University

Faculty of Science and Technology

Department of Mathematics and Computer Science

A Thesis Presented to Fulfil the Master's Degree in Computer Science

Option: Intelligent Systems.

Title:

Preprocessing for Arabic Neural Machine Translation

Prepared by:

Abbou Mohamed Elcherif & Guerrout Omar

<u>**Jury Members:**</u>

President: Mr.Cheragui Mohammed Amine

Examiner: Mr.Benatillah Djelloul

Supervisor: Mr.Mediani Mohammed

2019/2020

Abstract:

Data driven methods have become the way to go when it comes to NLP, especially in the case of machine translation, with the rise of neural machine translation, a new method that incorporates the use of recurrent neural networks. These NMT systems are very sensitive to the quality the training data; a large rich well-structured data set can make huge difference in the performance of the translation.

This work aims to study the impact of preprocessing on the performance of Arabic-English Neural Machine Translation Systems. We limit our research on the effect of text segmentation.

We introduce a new alignment-based segmentation technique that tries to address the issue of translating from and into the Arabic language. We perform multiple translation experiments in which we use different Segmentation methods. Our results shows that preprocessing the training data improves the performance of the NMT model. We also report that our technique improves the translation performance and even outperform BPE in some cases, however did not match BPE best performing configuration.

Key words: NLP, machine translation, NMT, segmentation, BPE, alignment, data preprocessing, Arabic translation.

ملخص:

أصبحت الأساليب المعتمدة على البيانات هي الطريق الذي يجب اتباعه عندما يتعلق الأمر بمعالجة اللغة الطبيعية ، خاصة في حالة الترجمة الآلية ، مع ظهور الترجمة الآلية العصبية ، وهي طريقة جديدة تتضمن استخدام الشبكات العصبية المتكررة. تعتبر أنظمة NMT حساسة للغاية لجودة بيانات التدريب ؛ يمكن لمجموعة كبيرة غنية من البيانات جيدة التنظيم أن تحدث فرقًا كبيرًا في أداء الترجمة.

يهدف هذا العمل إلى دراسة تأثير المعالجة المسبقة على أداء أنظمة الترجمة الآلية العصبية من العربية إلى الإنجليزية. نحن نحد من بحثنا حول تأثير تجزئة النص.

قدمنا تقنية تجزئة جديدة قائمة على المحاذاة تحاول معالجة مسألة الترجمة من وإلى اللغة العربية. أجرينا تجارب ترجمة متعددة استخدمنا فيها طرق تجزئة مختلفة. تظهر نتائجنا أن المعالجة المسبقة للبيانات تحسن أداء نموذج NMT بشكل ملحوظ. أبلغنا أيضًا أن أسلوبنا يعمل على تحسين أداء الترجمة بل ويتفوق على BPE في بعض الحالات ، ومع ذلك لم يتفوق على أفضل تكوينBPE .

الكلمات الأساسية: المعالجة الآلية للغة ، الترجمة الآلية ، NMT ، التجزئة ، BPE ، المحاذاة ، المعالجة المسبقة للبيانات ، الترجمة العربية

Résumé:

Les méthodes basées sur les données sont devenues la voie à suivre en matière de traitement du langage naturel, en particulier dans le cas de la traduction automatique, avec l'essor de la traduction automatique neuronale, une nouvelle méthode qui intègre l'utilisation de réseaux de neurones récurrents. Ces systèmes NMT sont très sensibles à la qualité des données de formation; un vaste ensemble de données riches et bien structurées peut faire une énorme différence dans les performances de la traduction.

Ce travail vise à étudier l'impact du prétraitement sur les performances des systèmes de traduction automatique neuronale arabe-anglais; nous limitons nos recherches sur l'effet de la segmentation du texte.

Nous avons introduit une nouvelle technique de segmentation basée sur l'alignement qui tente de résoudre le problème de la traduction depuis et vers la langue arabe. Nous avons effectué plusieurs experimentations de traduction dans lesquelles nous avons utilisé différentes méthodes de segmentation. Nos résultats montrent que le prétraitement des données améliore considérablement les performances du modèle NMT. Nous signalons également que notre technique améliore les performances de traduction et même surpasse BPE dans certains cas, mais ne correspond pas à la configuration la plus performante de BPE.

Mots clés: traitement du langage naturel, traduction automatique, NMT, segmentation, BPE, alignement, prétraitement des données, traduction Arabe.

Dedications:

The sake of Allah, my Creator and my Master,

My great teacher and messenger, Mohammed (May Allah bless and grant him), who taught us the purpose of life,

Our parents who love me unconditionally and who have taught me to work hard for the things that we aspire to achieve.

My beloved brothers and sisters whose presence gives us meaning in life,

My friends and loved one because they bring joy to our lives,

To all people who are thriving every day to leave the world better than they found,

I dedicate this work to you.

Mohammed Elcherif

Dedications:

In the name of Allah, the Most Gracious and the Most Merciful,

I wish to dedicate this thesis to my father (May allah have mercy on him bless and grant him jannat-ul-firdous),whom i lost just before finishing it, for always believing in me , for supporting me through the good and the bad and for the priceless lessons and memories that i will hold on to for the rest of my life.

My mother for always being there for me and all , my brothers and sister for bringing joy into my life.

All my family and friends for all the support and wishes you have given to me.

Omar.

Acknowledgements

Firstly, we would like to express our sincere gratitude to Mr. MEDIANI Mohammed for the continuous support of our research efforts, for his patience, motivation, and immense knowledge. His guidance helped us throughout all stages of research and writing.

Secondly, we thank the jury members for accepting to examine this work. We would like also to thank all our teachers at the department of mathematics and computer sciences and our classmates for the amazing journey throughout the last five years. We are grateful for knowing and learning from you.

Finally yet importantly, we would like to thank our families, our parents, our siblings, and our close friends for supporting us spiritually throughout writing this research work and in life in general.

Abbou Mohammed Elcherif and Guerrout Omar

Table of Contents:

List of tables:

List of figures:

List of acronyms:

NLP: Natural language Processing

MT: Machine Translation

NMT: Neural machine translation

LSTM: Long short-term memory

MRL: Morphology Rich Language

RNNs: Recurrent Neural Networks

DNNs: Deep Neural Networks

OOV: Out of Vocabulary

BPE: Byte Pair Encoding

ABS: Alignment-based segmentation

Chapter 1 : Introduction

1.1. Introduction

One of the most important aspects of human beings is their ability to collaborate to achieve great feats. Such collaboration is only possible because of our ability to share knowledge through communication. There are many ways of communication with each other whether it is through speech, text, signals, signs etc.

Although it might seem that communicating is a simple task for us humans, the ability to understand speech or the ability to read a written text off a screen and comprehend the meaning behind it is far from simple. We are yet to understand the full mechanisms behind this process. So how do we achieve human-like processing of languages?

Natural Language Processing (NLP) has been the answer to this problem with the goal of achieving human-like processing of languages.

Moreover, since most of the knowledge and data nowadays is online, having it accessible to all is vital, and one major issue is the language barrier due to the existence of so many languages across the globe. For a long time people used human translators (those who master two or more languages) but this solution is expensive and time consuming so another faster and cheaper solution was necessary, this necessity lead scientists to develop to Machine Translation (MT).

While translation is easy to achieve with pairs of language that are closely related and have similar morphology and rules like English-French and Spanish-Italian ,Languages that differ a lot from each other in morphology and rules of composition like for example the pairs Arabic-English or Chinese-Russian adds another layer of complication to the translation process [1] .

Arabic for example is the fourth most used language [2] on the internet and it is spoken by more than 400 million person globally, and used by 1.4 billion Muslim to recite Quran and perform their prayers. It has a huge vocabulary, rich morphology and a complicated sentence composition, making it a difficult language to translate from and to. This calls for special pre-processing measurements to be performed on the Arabic text for it to be well segmented and disambiguated for it to be easily digested by the translation algorithms resulting in an overall better translation performance.

1.2. Natural Language Processing (NLP)

Natural language processing is a sub-field of Artificial Intelligence, which incorporates the use of machine learning and computational linguistics; it focuses on achieving human-like processing of natural language. [3] it is composed of multiple systems that work together to create an end-to-end interaction between human and machine in human natural language.

In the early days of AI, NLP was believed to be interchangeable to (NLU) Natural Language Understanding; however, NLU is actually a subset of NLP responsible mainly of drawing inference out of text, it is in fact one the most challenging aspects of AI that to this day NLP systems have not achieved yet (check figure (1.1)) .

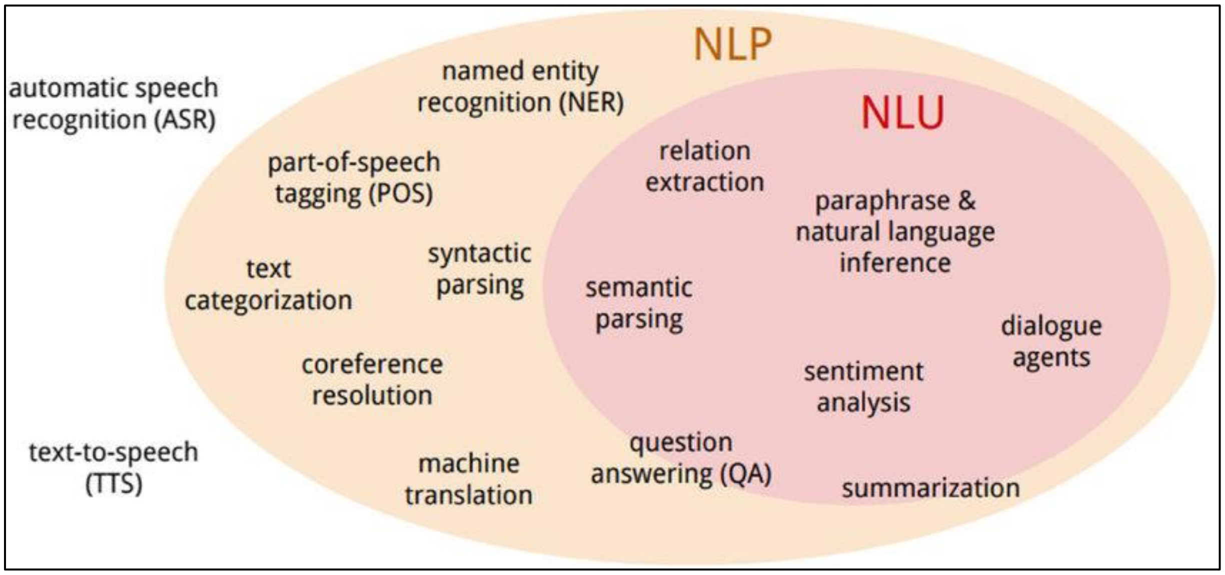


Figure 1.1 Terminology of NLP versus NLU

Thought we have not reached a true natural language understanding, NLP have made a great progress in many applications such as machine translation, which achieved translations comparable to human.

1.2.1. NLP process:

Some of NLP most important phases as shown in figure (1.2):

A. Lexical analysis:

This step involves splitting the input sentences into small entities called Tokens they represent the core information that is used throughout the processing.

B. Syntactic analysis:

In this phase, the data is parsed to check and validate the sentence composition for example a sentence like; “the fridge is inside the cheese” is to be rejected by the parser.

C. Semantic analysis:

In this third phase, the goal is to decipher the meaning behind the text, in other words the dictionary meaning of each token. For example, the sentence “the fire is cold” is rejected in this phase

D. Output transformation:

In this last step, the output from the semantic analyzer is transformed into the desired results.

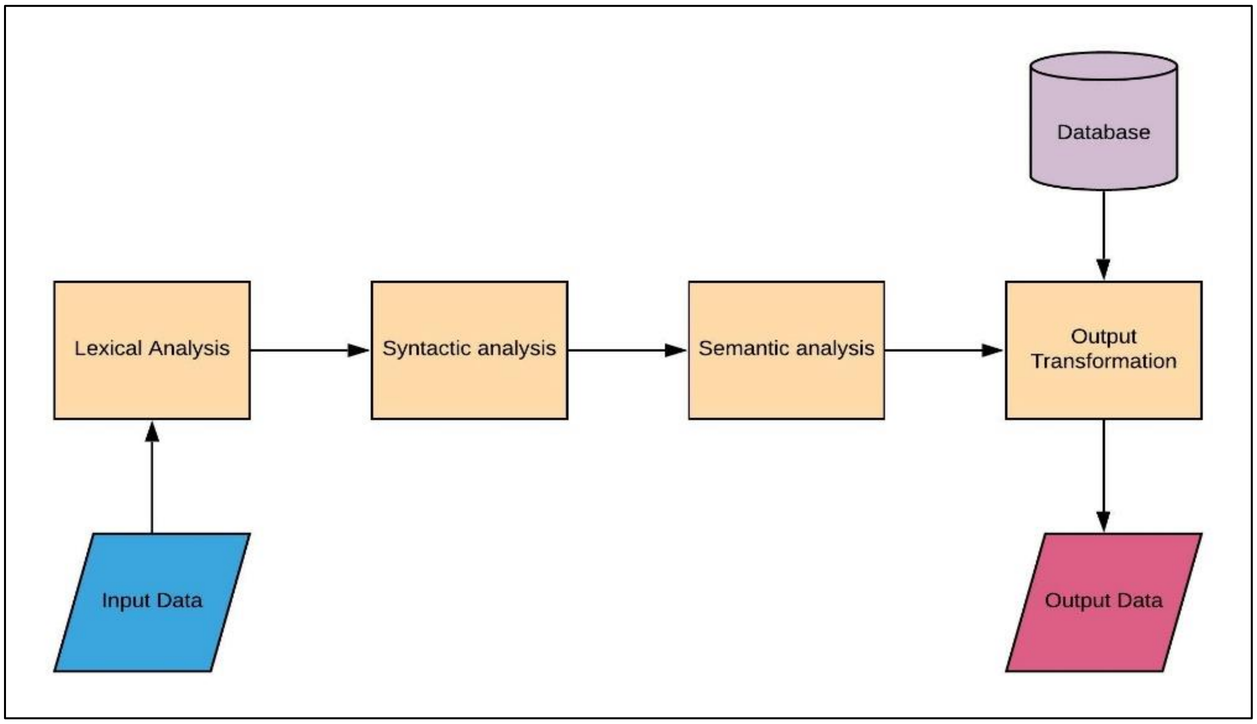


Figure 1.2: Steps of NLP

1.3. Machine Translation:

Machine translation is the automatic process of transforming a source text from one language to another.

Machine translation is relatively an old task, early attempts started in 1950s but the real progress happened in 1970s where there have many approaches to achieve automatic translation:

- Rule-based Machine Translation (RBMT): 1970s-1990s
- Statistical Machine Translation (SMT): 1990s-2010s
- Neural Machine Translation (NMT): 2014- to present

1.3.1. Rule-based Machine Translation:

Rule-based Machine Translation [26] translates using Rules written by humans from linguistic knowledge from the both the source language and the target language, this approach requires large set of rules (syntactic, semantic and morphological) and bilingual dictionary and generates the target text following the steps shown in figure (1.3).

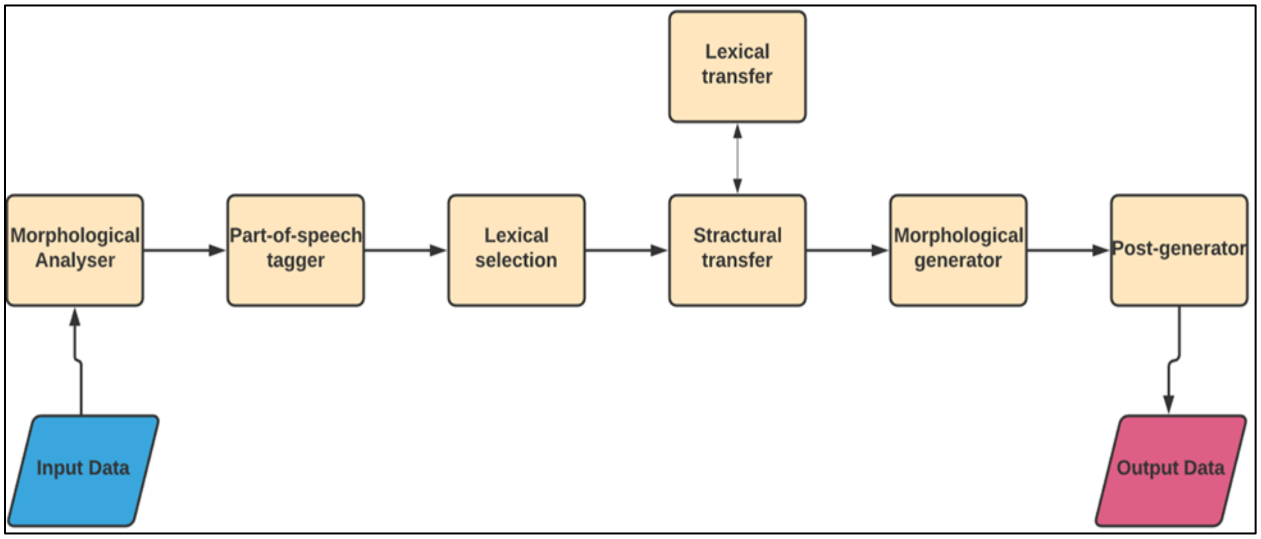


Figure 1.3: steps of RBMT

While this approach provides a simple and predictable translation and requires only a few computational resources its high development and maintaining or extending costs made it fade and replaced with other approaches like (SMT) AND (NMT).

1.3.2. Statistical Machine Translation (SMT):

Statistical Machine Translation [25] uses statistical analysis and predictive algorithms of bilingual text corpora (existing human translations of the source and target language) to produce a statistical model of translation.

This model is then used on an untranslated text to generate the most probable and reasonable translation this method can be applied either word-based, phrase-based, syntax-based and hierarchical phrase-based.

State of the art SMTs are usually phrase based so given a source text. $s_1^J = s_1 \ldots. s_2 \ldots. s_J$ the objective of SMT is to translate it to target language $t_1^i = t_1 \ldots. t_2 \ldots. t_i$ by choosing the target text with the highest probability (see the formula bellow).

$$T_1^i = \mathrm{argmax}\, P(t_1^i | s_1^J)$$

Where “s” and “t” are phrases from the source and target languages and “i” and “j” are the number of words in source and target language respectively.

1.3.4. Neural machine translation

Neural machine translation (NMT) in newly rising approach to machine translation, it was first introduced in 2014[4], then it caught the attention of researchers and became a hot topic in the field of machine translation due to its promising potential.

Neural machine translation uses large recurrent neural networks to perform translation its strength lies in its ability to learn directly from the data, a true end-to-end method [4], and its architecture is composed of two recurrent neural networks (RNNs) an Encoder and a Decoder, that work together to produce a translation from a text input check (figure 1.4).

However, NMT systems in their early stages actually performed worse than the existing phrase based systems due to its slower training and inference speed, the inability to deal with rare words, and the problem of long sentences. Nevertheless, recent models have improved upon NMT systems making on par if not outperforming all previous techniques.

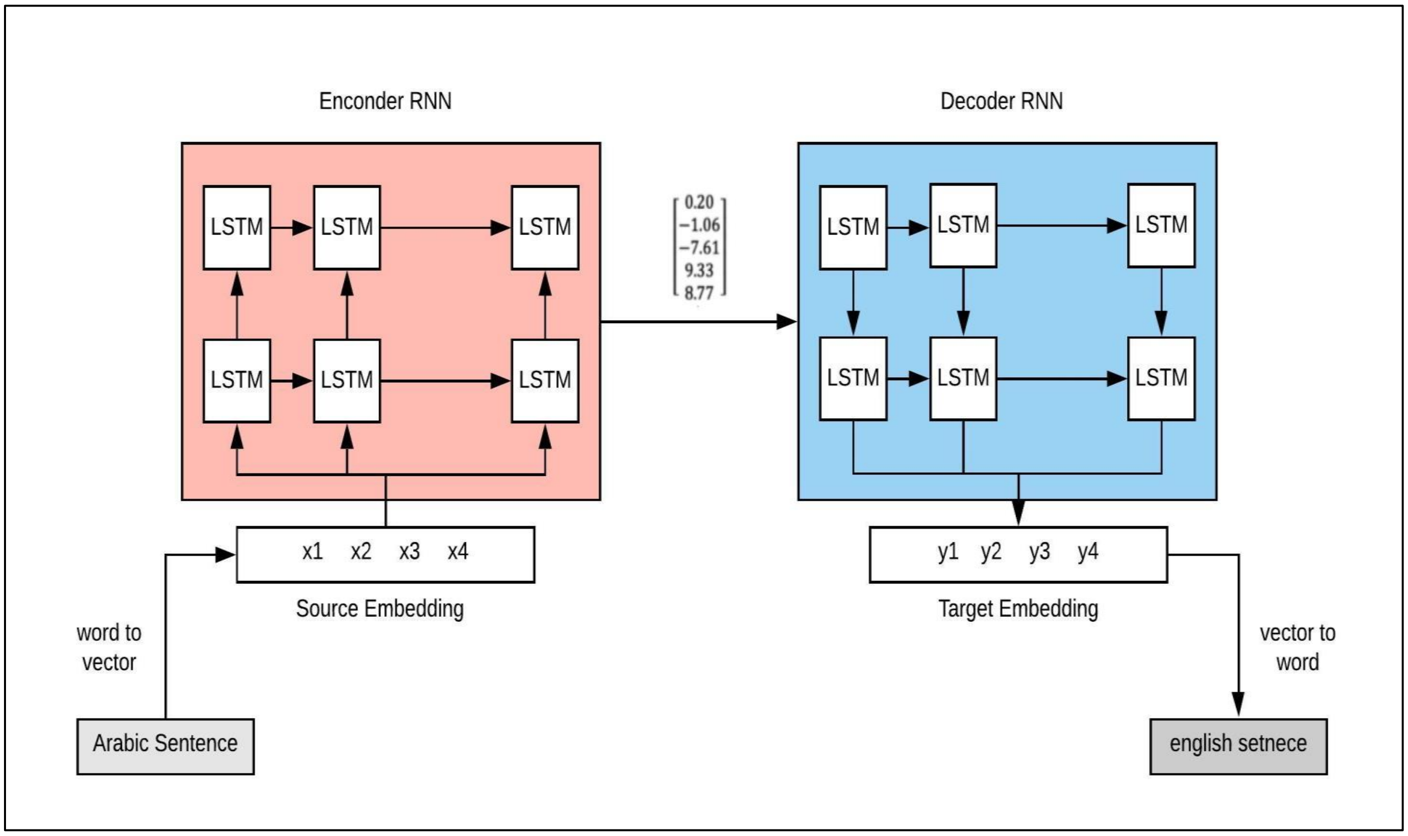


Figure 1.4: Diagram of the NMT system architecture

NMT process:

NMT process goes through many steps in order to achieve the desired translation check figure (1.5).

The first step in neural machine translation is training the model, which requires a large amount of data and computational power.

NMT systems usually use a bilingual corpus, which is a large collection of sentences in one language and their corresponding translation, these sentences must be aligned correctly each sentence by its translation.

Before inputting the data into the model it must be preprocessed, text preprocessing is an important step in machine translation it helps machine learning algorithms digest the inputted data more efficiently resulting in better model performance.

Generally, there are three main components to text preprocessing:

- Tokenization
- Normalization
- Noise removal

Firstly, tokenization also referred to as segmentation is the process of splitting a sequence of characters into small units called tokens; a 'token' is the smallest unit when describing syntax of a language example:

سأخبركم س+أخبر+كم

Text normalization in the process of converting the words into a uniform format, which will facilitates operations on them, an example to that is converting all letters to lower case letters, removing punctuation, stemming, and lemmatization.

Lastly, noise removal is the process of removing non-meaningful characters and digits that can interfere with analyzing the text such as (@ ,<>,§,#).

After preprocessing the data it is then fed to the neural network to begin its training, at the start we define multiple parameters concerning the size of neural network, the size of embedding, the optimization algorithms to use, the duration of training and much more, then the models is left learning and improving by the time.

When the training is done, the model is tested to evaluate its performance.

NMT model usually operates on closed vocabulary corpus meaning it assumes a fixed vocabulary size, since it embeds each word of the vocabulary to a sequence on bits (word embedding), However there are segmentation techniques that manipulate vocab size to avoid this issue.

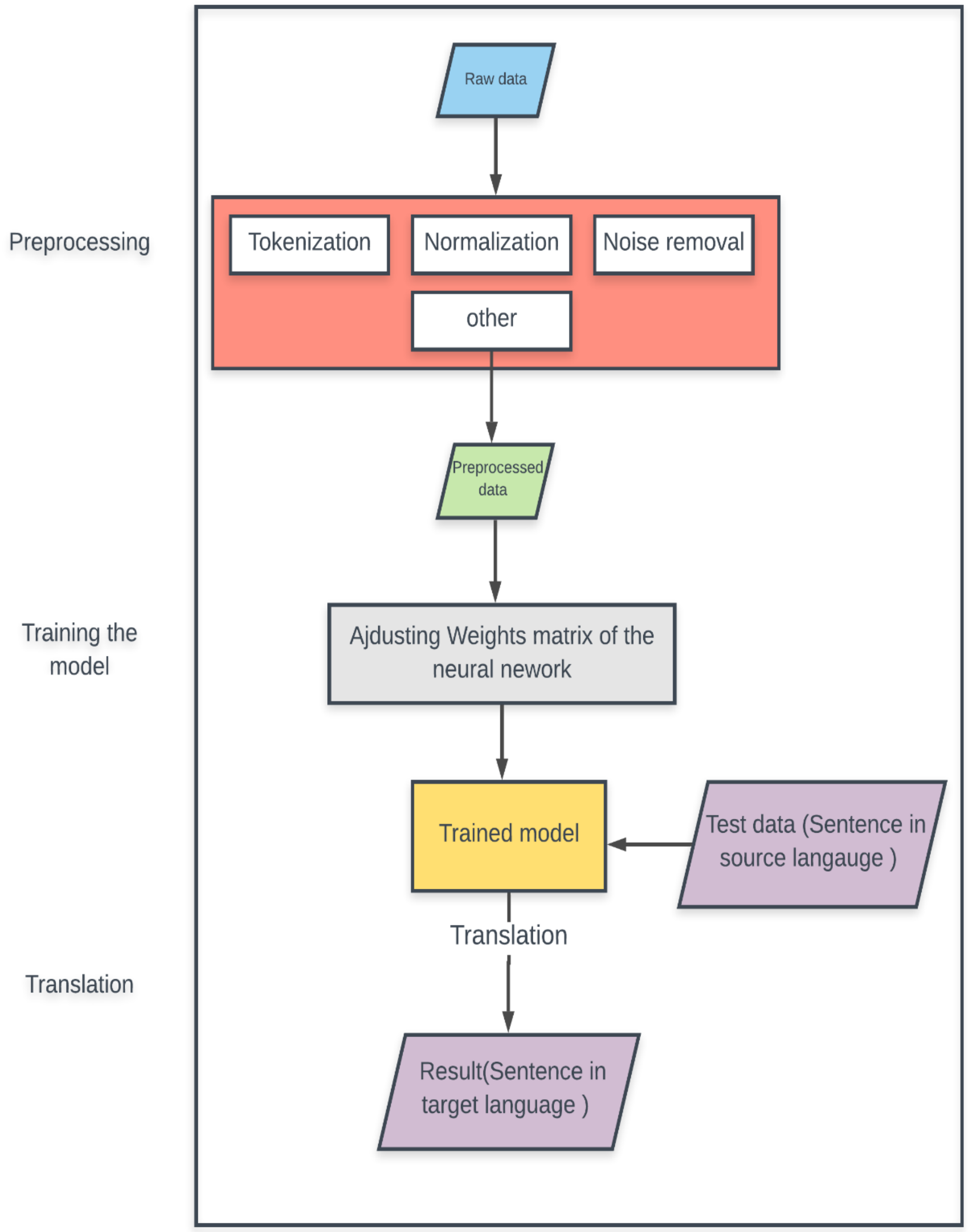


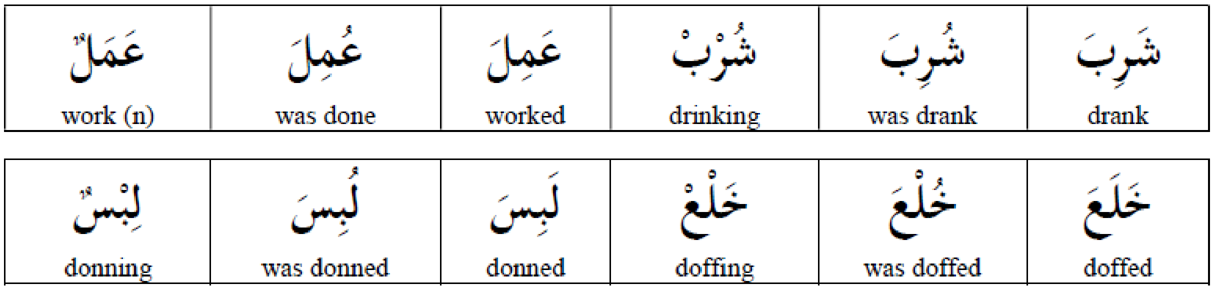
Figure 1.5: The NMT process

1.4. Challenges to Arabic language processing

1.4.1. Arabic script

One of the biggest challenges of Arabic language processing is the Arabic script itself [5], due to the lack of letters representing short vowels, absence of capitalization, multiple forms of letters and minimal punctuation, that's why it is challenging to processing compared to European languages for example:

In the Arabic text short vowels are represented by diacritics "التشكيل" which are a specific marks above or below the letters indicating what kind of short vowel is used, having this instead of dedicated latter increases the processing complexity, meaning each word might have multiple meaning depending on the use of diacritics as shown in figure (1.6) .



عَمَلٌ	عُمِلَ	عَمِلَ	شُرْبٌ	شُرِبَ	شَرِبَ
work (n)	was done	worked	drinking	was drank	drank
لِبْسٌ	لُبِسَ	لَبِسَ	خَلْعٌ	خُلِعَ	خَلَعَ
donning	was donned	donned	doffing	was doffed	doffed

Figure 1.6: representing how diacritics affect the meaning of the words

Adding to that in most Arabic text there is no diacritics; the reader is expected to predict them based on his prior knowledge.

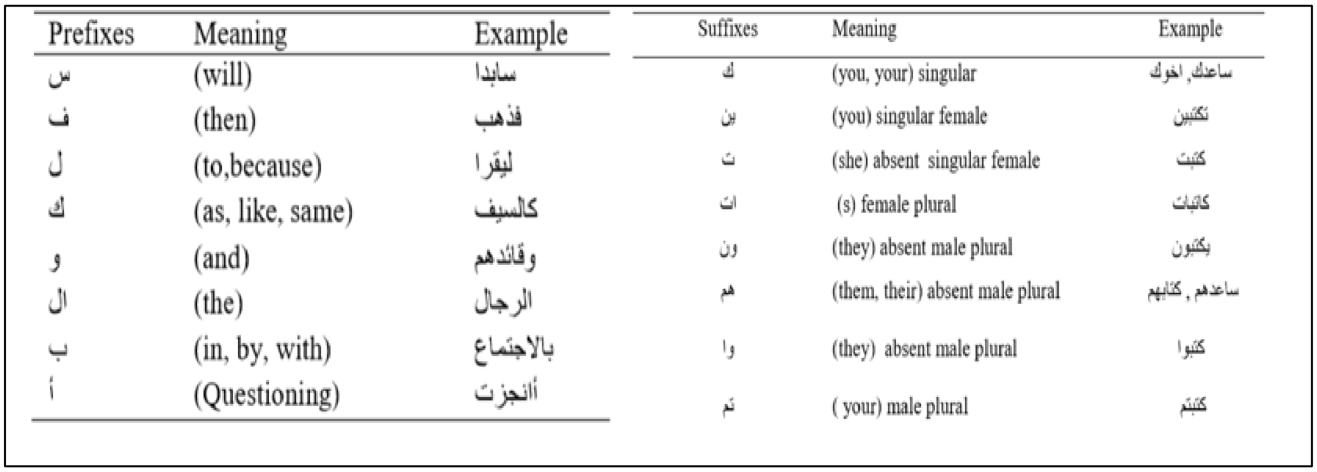
Arabic letters have multiple forms depending on their position for example: the letter "غ" (ghain) changes depending on whether it's in the beginning, the middle, the end of the word (غغغ).

In most European languages, sentences usually begin with an upper class letter and end with a period, which helps in the translation process since it is crucial to know exactly the beginning and the end of the sentences. However, in languages like Arabic, Japanese, Chinese, the absence of capitalization makes it difficult to know the boundaries of sentences, thus complicating the processing of their text.

1.4.2 Arabic morphology:

Arabic is considered a morphologically rich language [6] (MRL) which means that the grammatical nature of words is not necessary indicated by its position in the sentence but rather with a change in the word itself, these changes are fused into the words and it is often hard to define rules that separate the root from the added particles

Arabic also has a wide variety of affixes, which are letters, added to the beginning or the middle or the end of the word altering its meaning [7].



Prefixes	Meaning	Example
س	(will)	سابدا
ف	(then)	فذهب
ل	(to,because)	ليقرا
ك	(as, like, same)	كالسيف
و	(and)	وقائدهم
ال	(the)	الرجال
ب	(in, by, with)	بالاجتماع
أ	(Questioning)	أانجزت

Suffixes	Meaning	Example
ك	(you, your) singular	ساعدك, اخوك
ين	(you) singular female	تكتبين
ت	(she) absent singular female	كتبت
ات	(s) female plural	كاتبات
ون	(they) absent male plural	يكتبون
هم	(them, their) absent male plural	ساعدهم , كتابهم
وا	(they) absent male plural	كتبوا
تم	(your) male plural	كتبتم

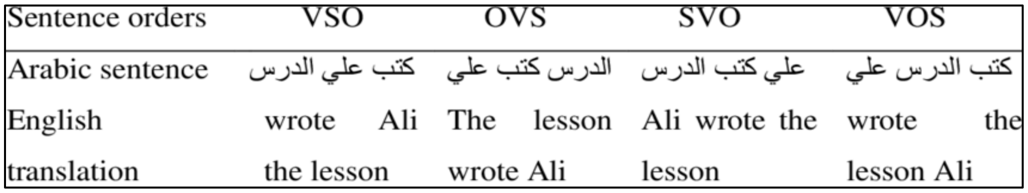
Figure 1.7: representing the Arabic affixes and their meaning

1.4.3 Arabic vocabulary

Arabic is the richest language vocabulary wise, according to a survey done by SEBIL Center in which they came up with an estimation of the size of the Arabic vocabulary, 12 million non-redundant Arabic word compared to 600 thousands English word, this gap between the vocabulary sizes makes it difficult to translate to and from.

1.4.4 Free word order

Most languages have somewhat a strict word order, meaning that for a sentence to have meaning it must follow a specific structure, for example in English we say “ Ali wrote the lesson” subject+verb+object, but in Arabic that sentence can be written in many different ways having the same meaning .



Sentence orders	VSO	OVS	SVO	VOS
Arabic sentence	كتب علي الدرس	الدرس كتب علي	علي كتب الدرس	كتب الدرس علي
English translation	wrote Ali the lesson	The lesson wrote Ali	Ali wrote the lesson	wrote the lesson Ali

Figure 1.8: representing the difference in word order of Arabic and English sentences

All these special characteristics of the Arabic language requires a complex preprocessing algorithms to tackle all these issues, resulting in a better quality training data.

1.5. Data view

Natural language processing domain and especially Machine translation domain acknowledges that is futile to try to write down all the rules and dictionaries that govern languages [8], but rather that all the necessary information to perform translation should be extracted directly from large amounts of translation examples.

Text corpora or text corpus is a collection text and there two main types of it: monolingual and parallel. Acquiring a large amount of text in single language will help us learn a lot about said language the word usage the structure of sentences and there is even a possibility to translate purely from a large amount of monolingual text with the help of Unsupervised Machine Translation. However, a better resource to use in MT is parallel corpora that comes in the form of sentence pairs, a source sentence and its translation.

1.5.1 Data adequacy

The process of machine translation relies heavily on data adequacy, the ability to match source text to its translation. Let us take for example the word (protection) which can have multiple Arabic translation depending on the context.

To predict a word translation a computer counts how many times the word (protection) is translated to those possible Arabic translations:

Protection	$\longrightarrow$	(الحماية)	15,365
Protection	$\longrightarrow$	(الوقاية)	10,985
Protection	$\longrightarrow$	(الأمن)	3,222

From what we see, the most likely translation is (الحماية), but (الوقاية) is also a high possibility meaning the system would be wrong many times. However, we can improve upon that by considering the boundaries.

Civil Protection	→	(الحماية المدنية)	2012
Civil Protection	→	(الوقاية المدنية)	2
Civil Protection	→	(الأمن المدني)	365

This example illustrates contextual information can improve the prediction of the correct translation significantly. However, there will be always times were the system predicts the wrong translation Hence, the engineering mantra of data-driven machine translation research is not to achieve perfect translation, but to drive down error rates [8].

1.5.2 Data Fluency

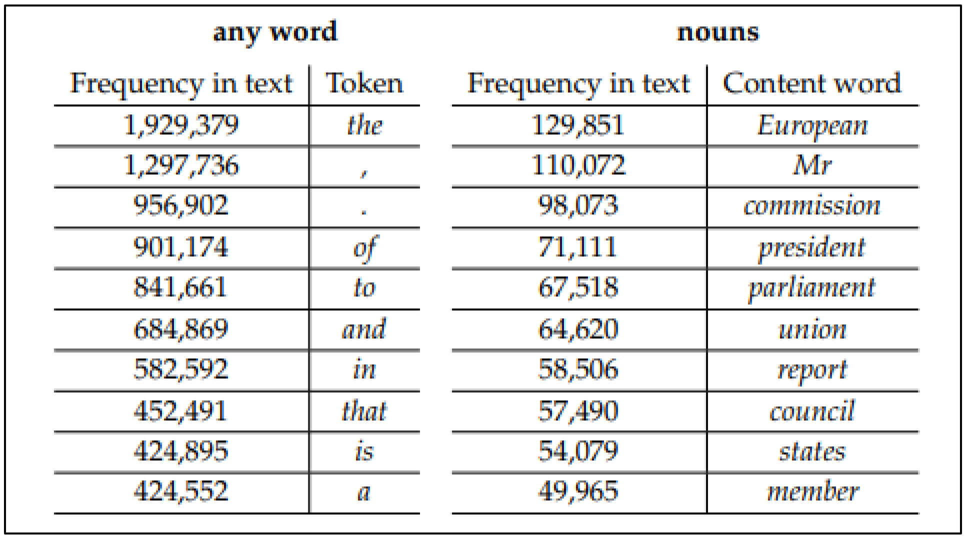
Parallel corpora does more than just matching words with their translation, it helps with arranging words in the correct way to insure the fluency of the output. This involves selecting the right word order the right function words and the convenient phrasing for the context.

Corpora data would tell us for example, 'he went to school 'is a better ordered sentence than 'school he to went' because the first sentence has been observed many times by the system making it a better choice.

1.5.3 Zipf's Law

Sparsity is a major issue in data driven methods. Let us assume that we have a corpus containing 100 million English words with 100,000 valid word, one would assume that this is a very rich statistics to learn about the language but unfortunately this is not the case.

The distribution of words in a corpus is highly skewed (see figure 2.9). Zipf's law states that the frequency f of a word (or its count in a corpus) multiplied with its rank r when words are sorted by frequency is a constant k.



any word		nouns	
Frequency in text	Token	Frequency in text	Content word
1,929,379	*the*	129,851	*European*
1,297,736	*,*	110,072	*Mr*
956,902	*.*	98,073	*commission*
901,174	*of*	71,111	*president*
841,661	*to*	67,518	*parliament*
684,869	*and*	64,620	*union*
582,592	*in*	58,506	*report*
452,491	*that*	57,490	*council*
424,895	*is*	54,079	*states*
424,552	*a*	49,965	*member*

Table 1.1: the most frequent words in a version of the English Europarl corpus that consists of 30 million words

Zipf’s law predicts that no matter how big the corpus is there will be many rare words in it, Gathering larger corpora will increase the frequency of words, but also reveal previously unseen words with low counts.

1.5. Conclusion

The need for a faster and cheaper way of translation gave birth to machine translation, which came with many challenges and solutions. In this first chapter, we introduced the concept of machine translation; we talked about the currently used techniques more specifically the state of the art neural machine translation systems; how it works, its limitation, and how it stacks against the previous techniques, then we elaborated on the NMT process and highlighted its important phases.

Then we spoke about Data preprocessing and the significant improvement it adds to the performance of the NMT systems, especially in the case of complex languages like Arabic.

After that, we talked about the unique characteristics of the Arabic language that makes it pose a challenge for machine translation.

Finally we discussed the data side and how its problematic when it comes to machine translation.

Chapter 2 : Text Segmentation

2.1 Introduction

When analyzing a natural language text, it is necessary to define its building blocks, defining these units is considered a challenge, especially when we take in consideration the diversity of human languages each with its unique writing rules, Natural languages contains various ambiguities and finding a solution to this particular problem is one of the main challenges of NLP.

Text segmentation is the process of splitting a sequence of characters into linguistically meaningful parts called segments or tokens, these segments can be characters, words, sentences or any type of information unit depending on the end goal of the text analysis [8].

Segmentation is an important step in text processing it allows us to extract coherent data, which we can help apply all sort of different NLP tasks such as Machine Translation, summarization, POS tagging in a more efficient and effective way.

Word segmentation also referred to as tokenization is the process breaking up a character sequence by identifying the word boundaries which is where a word ends and another start, this results in what is referred to as tokens, this process can be challenging especially in languages with no word boundaries markers in their writing system .

Sentence segmentation on the other hand is the process of splitting the text into sentences containing one or more words it is done by defining the sentence boundaries, in application word segmentation and sentence segmentation overlap each other one cannot be achieved without the other [8].

Segmentation algorithms are heavily dependent on the targeted language meaning a fine tuned segmentation algorithm for a specific language would be completely inadequate for another.

Arabic is considered a complex language that poses many challenges in NLP field, it has a rich morphological system and ambiguous writing rules [1], so when implementing a segmentation scheme for Arabic there are multiple aspect to take in consideration in order to achieve a correct segmentation.

2.2. Motivation behind text segmentation:

Assuming we want to analyze a specific language script, in order perform NLP tasks such as Machine translation. We will have to deploy an automatic mechanism to go through large amount of data trying to define features and find patterns specific to said language script, however this process will heavily depend on the coherence and the structure of the data to analyze.

Let us assume we want to translate a text written in English language to Arabic and we have a large aligned bilingual corpus English-Arabic meaning thousands of English examples with their respective translation, Many MT approaches use a statistical model, which relies heavily on counting the number of occurrences of a token in order to predict its translation.

This would have complications when dealing with languages like Arabic language for example:

Arabic	(لسيارته)	(سيارته)	(سيارة)	(السيارة)
English	(for his car)	(his car)	(car)	(the car)

The Algorithm will assume that each token from the example above is a unique word but in fact, it is a composition of words:

(ه \ سيارة \ ل)	(for / his / car)

NMT systems will be affected more with this problem since it usually operates on closed vocabulary corpus meaning it assumes a fixed vocabulary size, since it embeds each word of the vocabulary to a sequence on bits (word embedding).

This would affect the performance of the translation significantly if not addressed properly.

Segmentation has the potential to solve this particular issue, which will improve the training data quality, and by extension, the performance of the algorithms applied on it.

2.3. Challenges to text segmentation

What makes segmentation a challenging task is the fact that there are many factors to take in consideration when developing an algorithm for text segmentation.

In the next section, we will emphasize on the main types of issues that we must address when devolving such algorithms.

2.3.1. Language dependency

When talking about text segmentation we mainly focus on written languages that have an established writing system this is an important distinction because there are thousands of spoken languages and dialects but only small percentage of them have a system of symbols and rules representing the information in said language.

Languages can be categorized into Logographic; where each symbol represents a word resulting in a language having thousands of unique characters, syllabic in which symbols represent syllables, or alphabetic in which symbols are a representation of sounds, most modern language writing systems uses a combination of types so no writing can be classified as purely alphabetic , syllabic , logographic [9].

Adding to the diverse symbol types used in writing systems languages often use some sort of orthographical rules to denote the boundaries between linguistic units, for example English language uses white space to indicate word boundaries ,punctuation marks like (, . ; :) to indicate sentence boundaries making English a fairly easier text to segment, on the other hand languages like Arabic and Japanese don’t have explicit markers for words and sentence boundaries for example (ذهبنا للمتجر) which translate to (we went to the store) as we can see it is much harder to indicate the word boundaries in an Arabic script

This wide variety of writing systems complicates the process of text segmentation since we have to take in consideration all these language-specific and orthography-specific features in order to achieve a proper segmentation meaning each technique is going to be heavily dependent on the language it was developed for.

2.3.2. Data dependency (Corpus dependency)

The robustness of NLP systems is considered a problem since these systems need a well-formed input with a predefined structure for them to perform well.

Large corpora that includes multiple sources (newswire, emails, OCR data, social networks data) usually contains misspelling, erratic spacing and punctuation causing algorithms that requires a well-formed input to be much less successful on these texts, this is why it became clear that we need to develop segmentation algorithms capable of handling such irregularities.

2.3.3. Text segmentation evaluation

Evaluating and comparing segmentation algorithms very difficult because an algorithm may perform well on a specific corpus and not be successful on another, in other words an algorithm fine-tuned for a specific language (A) will most likely be inapplicable on language (B) , however there are some common evaluation algorithms for word and sentence segmentation that give information on their performance.

Word segmentation performance is usually measured with recall and precision where recall is defined as the percent of words in the manually segmented text identified by the segmentation algorithm [9], and precision is the percentage of words returned by the algorithms that occur in the same position as in the reference-segmented text.

Sentence segmentation performance score on the other hand is a single score equal to the number of punctuation marks correctly classified divided by the total number of punctuation marks.

So even with these evaluation algorithms it is only possible to compare between text segmentation algorithms only when applied on the same benchmark corpus that means in real world application we cannot know which algorithms will perform better only after testing it.

2.4. Related works:

Arabic word segmentation is considered as part of POS tagging problem [10], There have been many approaches conducted in the field of Arabic word segmentation the two main categories are statistical and rule-based approaches.

Rule-based approaches such as Khoja [11], and more recently Hadni et al. [12], which used a combination of lexicon and a predefined set of morphological rules to identify affixes and clitics.

As for the statistical approaches two main strategies are used , the first is to consider the segmentation as a classification problem and separate it from the POS tagging process then use machine learning to train the classifiers, various techniques have been applied (SVM) like in the work of A. Pasha et al. [13]. And k-nearest neighbor algorithm as in the work of M. Abdul-Mageed, M. Diab, S. Kübler [14]. The second strategy is that the segmentation is conducted during the morphological analysis, which contains the segmentation information and the POS tagging, an example to this is the work of Habash and Rambo [15] in which they used the SVM to choose the best solution. Zalmout and habash[16] used BI-LSTM taggers with various embedding levels to perform the morphological analysis.

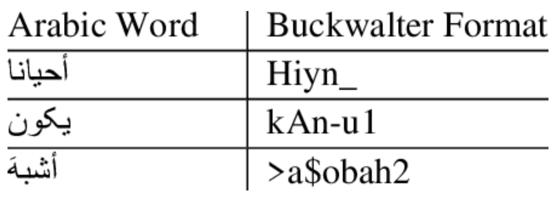
Other studies have used DNNs to perform word segmentation, a configuration of RNNs using BI-LSTM cells have been used for Chinese word segmentation and POS tagging [17] the reported experiments show F1 value above 97% which comparable to the state-of-art techniques .

One of the recent work on Arabic word segmentation is Farasa [18] in which they break the Arabic word into their constituent clitics based on SVM rank with linear kernels, taking in consideration the likelihood of stems, prefixes, suffixes and their combination. Farasa uses lexicons to solve the problem of rewriting during the segmentation, it was trained on different part of ATB and tested on a corpus containing 70 wiki news articles achieving an F-1 score of 98.94.

Other works that focuses on Arabic segmentation is MADAMIRA [13] a morphological analysis and disambiguation tool for Arabic text that combines the best features from its predecessors, MADA (Habash and Rambow, 2005; Habash et al., 2009; Habash et al., 2013) and AMIRA (Diab et al., 2007). It uses a streamlined java implementation making it faster and more robust than its predecessors, it achieved an F1- score of 99.1 on test data composed of 25k words.

2.4.1 MADAMIRA:

MADAMIRA [13] follows the design shown in figure (2.2). First, the data must be processed and transformed into Buckwalter format (see figure2.1) used within the MADAMIRA tool, and then the data is passed to the morphological analysis component, which develops a list of all possible analysis each word. The results are then passed to the feature-modelling component in which SVM and Language models are used to predict the word's morphological features. SVM for closed class features and language models for open class features, an Analysis ranking follows up which evaluates each word's analysis and then ranks them based on their score. The top ranked analysis is then passed to the tokenization component, which produce the desired tokenization for the word based on the scheme selected by the user. The next component Base phrase chunking is used to divide text into phrases and lastly Named Entity recognizer is used to divide text into phrases and to mark and categorize named entities within the text.



Arabic Word	Buckwalter Format
أحيانا	Hiyn_
يكون	kAn-u1
أشبهَ	>a$obah2

Figure 2.1: Buckwalter format

MADAMIRA impresses with its 11 tokenization schemes including ATB, D3, D3_BWPOS providing various segmentation option for the user to choose from. The word throughput for MADAMIRA is rated at 1013 word/sec, which is considerably faster than the Previous AMIRA system that tops at 255 words/sec.

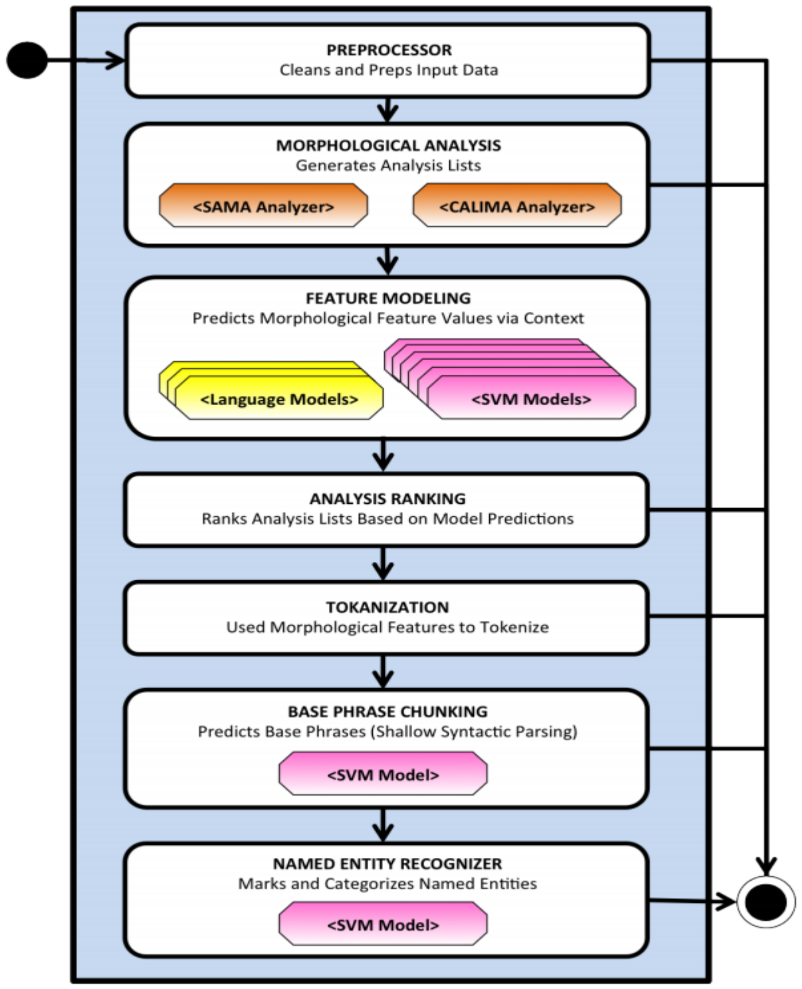

PREPROCESSOR
Cleans and Preps Input Data
MORPHOLOGICAL ANALYSIS
Generates Analysis Lists
<SAMA Analyzer>
<CALIMA Analyzer>
FEATURE MODELING
Predicts Morphological Feature Values via Context
<Language Models>
<SVM Models>
ANALYSIS RANKING
Ranks Analysis Lists Based on Model Predictions
TOKANIZATION
Used Morphological Features to Tokenize
BASE PHRASE CHUNKING
Predicts Base Phrases (Shallow Syntactic Parsing)
<SVM Model>
NAMED ENTITY RECOGNIZER
Marks and Categorizes Named Entities
<SVM Model>

Figure 2.2: Overview of MADAMIRA Architecture (adapted from [13])

2.4.2 Ferasa

Ferasa is an Arabic word segmenter based on SVM-rank with the use of linear kernels [19] it claims to be on part with state-of-art Arabic segementers (MADAMIRA and Stanford) while being significantly faster and more efficient.

Ferasa uses a set of conditional probability to determine prefixes and suffixes and stems as for any given word it extracts all character level segmentation possibility leading to a sequence of segments as shown in figure (2.3):

$$prefix_1+\ldots+prefix_n+\text{stem}+suffix_1+\ldots+suffix_m$$

Figure 2.3: Ferasa sequence structure

These prefixes and suffixes are predefined as follows:

- Valid prefixes: f, w, l, b, k, Al, s. (ف, و, ال, ب, ك, ل, س).
- Valid suffixes: : A, p, t, k, n, w, y, At, An, wn, wA, yn, kmA, km, kn, h, hA, hmA, hm, hn, nA, tmA, tm, tn (ا,ة,ت,ك,ن,و,ي,ات,ان,ون,وا,ين,كما,كم,كن,ه,هما,هم,هن,نا,تما,تم,تن,).

Also it generates a list of possible prefix and suffix combination for example (وبالعلم) which contains a prefix combination (و+ب+ال+علم) (w + b + al+Elm).

The learning process starts by constructing feature vectors for each possible segmentation and marking the correct segmentation for each word, and then a SVM-rank algorithm is used to learn features weights, during test all possible segmentation with valid prefixes and suffixes are generated and then given a score by the classifier ranking them from most likely to unlikely.

For training Ferasa used Parts 1,2 and 3 of the Penn Arabic Treebank (ATB), as for testing it was done test set composed of 70 WikiNews articles (from 2013 to 2014) covering various themes: politics, economics, health, science and technology, sports, art.

Ferasa achieved state-of-art F-1 score measured at 98.94% while being significantly faster than its competitors like MADAMIRA (F-1 score 99.1%).

2.4.3 Stanford segmenter

Stanford segmenter [20] uses a single clitic segmentation model that is accurate on both MSA and informal Arabic. It is an extension of the character-level conditional random field (CRF) model of Green and DeNero (2012). It handles two Arabic orthographic normalization rules that commonly require rewriting of tokens after segmentation. It also adds new features that improve segmentation accuracy and shows that dialectal data can be handled in the framework of domain adaptation. It also makes use of a simple feature called space augmentation (Daumé, 2007) that yields significant improvements in task accuracy.

Green and DeNero use a linear-chain model with X as the sequence of input characters, and Y* chosen according to the decision rule:

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} \sum_{i=1}^{N} \theta^\top \phi(\mathbf{X}, y_i, \ldots, y_{i-3}, i) \ .$$

Figure 2.4: decision rule for the Green and Denero model

Stanford segmenter is equipped to handles some Arabic-specific orthographic rewrites, as for example the letter (ت) becomes (ة) when the word is segmented and it occurs on final position.

Green and DeNero is a third-order Markov CRF, employing the following indicator features:

- A five-character window around the current character.
- N-grams consisting of the current character and up to three preceding characters.
- Whether the current character is punctuation.
- Whether the current character is a digit.
- The Unicode block of the current character.
- The Unicode character class of the current character.

In addition to these, it includes two more features specific motivated by the errors in the original system:

- Word length and position within a word
- First and last two characters of the current word, separately influencing the first two labels and the last two labels

The segmenter was trained and evaluated on Arabic Tree Bank (ATB), the Broadcast news treebank (BN), and Egyptian Arabic Treebank (ARZ), it achieved the following results:

- F1 score of 98.24- 97.39 – 92.09 in (ATB, BN, ARZ) respectively.
- TEDEval score of 98.74 – 98.29 – 92.32 in (ATB, BN, ARZ) respectively.

2.4.4. Byte Pair encoding (BPE)

Byte pair encoding (Gage, 1994) is a data compression technique that replaces the most frequent pair of bytes in a sequence with a single unused byte.

It was adapted into the NMT field in the work of Rico Sennrich [21]; BPE addresses the out of vocabulary Problem and enables NMT systems to perform open-vocabulary translation by encoding rare and unknown words as sequences of sub-word units.

BPE operates as follows (See figure 2.5):

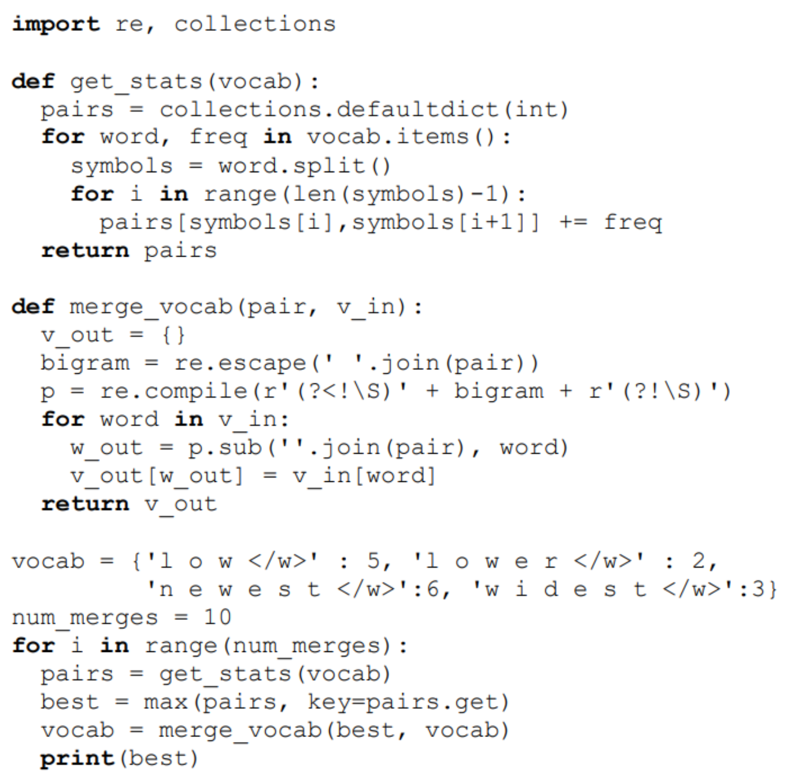
- Step 0: initializing the vocabulary of the corpus.

- Step 1: represent each word of the corpus as a sequence of characters while marking the word boundaries with special symbol that allows us the original tokenization after the translation is done

- Step 2: iteratively count all symbol pairs and replace each occurrence of the most frequent pair (‘A’, ‘B’) with a new symbol ‘AB’.

- Step 3: Merge every occurrence of the most frequent pair, add the new character n-gram to the vocabulary.

- Step 4: Repeat step 3 until the desired number of merge operations are completed or the desired vocabulary size is achieved.



```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

Figure 2.5: Minimal Python implementation of BPE algorithm (adapted from [27])

To apply BPE on the data, first, each word is split to a sequence of characters; the trained BPE model then applies the learned operations to merge the characters into larger, known symbols. This is applicable to any word, and allows for open-vocabulary networks with fixed symbol vocabularies.

2.5 Evaluation metrics

2.5.1. Word segmentation evaluation

a- Precision, Recall, F-1 measure:

When evaluating a segmentation technique the standard is to calculate word level precision, recall and their evenly weighted average F1-score to measure the performance of said technique.

The correctly segmented words are regarded as true positives (TP), and the total words returned by the segmenter are considered prediction positives (PP), to calculate the precision of a technique we use this formula:

Precision= TP /PP ………….. (1)

As for recall function, we simply divide true positives (TP) by reference positives (RP) which is the total number of words in the reference text:

Recall = TP/RP……………….. (2)

The F-1 measure is a combination of both precision and recall into a single measurement capturing both properties:

*F-1 Measure = (2 * Precision * Recall) / (Precision + Recall) … (3*

B- TEDEval

TEDEval is a novel metric for evaluating joint segmentation and parsing scenarios [22], it uses distance-based metrics defined for the space of trees over lattices.

TEDEval uses a cost function that assigns a cost unit the possible actions 'a' (adding, removing a lexeme) and then defines the cost of a sequence (a_1… a_n) an editing script (ES) performed on a tree Y2 (predicted segmentation) to transform it to Tree Y1 (reference segmentation) as shown in the following formula:

$$\mathrm{TED}(y_1, y_2) = \min_{\mathrm{ES}(y_1,y_2)} \mathrm{C}(\mathrm{ES}(y_1, y_2))$$

This cost is cosidred an error measurement which need to be normalized and transformed into a score, assuming we have the predicted structure p and the gold structure g we can extract and evaluation score using this formula:

$$\textsc{TedEval}(p, g) = 1 - \frac{\mathrm{TED}(p, g)}{|p| + |g| - 2}$$

The term |p| + |g| − 2 is a normalization factor defined in terms of the worst-case scenario, in which the parser has only made incorrect decisions. We would need to delete all lexemes and nodes in p and add all the lexemes and nodes of g, except for roots.

2.5.2. Machine translation evaluation

BLEU score:

It is critical to have a valid comparative metric to help guide research and provide a way to compare between outputs of different translation systems, however translation is an ambiguous task, meaning even two professional human translators would most likely produce a different translation to the same text. Adding to that not only we need to worry about matching words but also their order some languages like Arabic for example have free word order meaning there multiple valid orders to a sentence and knowing which is correct and not complicates the process even more.

That is why researchers at IBM came up with metric that compromises between ignoring and requiring matching word order, they called it BLEU. The idea behind BLEU score is that it does not just count the matching words from the output and the reference but also larger order N-grams [23] meaning it rewards the matching word order too (see figure (2.6)).

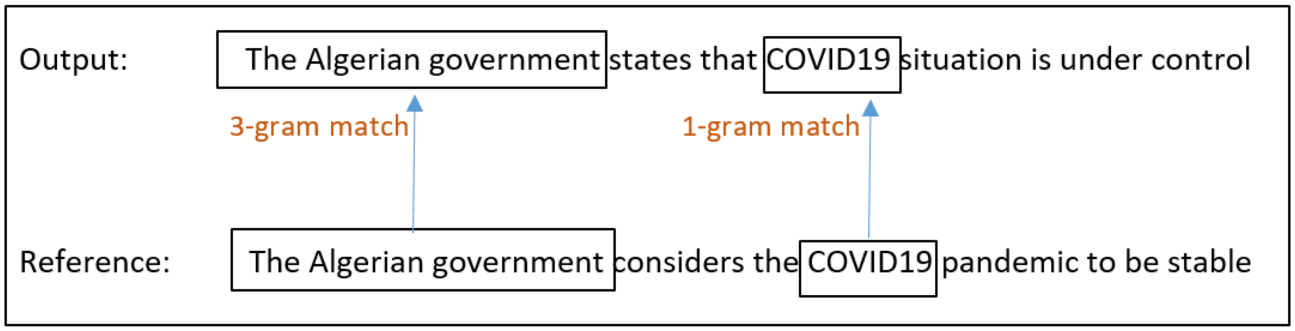

Output:
The Algerian government states that COVID19 situation is under control
3-gram match
1-gram match
Reference:
The Algerian government considers the COVID19 pandemic to be stable

Figure 2.6: Example of matching N-grams

The BLEU metric uses a brevity penalty it is based on the ratio between the number of words in the machine translation and reference translation.

The BLEU metric is defined as shown in the following formula:

$$\text{BLEU} = \text{brevity-penalty} \times \exp \sum_{i=1}^{4} \log \frac{\text{matching } i\text{-grams}}{\text{total } i\text{-grams in machine translation}}$$

$$\text{brevity-penalty} = \min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)$$

METEOR metric:

The METEOR [23] metric introduced a couple of novel ideas. One perceived flaw of BLEU is that it gives no credit at all to near matches. Meaning if the system translates the Arabic word (الحماية) to (safe) but in the reference they used the word (safety) even though they have similar meaning but blue score consider that translation to be a mismatch and gives no to credit to it. However meteor by the use of stemming can match the correct segments (safe = safe+ty) and acknowledge as a near miss.

Another way to detect near misses is using synonyms, or semantically closely related words. Meaning if the system translates the Arabic word (الحماية) to the word (security) instead of (safety) which is close in meaning may be irrelevant in bringing across the meaning of the sentences and should not be penalized.

METEOR incorporates the use of stemming and synonyms by first matching the surface forms of the words, and then backing off to stems and finally semantic classes. The latter are determined using Wordnet, a popular ontology of English words that also exists for other languages. The main drawback of METEOR is that its method and formula for computing a score is much more complicated that BLEU's. Linguistic resources such as morphological stemmers and synonym databases are required. The matching process involves computationally expensive word alignment. There are many more parameters – such as the relative weight of recall to precision, the weight for stemming or synonym matches – that have to be tuned.

2.7 Conclusion

One weakness of neural machine translation is its robustness concerning the training data. The data must be formatted, cleaned, and structured adequately for it to be properly digested by the neural network; as a result, the performance of these RNNs is tied directly to the quality of the training data.

In this chapter, we talked about text segmentation (tokenization), and how it can drastically improve the performance of the translation algorithms, and then we highlighted some of the main challenges that faces text segmentation.

After that, we discussed some of the current state of the art segmentation techniques such as MADAMIRA and Ferasa and we shared their benchmarks.

Then we finished the chapter with some of the evaluation metrics for both the segmentation and the translation.

Chapter 3 : Alignment-based Segmentation and results

3.1. The proposed method:

Most current text segmentation algorithms that are aimed toward Machine translation relies on monolingual corpus in training, meaning it extracts all the necessary features and the linguistic rules directly from the corpus of the languages intended for.

In this work, we want to propose a new method for text segmentation in which we make use of parallel corpus in the training process to try to breach the gap between the source language and the target language in the translation process. We theories that having a segmentation scheme tailored specifically for a specific pair of languages would yield better translation results.

The idea revolves around three common concepts in the machine translation field. Text Alignment, which is having each sentence from the source language side by side its translation from the target language. And occurrence counting which is how many times a specific word from source language is translated to the same word from the target language. Lastly, vocabulary size manipulation, a corpus vocabulary is the set of unique words used in the text corpus; it can also be character-based (e.g. each letter).

Our technique uses mentioned concepts to perform the segmentation, first we start by separating the characters on the Arabic side so that we have each letter by itself for example: (المدرسة) becomes (ا ل م د ر س ة). Then we align the training data, in our case an Arabic- English parallel corpus. Then Next step is to count the occurrences of sequences of Arabic characters that aligns with complete English words and sort them based on their frequency from highest to lowest. This would help us extract and filter the phrases from the Arabic side of the corpus that match complete words from the English side of the corpus, and then we train BPE model on the extracted phrases to joint those sequences of characters.

In short, we want to joint Arabic sequence of characters, which have one English word aligned with consistency. See figure (3.1) for an overview of Alignment-based Segmentation technique.

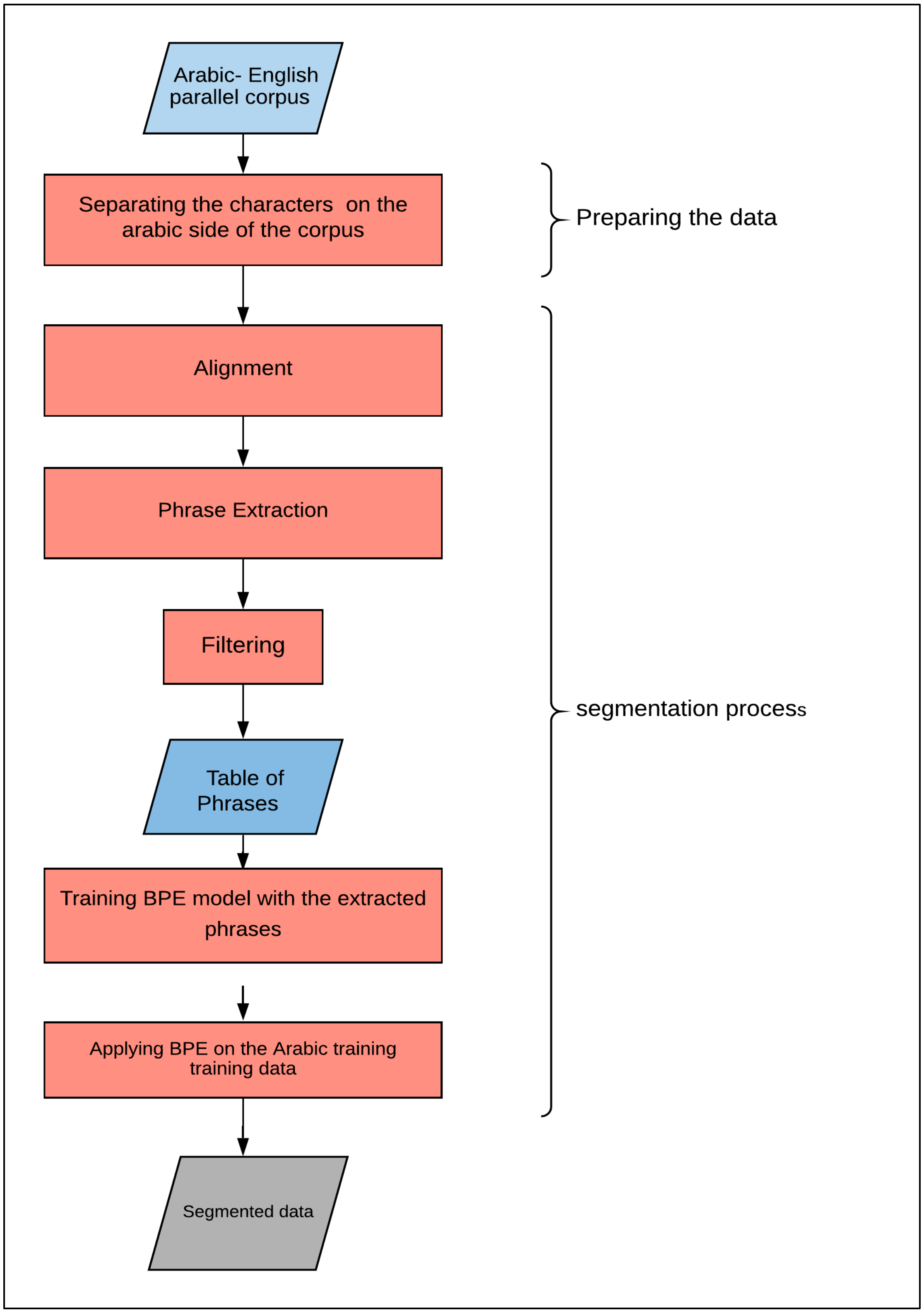

Arabic- English parallel corpus
Separating the characters on the arabic side of the corpus
Preparing the data
Alignment
Phrase Extraction
Filtering
segmentation process
Table of Phrases
Training BPE model with the extracted phrases
Applying BPE on the Arabic training training data
Segmented data

Figure 3.1: flowchart representing an overview for the proposed method

3.2. Used Hardware and Software:

A. hardware:

Most of our experiments are conducted on a desktop computer with the following specifications:

- CPU: AMD Ryzen 5 2600x series 6 cores / 12 threads running at 4.0 GHZ
  GPU: Nvidia GeForce 1070ti with 8 GB of ddr5 Vram and 2432 CUDA cores
- Memory: 16 GB of dual channel RAM running at 3200 MHZ
- Storage: 128 GB M.2 nvme SSD + 1000 GB HDD + 500 GB SSD

B. software:

Throughout our experiments, we used various software and tools and programming languages all running on Linux OS

- OS: Linux Ubuntu 18.04 LTS

For the segmentation:

Our segmentation algorithm is implemented using python with the use of other softwares such as

- `fast-align`: which is a simple, fast, unsupervised word aligner.
- `sentence piece`: to apply BPE algorithm
- `giza aligner:` used for alignment
- `moses:` collection of language processing tools

For translation:

We used Open-NMT[1] to create translation models; it is an open source ecosystem for neural machine translation and neural sequence learning. Open-NMT Started in December 2016 by the Harvard NLP group and SYSTRAN.

[1] https://opennmt.net/

3.3 Used Data:

Our training and testing was conducted on TED talks parallel corpora.

TED parallel Corpora is growing collection of Bilingual parallel corpora, Multilingual parallel corpora and Monolingual corpora extracted from TED talks[2] for 109 world languages. It includes Monolingual corpus, 12 languages for Bilingual parallel corpus over 120 million aligned sentences and 13 languages for Multilingual Parallel corpus with more than 600k sentences. The goal of the extraction and processing was to generate sentence-aligned text for machine translation systems.

We use the official IWSLT 2017 evaluation campaign Data. The transcripts are given as pure text (UTF8 encoding), one or more sentences per line, and are aligned (at language pair level, not across pairs).

The Arabic-English corpus we used is composed of about 400k sentences (see tables bellow)

Data set	Sentences	tokens	
		Arabic	English
Training	400,000	3,400,000	4,200,000
Validation	1830	20,000	26,000
Testing(IWSLT17.TED.tst2014)	1316	18,000	21,000

Table 3.1: *Summary of statistics of the training and test data*

	Arabic side			English side		
Rank	Token	occurrence	Coverage (%)	Token	occurrence	Coverage (%)
1	في	98929	2.87	The	198181	4.72
2	من	85804	2.49	And	142978	3.40
3	أن	47220	1.37	To	117732	2.80
4	على	41871	1.21	Of	109176	2.60
5	و	38636	1.12	A	100429	2.39
6	هذا	29426	0.85	That	76710	1.83
7	ما	25060	0.73	In	73816	1.76
8	هذه	23349	0.68	I	61580	1.47

[2] www.ted.com

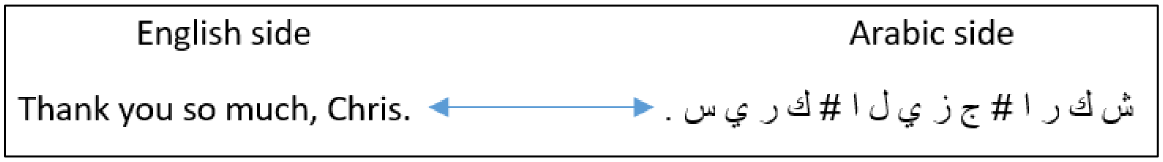
9	لا	22784	0.66	Is	58992	1.40
10	هو	20426	0.59	you	55564	1.32

Table 3.2: some statistics about the Arabic vocabulary

3.4. Implementation details

3.4.1. Character separation:

The first thing we wanted to do is reduce the vocabulary size of the Arabic side of the corpus, meaning we separate the Arabic characters from each other so that we have each letter by itself. Then we mark the words boundaries so that we be able retrieve the original data ones the process is done and to do that we simply insert '#' at the start of each word see figure 3.2 for an example.


English side

Arabic side

Thank you so much, Chris.

ش ك ر ا # ج ز ي ل ا # ك ر ي س .

Figure 3.2: format of the training data

And to do that we used the command 'sed'. SED command in UNIX is stands for stream editor and it can perform lots of function on file like, searching, find and replace insertion or deletion:

```
cat "ar-cleanv2.txt" | sed -e "s/ /#/g" -e "s/\(.\)/\1 /g" -e "s/\(\s\) */\1/g"> seperated.txt
```

3.4.2 Performing the Alignment:

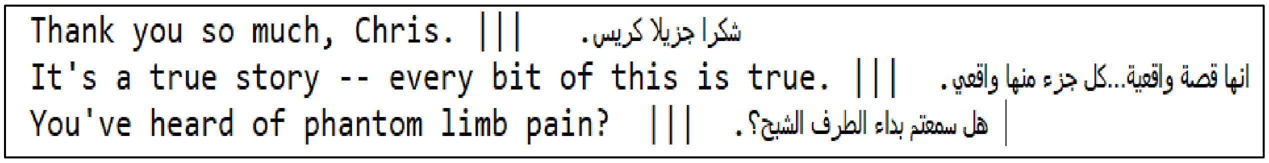
The next step is to perform the alignment. Most of the text alignment implementations are based on the Expectation Maximisation Algorithm.

Assuming we have a pair of language (A-B) we start by aligning language B according to language A meaning token in position '1' in the A language side is aligned with 'n' tokens from the side of language B.

Then we do the opposite we align language A according to language B so that we have for example tokens 'm' from language A side are aligned with the token in position '1' on the B language side.

Finally, we combine both ways alignments to get the alignment output ‘m, n’.

In our experiments we used fast align [24]. Its input must be tokenized and aligned into parallel sentences. Each line is a source language sentence and its target language translation, separated by a triple pipe symbol with leading and trailing white space (|||). See figure 3.4



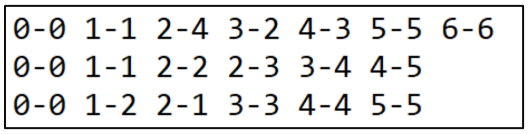
```
Thank you so much, Chris. |||  شكرا جزيلا كريس.
It's a true story -- every bit of this is true. |||  انها قصة واقعية...كل جزء منها واقعي.
You've heard of phantom limb pain?  |||  هل سمعتم بداء الطرف الشبح؟.
```

Figure 3.3: Fast align input format

Fast align generates asymmetric alignments, by treating either the left or right language in the parallel corpus as primary language being modelled. To generate source–target (left language–right language) we simply use the following command:

```
fast_align -i data.ar-en  -d -o -v
```

Fast align produces outputs in the widely used i-j “Pharaoh format,” where a pair i-j indicates that the ith word (zero-indexed) of the left language (by convention, the source language) is aligned to the jth word of the right sentence (by convention, the target language).



```
0-0 1-1 2-4 3-2 4-3 5-5 6-6
0-0 1-1 2-2 2-3 3-4 4-5
0-0 1-2 2-1 3-3 4-4 5-5
```

Figure 3.4: Fast align output format

3.4.3. Phrase Extraction and filtering:

We use the alignment output to align segments from the Arabic side that match complete words from the English, this would help us see which Arabic characters can be reliably merged.

To be able to see repetitive alignments so that we can extract phrases we sorted the resulted alignments by their frequency from highest to lowest see table (3.3) for an example.

Order	Alignments	Frequency
1	in ||| ف ي	3492
2	this||| ذ ه	3349
3	on ||| ل ى	3287
4	now||| ل آ	1961
5	like ||| ث ل	1335
6	So ||| ذ ل	988
7	things ||| ش ي ا ء	523
8	part ||| ج ز ء	517
9	still ||| ز ا ل	408
10	said ||| ق ا ل	361
11	person ||| ش خ ص	295

Table 3.3: The Table of phrases with their frequencies

Our alignment file contained more than 6 million alignments of which only about 100k had a frequency above five, we only focused on repeated alignments.

What we want here is to extract all the phrases from the Arabic side that matches complete words from the English side while having high frequency.

We filtered out all the alignments that occurred less than 5 time and the alignments that contained more than one word on the English side, and also all the alignment that had end of word marker.

3.4.4. Training and Applying BPE:

Byte pair encoding is a simple data compression technique, the idea behind it is to iteratively replace the most frequent pair of bytes with an unused byte, and this idea was adapted into word segmentation algorithms by merging the most frequent pair of characters or sequence of characters.

The reason we wanted to trained BPE model on the extracted phrases is so that when applying the BPE on the training data it would merge the characters based on the aligned phrases from the table.

We start by training the modified BPE model on the extracted phrases using the following command:

```
#Learn bpe from counts
python ~/subword-nmt/learn_bpe.py -i cnt4bpe.2.nbow -o v4k.p.2 -s 4000 --dict-input -b
#Apply the learned model
python ~/subword-nmt/apply_bpe.py -i train.ar.orig -c v4k.p.2 -o train.ar.seg.v4k.p2
```

We trained two different BPE models: Alignment-based model (our technique) and monolingual (standard BPE). We modified Sennrich's BPE [21] implantation to support the alignment input (see the appendix for the most relevant parts of the algorithm).

As you can see, the phrases from alignment-based model and the monolingual are quite different.

After that we simply apply the models on the training data and use it to train the NMT system

Alignment-based model	Monolingual model
أصبح ت	أص بحت
أن فس	أنفس نا
ا ستخدام	ا ستخدام
إق تصاد	اق تصا
بال طبع	بالط بع
بال فعل	بالف عل
صغير ة	ص غيرة
صور ة	ص ورة
طريق ة	طري قة
عظ يم	ع ظيم
ل شخص	لشخ ص
ل هذا	له ذا

Table 3.4: comparative table between the Alignment-based BPE model and standard BPE model

3.5. Experiments and results:

3.5.1. Our Neural machine configuration:

To test our segmentation we trained multiple neural networks using Open-NMT.py to perform translation between Arabic and English language. Training such Networks requires a lot of resources and it is very sensitive to hyper-parameters. To find the correct configuration experiments must be conducted.

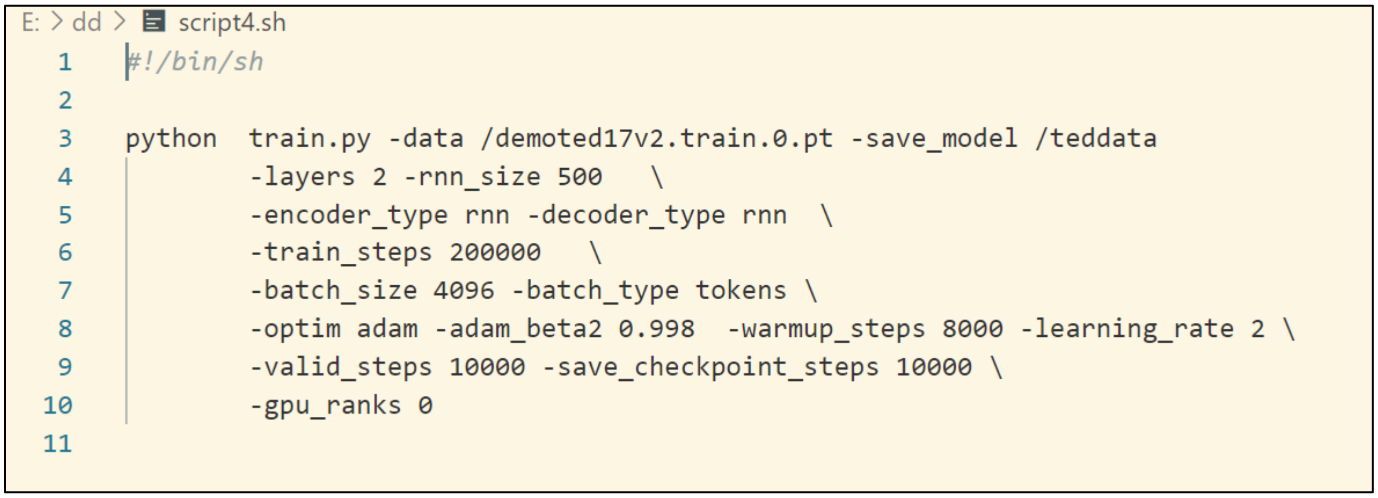
To give you an idea on how much resources it takes to train state of the art NMT systems see the table below.

Rank	Model	Training time
1	Transformer	3 days on 8 GPUS
2	SliceNet	6 days on 32 GPUS
3	GNMT	1 day on 96 GPUS

Table 3.5: training duration of state of the art NMT models

Choosing the correct configuration for the training is challenging task in itself we had to balance between the performance of the system and the training duration

After many experiments, we settled on the following the configuration see figure 3.5:


E: > dd > script4.sh

```
#!/bin/sh

python  train.py -data /demoted17v2.train.0.pt -save_model /teddata
        -layers 2 -rnn_size 500   \
        -encoder_type rnn -decoder_type rnn  \
        -train_steps 200000   \
        -batch_size 4096 -batch_type tokens \
        -optim adam -adam_beta2 0.998  -warmup_steps 8000 -learning_rate 2 \
        -valid_steps 10000 -save_checkpoint_steps 10000 \
        -gpu_ranks 0

```

Figure 3.5: translation model configuration

Since our hardware contains only one GPU, to be able to conduct multiple experiments we had to settle for a small neural network containing two layer LSTM with 500 hidden units on both the encoder and decoder trained for a maximum of 200k steps.

When training the model with different types of data we had various convergence rates depending on the data some reached their best result before the trained was completed, we noticed that after 100k steps many models suffered from over fitting. However, since we can save checkpoints, we can simply take the checkpoint with the best performance and stop the training if we see no improvement after a specific number of steps.

200k steps equates to approximately 8 hours of training with the configuration mentioned above, some models took about 5 hours to reach their best performance some showed no sign of convergence even after 20 hours.

The Table below shows the training duration of some of the models we trained with different processing on the training data

Data type	Training steps (best checkpoint)	Duration
Bpe (4k-4k-i)	95000	3h 49min
Bpe(4k-4k)	110000	4h 1min
raw	200000	8h
segchar	200000	8h

Table 3.6: Training duration for some of our experiments

3.5.2 English to Arabic translation results:

We performed 12 different experiments on English to Arabic translation; we tested multiple types of data and reported the performance of the translation using BLEU metric.

Our first set of experiments was to try to find which vocab size works best for our model so we performed BPE on our data in multiple configuration and reported the BLEU score of each. We did this to set a baseline to compare our results.

Language /vocab size	Arabic/ 4k	Arabic/8k	Arabic/16k
English/ 4k	9.52 %	9.69 %	9.58 %
English/8k	7.67 %	7.73 %	6.17 %
English/16k	9.33 %	9.64 %	9.71 %

Table 3.7: the impact of vocab size on the translation performance for BPE algorithm

From what we can see the 4K and the 16K combinations performed well achieving the highest results on average (16K-16K scored 9.71 and 4K-4K scored 9.58), however it seems that when selecting 8K vocab size on the English side the results drop significantly as low as 6.17 in the case of 16K-8K.

In the next set of experiments, we are going to put our technique Parallel alignment segmentation to the test with the best performing configuration of BPE:

For the sake of simplicity when naming the data types, we chose this format [name of the technique (size of the source language EN vocab-- size of target language AR vocab)].

Model name	Data type	BLEU score
m-raw	Raw (non-processed)	5.34
m-bpe-4k4k	BPE(4k-4k)	9.52
m-bpe-16k-16k	BPE(4k-16k)	9.71
m-bpe-4k-8k	BPE(8k-4K)	7.67
m-ABS (our technique)	ABS(4K-4k)	8.09

Table 3.8: performance comparison between our technique and BPE with different configurations (English to Arabic)

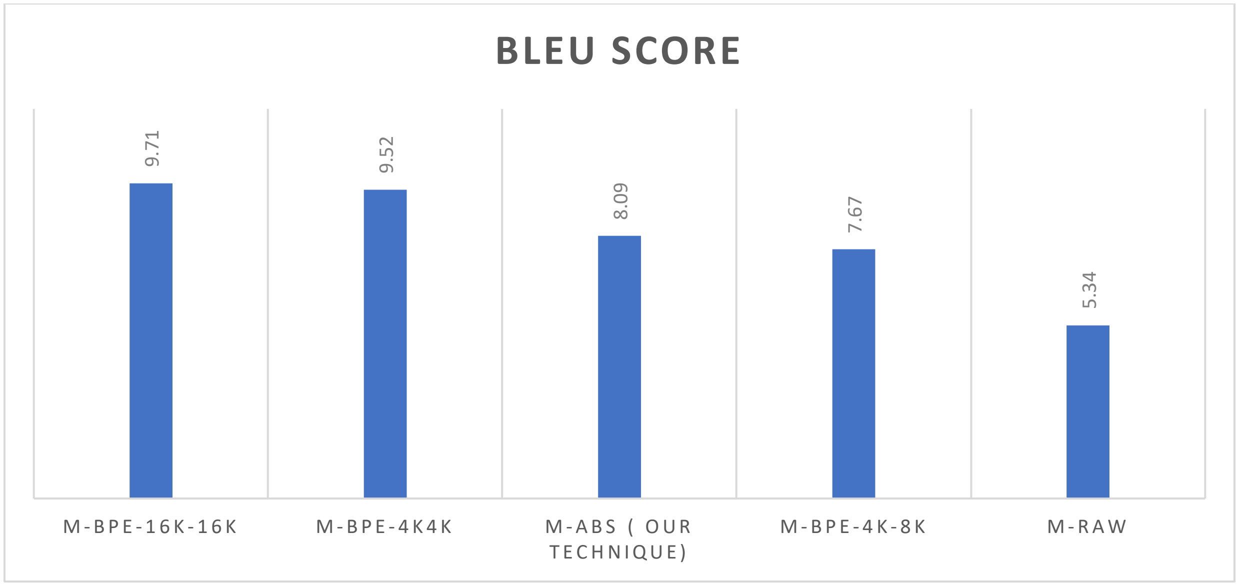


Figure 3.6: Comparative chart (English to Arabic)

As we can see from the result table and the comparative chart our segmentation technique did improve the translation performance compared to the raw data model and some of the BPE configurations, however it fell short of reaching the performance of the best configuration 4K/ 16K of the BPE technique.

BLEU score indicates the overall performance of the model, it is calculated over the entirety of the test data set which includes over 1000 sentence, meaning if a model has a higher score than another that doesn’t mean that it translated all the test examples better.

3.5.3. Arabic to English translation results:

We wanted to test our segmentation technique in both ways to see how it performs so we trained our model to translate from English to Arabic too.

The next set of experiments follows the same structure of the previous one the only difference now is that Arabic is the source language and English is the target one.

[Model name (size of vocab source language AR -- size of vocab in the target language EN)].

Model name	Data type	BLEU score
m-raw	Raw (non-processed)	13.25
m-bpe-4k4k	BPE(4k-4k)	19.62
m-bpe-16k-16k	BPE(4k-16k)	20.97
m-bpe-4k-8k	BPE(8k-4K)	15.31
m-parallel(our technique)	ABS(4k-4k)	18.89

Table 3.9: performance comparison between our technique and BPE with different configurations (English to Arabic)

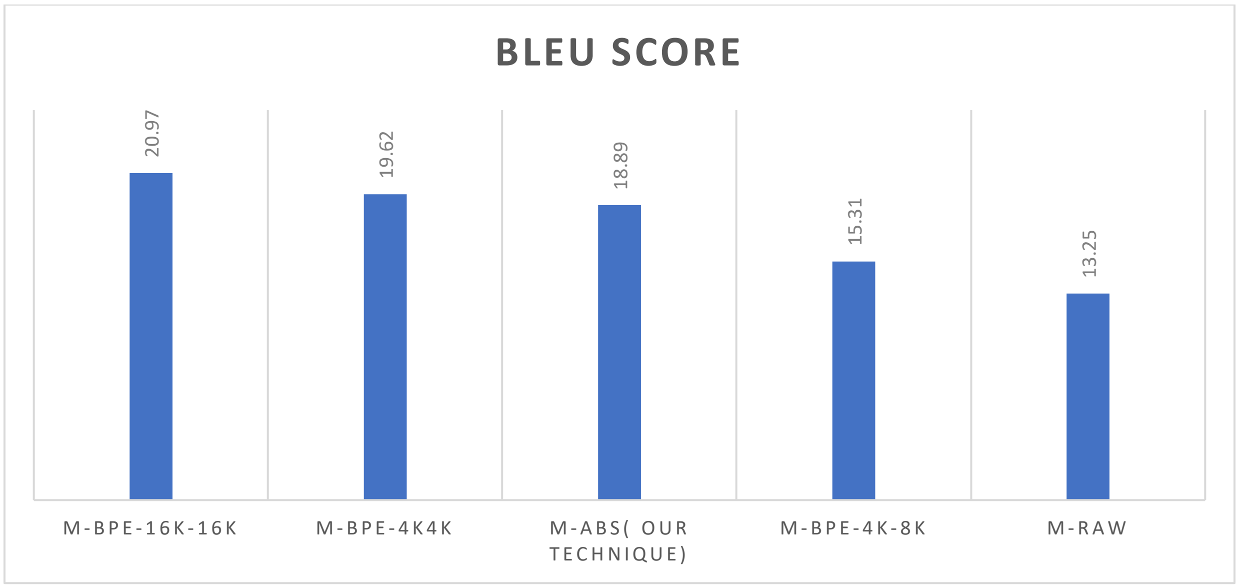


Figure 3.7: Comparative chart (English to Arabic)

As we can see it is much easier to translate from Arabic to English than the way around, the BLEU score of all the models is significantly higher.

We also notice the same pattern as the previous experiment our model scored 18.89 improving upon the raw model and 4k-8k BPE configuration, but again fails to outperform BPE's best configurations.

We assume that the reason our technique did not match the BPE is because in our technique operates only on one side the corpus (Arabic). We think if we can include the English side as well in the process we might be able to improve the performance of ABS, this requires further experiment to confirm.

3.6. Conclusion:

In this chapter, we first introduced our proposed method and detailed all the steps involved in its implementation, our Parallel Alignment segmentation (PAS) technique aims to joint Arabic sequence of characters, which have one English word aligned with consistency.

Then, we detailed our NMT model configuration and how it was implemented, and then we gave information about the data we used in training and testing.

After that, we showed our experiments results, which demonstrates the performance of our technique and puts it in comparison with BPE algorithm, our technique (ABS) showed decent performance, however it fell short when compared to BPE best configurations.

Finally, we gave some assumptions on how we can improve our technique.

Chapter 4 : General conclusion and future works

General conclusion

Achieving human like language processing is what NLP field researchers strive for, developing a system capable of understanding human natural language with all its ambiguity, irregularity, and diversity is no simple feat that we are yet to reach it.

In This work, we explored the task of Machine translation which one of the most important tasks of NLP, we highlighted its mechanism, challenges, and dependencies. There have been many approaches to MT starting from rule-based techniques (RBMT) all the way to recent neural machine translation (NMT) with each having its own limitations and advantages.

What all these techniques have in common and especially NMT systems is their reliance on the quality of the training data, having a large, well-structured data set with rich vocabulary about a specific language would in fact improve that performance of the translation model.

Parallel corporal are commonly used data structure when it comes to NMT, which comes in the form of sentence pairs, a source sentence and its translation. In our case, we chose the pair English-Arabic since Arabic is a good example of a morphological rich language and English is the most used languages online.

In this work, we studied the impact of pre-processing on the performance of Arabic neural machine translation. We introduced our Alignment-based segmentation technique (ABS) and tested its performance in various experiments, we report that our technique improves the translation performance and even outperform BPE in some cases however did not match BPE best performing configuration.

We managed to implement a segmentation technique (ABS) that in fact did improve the performance of the NMT system. However, it is still in need of improvement so as follow up to this work we want to add a proper segmentation process to the English side of the corpus and perform more experiments to fine tune our technique, we also want to test our technique with more sophisticated NMT models such as transformer and GNMT models.

We would also want to point out that more data is needed when it comes to Arabic corporal, having more training data is a necessity if we want to push forward the Arabic neural machine translation.

Bibliography:

[1] Shaalan, K. (2010). Nizar Y. Habash, Introduction to Arabic natural language processing (Synthesis lectures on human language technologies). *Machine Translation*, *24*(3-4), 285–289. https://doi.org/10.1007/s10590-011-9087-8

[2] Guellil, I., Saâdane, H., Azouaou, F., Gueni, B., & Nouvel, D. (2019). Arabic natural language processing: An overview. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2019.02.006

[3] Farghaly, A., & Shaalan, K. (2009). Arabic Natural Language Processing. *ACM Transactions on Asian Language Information Processing*, *8*(4), 1–22. https://doi.org/10.1145/1644879.1644881

[4] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Dean, J. (2016, October 8). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. arXiv.org. https://arxiv.org/abs/1609.08144.

[5] Farghaly, A., & Shaalan, K. (2009). Arabic Natural Language Processing. *ACM Transactions on Asian Language Information Processing*, *8*(4), 1–22. https://doi.org/10.1145/1644879.1644881

[6] Sawalha, M., Atwell, E., & Abushariah, M. A. M. (2013). SALMA: Standard Arabic Language Morphological Analysis. *2013 1st International Conference on Communications, Signal Processing, and Their Applications (ICCSPA)*. https://doi.org/10.1109/iccspa.2013.6487311

[7] Koehn, P. (2020). *Neural machine translation*. Cambridge University Press.

[8] Pak, I., & Teh, P. L. (2017). Text Segmentation Techniques: A Critical Review. *Innovative Computing, Optimization and Its Applications Studies in Computational Intelligence*, 167–181. https://doi.org/10.1007/978-3-319-66984-7_10

[9] Dale, R., Moisl, H., & Somers, H. (2000). *Handbook of natural language processing*. Dekker. Chapter 2: Tokenisation and Sentence Segmentation

[10] Almuhareb, A., Alsanie, W., & Al-Thubaity, A. (2019). Arabic Word Segmentation With Long Short-Term Memory Neural Networks and Word Embedding. *IEEE Access*, *7*, 12879–12887. https://doi.org/10.1109/access.2019.2893460

[11] Khoja, S. (2001). APT: Arabic part-of-speech tagger.

[12] Hadni, M., Ouatik, S. A., Lachkar, A., & Meknassi, M. (2013). Hybrid Part-Of-Speech Tagger for Non-Vocalized Arabic Text. *International Journal on Natural Language Computing*, *2*(6), 1–15. https://doi.org/10.5121/ijnlc.2013.2601

[13] Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., … Roth, R. (2014). MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and

Disambiguation of Arabic. *ACL Anthology, Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, 1094–1101.

[14] Abdul-Mageed, M., Diab, M., & Kübler, S. (2013). ASMA: A System for Automatic Segmentation and Morpho-Syntactic Disambiguation of Modern Standard Arabic. *ACL Anthology*, *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP*, 1–8.

[15] Habash, N., & Rambow, O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05*. https://doi.org/10.3115/1219840.1219911

[16] Zalmout, N., & Habash, N. (2017). Don't Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic. *ACL Anthology*, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 704–713.

[17] Zalmout, N., & Habash, N. (2017). Don't Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic. *ACL Anthology*, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 704–713.

[18] Darwish, K., & Mubarak, H. (2016). Farasa: A New Fast and Accurate Arabic Word Segmenter. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1070–1074.

[19] Abdelali, A., Darwish, K., Durrani, N., & Mubarak, H. (2016). Farasa: A Fast and Furious Segmenter for Arabic. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. https://doi.org/10.18653/v1/n16-3003

[20] Monroe, W., Green, S., & Manning, C. D. (2014). Word Segmentation of Informal Arabic with Domain Adaptation. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. https://doi.org/10.3115/v1/p14-2034

[21] Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/p16-1162

[22] Tsarfaty, R., Nivre, J., & Andersson, E. (2012). Joint Evaluation of Morphological Segmentation and Syntactic Parsing. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 6–10.

[23]Koehn, P. (2020). *Neural machine translation*. Cambridge: Cambridge University Press.

[24] Dyer, C., Chahuneau, V., & Smith, N. A. (2013). A Simple, Fast, and Effective Reparameterization of IBM Model 2. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 644–648.

[25] Khan Jadoon, N., Anwar, W., Bajwa, U.I. *et al.* Statistical machine translation of Indian languages: a survey. *Neural Comput & Applic* **31,** 2455–2467 (2019). https://doi.org/10.1007/s00521-017-3206-2

[26] Peter F. Brown, John Cocke (1990). statiscal approach to machine translation. Computational linguistics

Appendix:

```
def learn_bpe(infile, outfile, num_symbols, min_frequency=2, verbose=False, is_dict=False, tot
al_symbols=False, num_workers=1, no_bow=False,count_cvt=int):
    """Learn num_symbols BPE operations from vocabulary, and write to outfile.
    """

    outfile.write('#version: 0.2\n')

    vocab = get_vocabulary(infile, is_dict, num_workers,count_cvt)
    if no_bow:
        vocab = dict([(word2tuple(x) ,y) for (x,y) in vocab.items()])
    else:
        vocab = dict([(tuple(x[:-1])+(x[-1]+'</w>',) ,y) for (x,y) in vocab.items()])
    sorted_vocab = sorted(vocab.items(), key=lambda x: x[1], reverse=True)

    stats, indices = get_pair_statistics(sorted_vocab)
    big_stats = copy.deepcopy(stats)

    if total_symbols:
        uniq_char_internal = set()
        uniq_char_final = set()
        for word in vocab:
            for char in word[:-1]:
                uniq_char_internal.add(char)
            uniq_char_final.add(word[-1])
        sys.stderr.write('Number of word-
internal characters: {0}\n'.format(len(uniq_char_internal)))
        sys.stderr.write('Number of word-
final characters: {0}\n'.format(len(uniq_char_final)))
        sys.stderr.write('Reducing number of merge operations by {0}\n'.format(len(uniq_char_i
nternal) + len(uniq_char_final)))
        num_symbols -= len(uniq_char_internal) + len(uniq_char_final)

    # threshold is inspired by Zipfian assumption, but should only affect speed
    threshold = max(stats.values()) / 10
    for i in range(num_symbols):
        if stats:
            most_frequent = max(stats, key=lambda x: (stats[x], x))

        # we probably missed the best pair because of pruning; go back to full statistics
        if not stats or (i and stats[most_frequent] < threshold):
            prune_stats(stats, big_stats, threshold)
            stats = copy.deepcopy(big_stats)
            most_frequent = max(stats, key=lambda x: (stats[x], x))
            # threshold is inspired by Zipfian assumption, but should only affect speed
            threshold = stats[most_frequent] * i/(i+10000.0)
            prune_stats(stats, big_stats, threshold)

        if stats[most_frequent] < min_frequency:
            sys.stderr.write('no pair has frequency >= {0}. Stopping\n'.format(min_frequency))
            break

        if verbose:
            sys.stderr.write('pair {0}: {1} {2} -
> {1}{2} (frequency {3})\n'.format(i, most_frequent[0], most_frequent[1], stats[most_frequent]
))
```

```
        outfile.write('{0} {1}\n'.format(*most_frequent))
        changes = replace_pair(most_frequent, sorted_vocab, indices)
        update_pair_statistics(most_frequent, changes, stats, indices)
        stats[most_frequent] = 0
        if not i % 100:
            prune_stats(stats, big_stats, threshold)
```

```
def replace_pair(pair, vocab, indices):
    """Replace all occurrences of a symbol pair ('A', 'B') with a new symbol 'AB'"""
    first, second = pair
    pair_str = ''.join(pair)
    pair_str = pair_str.replace('\\','\\\\')
    changes = []
    pattern = re.compile(r'(?<!\S)' + re.escape(first + ' ' + second) + r'(?!\S)')
    if sys.version_info < (3, 0):
        iterator = indices[pair].iteritems()
    else:
        iterator = indices[pair].items()
    for j, freq in iterator:
        if freq < 1:
            continue
        word, freq = vocab[j]
        new_word = ' '.join(word)
        new_word = pattern.sub(pair_str, new_word)
        new_word = tuple(new_word.split(' '))

        vocab[j] = (new_word, freq)
        changes.append((j, new_word, word, freq))

    return changes
```