**T.R.**
**GEBZE TECHNICAL UNIVERSITY**
**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# QUANTITATIVE WAYS OF MEASURING NATURAL LANGUAGE CHANGE THROUGH TIME AND LOCATION

**MUHAMMED ENES ALMAHDI**
**A THESIS SUBMITTED FOR THE DEGREE OF**
**MASTER OF SCIENCE**
**DEPARTMENT OF COMPUTER ENGINEERING**

**GEBZE**
**2020**

**T.R.**

**GEBZE TECHNICAL UNIVERSITY**

**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# QUANTITATIVE WAYS OF MEASURING NATURAL LANGUAGE CHANGE THROUGH TIME AND LOCATION

**MUHAMMED ENES ALMAHDI**

**A THESIS SUBMITTED FOR THE DEGREE OF**

**MASTER OF SCIENCE**

**DEPARTMENT OF COMPUTER ENGINEERING**

THESIS SUPERVISOR

PROF. DR. YUSUF SINAN AKGÜL

**GEBZE**

**2020**

**T.C.**
**GEBZE TEKNİK ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# DOĞAL DİLLERİN ZAMAN VE KONUMA BAĞLI DEĞİŞİMLERİNİN NİCEL OLARAK ÖLÇÜLMESİ

**MUHAMMED ENES ALMAHDI**

**YÜKSEK LİSANS TEZİ**
**BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI**

TEZ DANIŞMANI
PROF. DR. YUSUF SINAN AKGÜL

**GEBZE**
**2020**

| | |
|---|---|
| GEBZE TEKNİK ÜNİVERSİTESİ | **YÜKSEK LİSANS JÜRİ ONAY FORMU** |

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 03/01/2020 tarih ve 2020/02 sayılı kararıyla oluşturulan jüri tarafından 07/01/2020 tarihinde tez savunma sınavı yapılan Muhammed Enes ALMAHDI'ın tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

**JÜRİ**

ÜYE
(TEZ DANIŞMANI) : Prof. Dr. Yusuf Sinan AKGÜL

ÜYE : Prof. Dr. Banu DİRİ

ÜYE : Dr. Öğr. Üyesi Burcu YILMAZ

**ONAY**

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../..... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

Prof. Dr. Ümit DEMİR
Gebze Teknik Üniversitesi
Fen Bilimleri Enstitüsü Müdürü

# SUMMARY

Over successive generations, languages have evolved, with new languages and dialects branching out; new words emerge, pronunciations vary, and morphology develops.

The process of finding substituted words in a language, as well as knowing how similar languages are, is the cornerstone of studying the development of languages. The challenge in studying Eastern languages occupies in the scarcity of parallel corpora. Current approaches that study the development of languages are either based on parallel corpora or are not of high quality.

The goal of this work is to build an effective system that automatically detects word substitution and inter-language similarity using unsupervised learning, i.e., without parallel corpora. To discover word substitution, we employ an adversarial training procedure to learn how to align between time-based word embeddings spaces and time-independent global word embeddings space with a simple and effective dictionary-based validation method. Furthermore, we estimate the inter-language similarity based on the perplexity of n-gram models that trained on monolingual texts.

We apply our proposed models on Turkic languages and Arabic dialects. We identify word substitutions, in addition to finding the most changed periods during the last 100-years stage of Turkish language development. Moreover, we create fully connected similarity graphs for Turkic languages and Arabic dialects. We visualize the similarities in a heatmap, and we present a map showing the inter-language similarity and the influence of the geographical distribution.

**Keywords: Natural Language Processing, Word Embedding, Word Substitution, Language Similarity, Language Modeling.**

# ÖZET

Birbirini izleyen nesiller boyunca diller gelişmiştir, yeni diller ve lehçelere dallanmıştır; yeni kelimeler ortaya çıkmış, telaffuzlar değişmiş ve morfoloji gelişmiştir.

Bir dilde ikame edilmiş sözcükleri bulmak ve benzer dillerin benzerliklerini bilmek, dillerin gelişimini incelemenin temel taşını oluşturmuştur. Doğu dillerinde paralel korporanın azlığı bu dilleri incelemedeki zorluğu ortaya çıkarmıştır. Dillerin gelişimini inceleyen mevcut yaklaşımlar ya paralel korporaya dayanmıştır ya da yüksek başarım gösterememiştir.

Bu çalışmanın amacı gözetimsiz öğrenmeyle paralel korpora kullanmadan kelime ikamesini ve diller arası benzerliği otomatik olarak tespit eden etkin bir sistem oluşturmaktır. Kelime ikamesini keşfetmek ve zamana dayalı kelime vektör uzayları ile zamandan bağımsız evrensel kelime vektör uzayının nasıl hizalanacağını öğretmek için basit ve etkin sözlük tabanlı doğrulama yöntemi ile çekişmeli eğitim prosedürünü kullandık. Ayrıca, tek dilli metinler üzerinde eğitilmiş n-gram modellere dayanarak diller arası benzerliği kestirdik.

Önerilen modellerimizi Türk dilleri ve Arap lehçelerine uyguladık. Türk dil gelişiminin son 100 yıllık döneminde en çok değişen dönemleri bulmanın yanı sıra sözcük ikamelerini de belirledik. Ayrıca Türk dilleri ve Arap lehçeleri için tam bağlantılı benzerlik çizgeleri oluşturduk. Bir ısı haritasında benzerlikleri görselleştirdik ve bunu diller arası benzerliğe coğrafi dağılımın etkisini gösteren bir harita sunduk.

**Anahtar Kelimeler: Doğal Dil İşleme, Kelime Gömme, Kelime İkame Etme, Dil Benzerliği, Dil Modelleme.**

# ACKNOWLEDGMENTS

I am glad to express my very profound gratitude to my father and mother for providing me with unlimited support and continuous encouragement throughout my years of study.

To whom was the best supporter of my scientific and research journey, and spared no effort in helping me, my dear wife. This accomplishment would not have been possible without you.

I would also like to acknowledge Prof. Dr. Yusuf Sinan AKGÜL as my supervisor of this work, and I am gratefully indebted to him for his valuable assistance on this thesis.

I thank Rima ALMAHDI, Riad ALMAHDI, Abdullah BAKIRCI, and all my friends for their love and support.

# TABLE OF CONTENTS

# LIST OF SYMBOLS AND ABBREVIATIONS

| Symbols and Abbreviations | | Explanations |
|---|---|---|
| BERT | : | Bidirectional Encoder Representations from Transformers |
| CRF | : | Conditional Random Fields |
| CBOW | : | Continuous Bag of Words |
| CNN | : | Convolutional Neural Network |
| DAN | : | Deep Averaging Network |
| DTW | : | Dynamic Time Warping |
| ELMo | : | Embeddings from Language Models |
| GloVe | : | Global Vectors for Word Representation |
| HMM | : | Hidden Markov Model |
| KNN | : | K-Nearest Neighbors |
| LSTM | : | Long Short-Term Memory |
| MLP | : | Multi-Layer Perceptron |
| NLP | : | Natural Language Processing |
| RNN | : | Recurrent Neural Network |
| SVM | : | Support Vector Machine |
| TBMM | : | Türkiye Büyük Millet Meclisi (Turkish Grand National Assembly) |
| USE | : | Universal Sentence Encoder |

# LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

Languages remain one of the greatest inventions of human creativity, whereas primitive ancestors communicated through signs, despite the many differences between the shapes of their letters, their pronunciation, and the appearance date since the appearance of man on the face of the earth [Peterson, 2015].

Languages have evolved over successive generations in a way that makes it difficult to define the boundary between one language and another. So, it is challenging to determine the oldest language and decisiveness in the emergence of one language over the ruins of another language.

For example, the word “selfie” was not present in a few years, but today it is a word that we pronounce and know its meaning without finding linguistic meaning in existing dictionaries, and there are new scientific terms that are invented periodically in various fields, there are diseases, for example, that did not exist, and the names of newly made inventions. Other things emerging in life that directly affect the development of the language, and therefore, the dictionaries are regularly reviewed and developed by adding new words or terms and removing - sometimes - existing words.

Let us go back in time to the ancient era, when the world was less densely populated. People shared one language and one culture, but it was divided into smaller, dispersed tribes in their search for better living conditions. Due to their migration and stability in new places, they became isolated from each other and developed in different ways commensurate with the things around them, such as the climate and animals. Centuries passed while they were eating different food and using new tools that made that common dialect transform into different languages radically continuing to divide with increasing population density. For this reason, linguists try to draw this process by tracing multiple languages to their past as much as they can into their primitive language, and through this process, we note that there are related languages that constitute what is called the language family.

Take Turkic languages as an example. Turkic languages, as a set of languages, divided into 40 different written languages, spoken by 180 million people as a mother tongue and approximately 250 million people as a second language from Eastern Europe to Siberia and west of China [Dybo, 2007].

An important feature that makes Turkic languages different from other language families is that their speakers have lived as nomads for a long time and that these languages are always influenced by each other. In addition to having common words used in the same sense in a large number of Turkic languages, sentence structures always remain the same. Therefore, it is common to see that Turkic languages are not a language family, they are the dialects of a single language, and we can see that they are called Turkic dialects [Tekin, 1978]. Table 1.1 shows how Turkic languages are similar to each other.

Table 1.1: Translation of a sentence in some Turkic languages.

| **Language** | **Translation of "The mother teaches her son reading and writing."** |
|---|---|
| Turkish | Anne oğluna okuma ve yazma öğretir. |
| Azerbaijani | Ana oğluna oxumağı və yazmağı öyrədir. |
| Uzbek | Ona o'g'liga o'qish va yozishni o'rgatadi. |
| Kazakh | Anasy ulyn oqýǵa jáne jazýǵa úıretedi. |
| Kyrgyz | Apam anın uulu okup jana jazuuga üyrötöt. |
| Tatar | Änise ulın uqırğa häm yazarğa öyrätä. |

Turkish language, as one of the Turkic languages, spoken in Southeastern Europe and West Asia, is an additive language belonging to the Turkic languages language family. It forms the continuation of the Ottoman Turkish from the Oghuz language group of the Turkic language family. Turkish is the 22nd most spoken language in the world, with approximately 80 million people speaking [Web 1, 2019].

Turkish has grammatical features such as the affinity and vowel harmony that it shares with many other Turkic languages. In terms of sentence structure, language usually has a subject-object-verb order. Unlike other languages such as German and Arabic, there is no grammatical gender (masculinity, femininity, gender discrimination). Furthermore, Turkish speakers can understand other Oghuz languages such as Azerbaijani, Gagauz, and Turkmen. The Turkish language was written in the Latin alphabet since 1928. The Turkish Language Association controls the spelling

rules in Standard Turkish. The Istanbul dialect, also called Istanbul Turkish, is the standard form of Turkish, and it is the base of the written Turkish language. However, there are various Turkish dialects in Southeast Europe and the Middle East, and these dialects have various silent differences with Istanbul Turkish [Campbell, 2003].

Table 1.2: Pronunciation of a sentence in some Arabic dialects.

| **Arabic dialect** | **Pronunciation of "I only found this library."** |
|---|---|
| Modern Standard Arabic | lam ʾağid siwā hāḏehi-əl-măktba |
| Algerian (Algiers) | ma-lqīt ɣīr hādi-əl-măktaba |
| Egyptian (Cairo) | ma-lʔet-ʃ ella el-maktaba di |
| Gulf (Kuwait) | ma ligēt illa hal l-maktaba |
| Hejazi (Jeddah) | ma ligīt ɣēr hādi al-maktaba |
| Jordanian (Amman) | ma lagēt illa hal ʕal-mektebe |
| Lebanese (Beirut) | ma lʔēt illa hal-i-ʕal-mektebe |
| Mesopotamian (Baghdad) | ma ligēt ɣīr hāði-il-maktaba |
| Moroccan (Casablanca) | ma-lqīt-ʃ mən-ɣīr hādi-lmăktaba |
| Northern Jordanian (Irbid) | ma lagēteʃ illa hal ʕal-mektebe |
| Syrian (Damascus) | ma lʔēt illa hal ʕal-maktebe |

Arabic is another example of the diversity of languages and dialects. Pre-Islamic, Arabic dialects were varied and different in vocabulary, styles, and structures. However, there was a standardized dialect used to write poems, covenants, and covenants. The standardized dialect continued after the emergence of Islam, which is the language in which the Holy Quran was revealed [Kamusella, 2017].

Currently, Arabic has many different colloquial dialects, and colloquial dialects do not have grammatical or morphological rules, dictionaries for their vocabulary and words, or a method for writing them. Some colloquial dialects are closer to classical than other dialects. Table 1.2 shows how Arabic dialects are similar to each other.

As a motivation from classical linguistics, historical linguistics is the scientific study of language change over time [Bynon, 1977]. Principal concerns of historical linguistics include describing and accounting for observed changes in particular languages in addition to reconstructing the prehistory of languages and determining their relatedness, grouping them into language families. Additionally, historical linguistics tries to develop general theories about how and why language changes.

In the world of natural language processing, one of the fundamental problems in eastern languages and dialects processing, Turkic languages and Arabic dialects, for example, is the lack of labeled data, even some of them are almost spoken dialects only. The main difficulty that the supervised method faces in dealing with these languages is that there are few parallel-corpus. Therefore, it is not easy to analyze the relationships between these languages, as is the case in machine translation systems.

Variation in the scripts is also one of the problems that we face when dealing with languages since although the Turkic languages come from the same origin, i.e., some of them are written in Latin script, while the others are written in Cyrillic or Arabic script.

In this work, we focus on overcome previously mentioned challenges. So, we try to answer the following question: “Can we find quantitative ways to measure how similar are two languages? Despite the different scripts, especially those for which we do not have parallel corpora.”. This inquiry leads us to the sub-question: “Can we find the word substitutions without parallel texts?”.

We try to answer the previous questions by studying language change according to two factors; the time and the location factors. The main hypotheses of this work are that the languages are more similar when the places in which these languages are spoken are geographically close, in addition to the ability to detect word substitutions without using any parallel corpus.

To evaluate the hypotheses, we suggest two quantitative methods to measure the temporal and location change in languages by automatically detecting word substitution and language similarity. In both methods, we use mono data only, i.e., we do not need parallel texts.

When studying language change according to the time factor, we have texts divided according to periods. After preprocessing, we train time-dependent word embedding for each period in addition to time-independent global word embedding.

Then we align each period embedding to the global embedding. We find word substitution by the proposed energy function, which uses the global and period similarities to score the substitutions. To validate our results, we suggest a simple but effective dictionary-based method.

When studying language change according to the location factor, we have texts divided according to the region in which this language or dialect is spoken. We start with data preprocessing, including transliteration to Latin script in the case of Turkic languages. Then, we train an n-gram model for each language independently. We use a perplexity-based scoring function to find out how similar two languages by testing language data on another language model. To evaluate the results, we compare the results of the proposed model to the geographic distance of where these languages are spoken.

Through utilizing our work on the Turkish language, we find a list of word substitutions during the last 100 years, without the use of dictionaries. We also defined the word replacement date, as well as the periods in which most replacements occurred. Furthermore, we created matrices of similarities between Turkic languages and Arabic dialects. We also clarified and compared the similarities between languages through a heatmap and through a geographical map showing the distance between cities that speak these languages and the similarity level between them.

We organize this thesis as follows: we show many of the best-known word embedding methods and their applications in section 2. Then, we show alignment, transfer, and similarities, i.e., inter-language techniques in section 3. In section 4, we explain the language modeling concept and some of its approaches. We show the used datasets in section 5. The two proposed models detailed in section 6. Before we end with a summary of the work, we show results and experiences and discuss them in section 7.

# 2. WORD EMBEDDINGS

Word embedding is an approach of distributional semantics that represents words as real number vectors. This representation has useful grouping properties. It groups words that are semantically and syntactically similar. For example, we hope that the words “hoopoes” and “nightingale” are close, but “Istanbul” and “hoopoes” are not close because there is no strong relationship between them.

Therefore, words are represented as vectors of real values. Each value captures a dimension of word meaning. So, semantically similar words should have similar vectors. Simplified, each dimension of the vectors represents a meaning. The numerical value in each dimension captures the closeness of the association of the word to that meaning. Its objective is to quantify and categorize semantic similarities between linguistic elements. This type of representation is dense. Hopefully, synonyms and interchangeable words are nearby in that space.

Vector space models have been used in distributional semantics since the 1990s. Since then, different models have been developed to estimate continuous representations of words; an example is Latent Semantic Analysis (LSA) [Landauer et al., 2013]. The word embedding was initially conceived by [Bengio et al., 2003], who trained these types of vectors in a neuronal probabilistic model. However, [Collobert and Weston, 2008] were possibly the first to demonstrate the power of word embeddings, in which they pointed out word embeddings as a highly useful tool in different NLP tasks. Also, they present a neural network architecture on which many of the current approaches are based. Word embeddings were widely known, thanks to the work of [Mikolov et al., 2013a] who published Word2Vec, a tool to train and use word embeddings. A year later, [Pennington et al., 2014] introduced GloVe, a new tool for the generation of word embeddings. GloVe unlike Word2Vec, is a counting based model.

## 2.1. Word Embedding Applications

As of this moment, word embeddings have become one of the main concepts in natural language processing. Word embeddings capture the meaning of words and translate them into a vector representation that can be used as input for all types of

neural networks. This has caused its use has spread rapidly and is currently a fundamental piece in the architecture of all types of models that perform NLP tasks.

Some of the most relevant applications are shown in the following subsections. These are just some of the applications of word embeddings. Since they are capable of encoding the meaning of words and the relationships between them, it is possible to apply them to all kinds of tasks.

### 2.1.1. Sentiment Analysis

With the growth of the popularity of social networks, it is very interesting to develop a system capable of, for example, analyzing whether the opinions of a product are positive or negative.

A wide variety of technologies has been used to assess sentiment analysis tasks. In the latter years, machine learning techniques proved to be very effective; in particular, the systems based on deep learning techniques represent the state of the art. Some modern systems make use of CNNs where the input is sequences of words represented as word embeddings. An example of such systems is [Dos Santos and Gatti, 2014]. In this field, word embeddings have been widely used as a way of representing words in sentiment analysis tasks, and proved to be very effective [Petrolito and Dell'Orletta, 2018].

### 2.1.2. Text Generation

Given a language model, we can generate text by an iterative process; we select a word based on the sequence so far, add this word to the sequence, and repeat. Therefore, we just need to know how to pick the next word.

Word embeddings are useful in capturing semantic meanings of words. Pre-trained embeddings can improve the accuracy of neural language models [Verwimp and Bellegarda, 2019]. Through the use of RNN-based language models, for example, it is possible to generate text automatically. By combining these models with convolutional networks, it is even possible to create systems that annotate or describe images.

Sampling, greedy and beam search are some of text generation strategies. In sampling, we sample from the conditional word probability distribution. While in the

greedy strategy, we always pick the word with the highest probability. The greedy approach does not always result in the final sequence with the highest overall probability. A beam search keeps track of several probable variants at each step to avoid being led astray by local maxima.

### 2.1.3. Translation Systems

Generally, these systems are formed by a neural network that acts as an encoder and a neural network that acts as a decoder [Cho et al., 2014]. Both the input and output of these neural networks are sequences of words, and word embeddings represent these words. One of the most famous examples of these types of systems is Google Translate; this translator makes use of the seq2seq model [Britz et al., 2017]. These systems make use of parallel corpus, that is, identical texts in different languages. However, the development of models that do not need to use parallel corpus is receiving significant attention. Some examples of such models are [Lample et al., 2017] or [Artetxe et al., 2017].

### 2.1.4. Chatbox, or Question Answering Systems

These systems are gaining increasing popularity. Some examples are Google Assistant that can be found in a large number of smartphones or Amazon Alexa. The seq2seq [Britz et al., 2017] model, in addition to text translations, can also act as a chatbox if small adjustments are made and trained for it.

## 2.2. Methods of Generating Embeddings

Since Word2Vec [Mikolov et al., 2013a] word embeddings began to become popular, a lot of different methods have emerged to generate word embeddings. In this section, we have chosen the best-known word embeddings, in addition to some that have been interesting to us, either because of their performance or because they are very different from the rest of the word embeddings generation methods. Pre-calculated vectors usually accompany word embeddings methods.

As previously mentioned, many efforts have been made to obtain a vector representation of the texts, which facilitates the calculation of the distance between

them in the representation space and, thus, the design of models for the treatment of different tasks of NLP. While proposed techniques for obtaining word embeddings constitute important steps in this regard, the need to capture the entire content of a text in the text representation vector is necessary. For this reason, techniques have been proposed to obtain embeddings at a higher level where their entire context is captured [Perone et al., 2018].

### 2.2.1. Word2Vec

Word2Vec [Mikolov et al., 2013a] refers to a group of models for producing Word Embeddings. They are flat, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2Vec takes a huge amount of text as input and creates a few hundred dimensional vector space, with each word in the corpus associated with a corresponding vector in space. Word vectors are positioned in vector space so that words that share similar contexts in the corpus are close to each other in space.

Word2Vec can use one of two architectures to create Word Embeddings: Continuous Bag-of-Words (CBOW) or Skip-Gram. The CBOW model predicts the current word based on the surrounding context words. The order of the context words does not influence the prediction. In skip-gram architecture, the model uses the current word to predict the surrounding window of context words. CBOW models aim to assign a word to a context. On the other hand, Skip-gram is designed to create a context for a word.

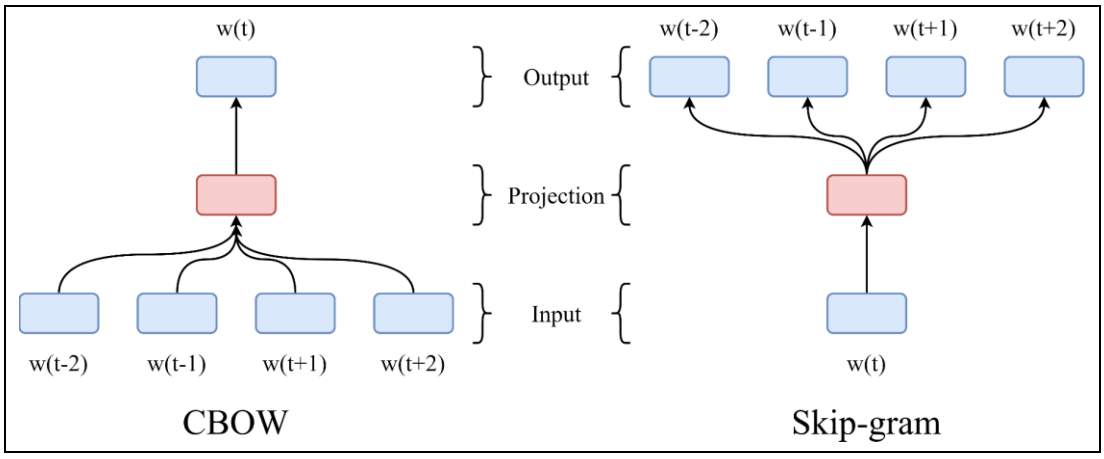



Figure 2.1: CBOW and Skip-gram models.

For example, the following sentence is given: “We went to Istanbul and Konya”. For CBOW, the task results from the input “We went to x and Konya” to predict the word “Istanbul” for x. Conversely, with Skip-gram for the word “Istanbul” the context “We went to x and Konya” must be predicted. The implementation of the Skip-gram and CBOW models is a feed-forward neural network.

In order to understand the learning process, we need some prior knowledge of neural networks. The smallest unit in neural networks is the neuron. A neuron can absorb, modify, and transmit information. Neurons are organized in layers. A network consists of an input layer (encoder) that receives signals from the outside, any number of hidden layers, and an output layer (decoder) that outputs the processed signals again. Weights control the behavior of the neurons towards information. For a network to solve a problem, the weights of the neurons in the hidden layers must take values that give the correct result at the output layer's output, in relation to the signals input in the input layer. The weights of the neurons are randomized, and after each iteration of one packet of information (batch) through the network, the distance of the last layer's output is compared to the target value. This distance (loss) is used to determine through backpropagation which weights need to be changed to get closer to the target value. This change of weights is the actual learning process within a neural network.

In the CBOW model, the input layer receives the input “We went to x and Konya” and should pass the output “Istanbul” in the output layer. The network used for Word2Vec has only one hidden layer. Its weights are thus optimized until the desired result is calculated as accurately as possible for all sentences and target words. The embeddings are generated by extracting the state of the hidden layer neurons for each target word before it is decoded by the output layer.

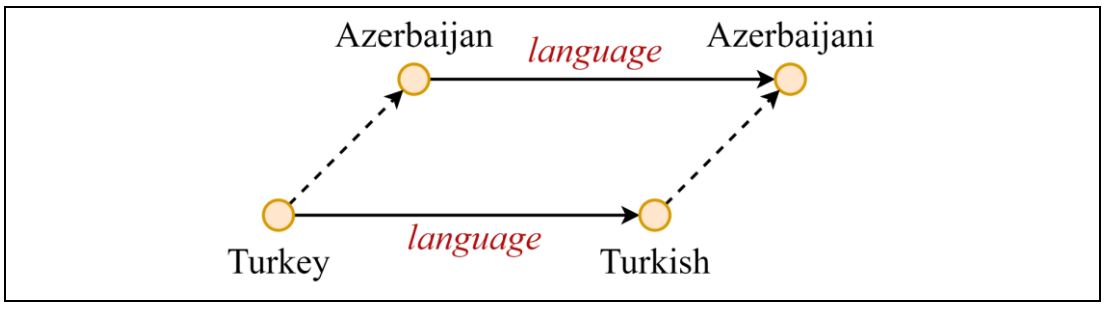



Figure 2.2: Examples of word vectors in 2-dimensional space.

The generated vectors can be used to perform arithmetic operations on semantic relations. The best-known example is “glider” - “bicycle” + “motorcycle”, which leads to the result “airplane”.

### 2.2.2. GloVe

GloVe [Pennington et al., 2014], unlike Word2Vec, is a counting based model. GloVe generates a large matrix where the information of the concurrence between words and contexts is stored. That is, for each word, we count how many times that word appears in some context. The training objective of this matrix is to learn vectors so that the scalar product between the words is equal to the logarithm of the probability of co-occurrence between the words. The number of contexts is very high. Therefore, a factorization of said matrix is performed to obtain one of the smaller dimensions. Thus obtaining a vector that represents each of the words. The advantage of GloVe over Word2Vec is that it is easier to parallelize the training. Therefore, it is possible to use more information during training. Therefore, it is possible to use a higher amount of data during training.

### 2.2.3. FastText

Fixed vocabulary is one of the core problems of using Word2Vec. A word that is not contained in the data with which the Word Embedding was trained cannot be assigned any vectors. Analogously, the representation of a rare word is less certain than a frequent one. This is especially critical for languages where words are heavily inflected or tend to form compounds. Even if the ideal case that every thinkable word should be included in the training corpus occurs, a model that assigns each word its own vector would hardly be processable because of its size. FastText [Bojanowski et al., 2017] addresses these issues by not only calculating representations for words but for its characters. Each word is treated as the sum of its character compositions called n-grams. The vector for a word is composed of the sum of its n-grams. In addition, the word as a whole is always included. In this way, it is expected to obtain better representations for “rare” words, which have very few occurrences in texts corpus, and thus be able to generate vectors for words that are not found in the vocabulary of word embeddings.

### 2.2.4. LexVec

LexVec [Salle et al., 2016] is a model that seeks to obtain better results thanks to the combination of GloVe and Word2Vec. There are different versions since the model has been improved over time. For example, there is a version that uses context vectors [Levy and Goldberg, 2014]. Context vectors seek to improve the performance of word embeddings in analogy tasks. Generally in the models, we only take into account the words found in the context of the main word, for example in the phrase "The little boy left quickly" if the main word is "boy", the context would be formed by "(The, little, left, quickly)". Context vectors also take into account the relative position that words occupy around the target word. For example, in the previous sentence, it will have a context "(The-2, little-1, left+1, quickly+2)". Therefore, we know that the "quickly+2" is two positions to the right of the objective word. We hope that this information will help us obtain better word representations. The latest and most recent version [Alexandre and Aline, 2018] incorporates the same as fastText n-grams.

### 2.2.5. PDC and HDC

PDC and HDC [Sun et al., 2015] are extensions of the CBOW and Skip-gram model, respectively. They seek to capture syntagmatic and paradigmatic relationships at the same time during training. PDC is a model where an objective word is predicted from its surrounding context, in addition to the document in which it appears. The prediction of the word using its context captures the paradigmatic relationship of words since words in similar contexts will tend to have similar representations. This model also causes words that tend to appear in the same document to tend to have similar representations, thus capturing syntagmatic relationships. HDC is similar to PDC, but it applies to the Skip-gram model. In this case, the document is used to predict an objective word, and from that word, its context is predicted.

### 2.2.6. Context2Vec

Context2Vec [Melamud et al., 2016] is an extension of the Word2Vec CBOW model. The main difference between the two models is that CBOW represents the context around a word as the average of the embeddings of the surrounding words.

Context2Vec proposes a more complex approach. The context to the left of the target word and the context to the right of the target word are introduced into independent neuronal models (Recurrent Neural Network LSTM [Hochreiter and Schmidhuber, 1997]). The results are then combined by a new neural network (Multilayer Perceptron). In this way, it is expected to extract the most relevant information from the context of the target word. Another fundamental difference is that while CBOW takes a certain number of words around the target word, for example, two words are taken on the left and two on the right, Context2Vec is able to use the complete phrase where the target word is located.

### 2.2.7. Universal Sentence Encoder (USE)

USE [Cer et al., 2018] is a model proposed by Google to obtain sentence embeddings. Given any text, with this model, a vector is obtained as representation. It has been trained with a variety of sources and tasks, so it can be adjusted to model the semantics of a sequence of words for a large number of purposes. This model has two main versions. The first version is based on the Transformer model [Vaswani et al., 2017], and the second version is based on a DAN model [Iyyer et al., 2015], which takes the average of the word embeddings as input to a feed-forward deep neural network. Although the first version is more computationally expensive, it is designed to generate models with greater precision. On the other hand, the Transformer is a model consisting of an encoder and a decoder. The encoder, which is the part of the model on which USE is based, is made up of a stack of 6 identical layers.

Each layer has a special attention mechanism and a feed-forward neural network, among which there are residual connections followed by normalization. Besides, the model incorporates information about the position of each term in the sequence (positional encodings) to control the order since there is no recurrence in the model.

### 2.2.8. Embeddings from Language Models (ELMo)

ELMo [Peters et al., 2018] distinguishes from word2vec and fastText by directly following the concept of traditional language models. These language models compute a fixed number of consecutive words of a text, the probability of the next word [Seymore et al., 1999]. For the training of embeddings, not only the context before the

target word but also the following context is used. Although the task, i.e., the prediction of a word based on its context, is similar to the CBOW model, it differs in that the multitude of contexts in a word is not used to calculate a fixed vector for each word but the vector of a word in dependency of its current context. Therefore, ELMo is a context-sensitive embedding.

Given a segment of $N$ tokens $(t_1, t_2, .. t_N)$, a language model calculates the probability for each token $k$ based on the previous tokens $(t_1, t_2, .. t_{k-1})$. Conversely, a returned language model calculates the probability based on $(t_{k+1}, t_{k+2}, ..)$. In order to make the technical implementation of this concept understandable, a brief digression into the functioning of LSTMs is needed.

#### 2.2.8.1. Long Short-Term Memory (LSTM)

LSTMs [Hochreiter and Schmidhuber, 1997] are used in Recurrent Neural Networks and allow the network architecture to obtain information about past iterations. Feed-forward networks always adjust their weights based on the currently processed batch, without the possibility that previously processed signals can influence the treatment of the current training data. Therefore, they are not suitable for the prediction of dependent sequences, such as evolution over time. Actually, an LSTM layer is not a layer, but its network consisting of four neural layers. These are divided into three sigmoid and one $tanh$ layer.

In the LSTM cell, the first sigmoid layer is the forget gate. This regulates how much and, above all, which information from the previous LSTM cell should be passed on to the cell state. The next unit, consisting of the second sigmoid and the tanh layer, forms the input gate, which determines what information from the current input will be added. The last sigmoid layer, the output gate, calculates from the input and the cell state which information is passed on to the next layer of the overall network as well as to the next LSTM cell.

The architecture of the ELMo network includes two layers of LSTMs, which in turn are divided into the forward and backward blocks. As already described, this structure is trained according to the concept of language models. The model can be used in other neural networks. The authors suggest to set up only one last layer on the embedding, which filters the information relevant to the task. Tests suggest that the

first LSTM layer contains more information about the grammatical and syntactic properties of speech because its vectors can achieve better results for tasks such as POS tagging than the second layer [Peters et al., 2018]. This is more suitable for tasks that require semantic information, such as disambiguation.

### 2.2.9. Bidirectional Encoder Representations from Transformers (BERT)

Bert Embeddings [Devlin et al., 2018], like ELMo, is one of the context-sensitive embeddings. Bert differs from ELMo in three key areas: tokenization, training of the language model, and network architecture. Bert uses neither a classic 1: 1 relationship between token and word, nor a generic n-gram method like fastText. Instead, the WordPiece tokenization method introduced by [Wu et al., 2016] is used. Tokenization is defined as an optimization problem: Given a number of character n-grams to use; Which must be selected to represent a corpus fully? Bert uses 30,000 pieces. Although the model seems questionable from a linguistic point of view, as it ignores morphological structures, its use, for example in machine translations, leads to better results.

A masked language model does the training of Bert Embeddings. The input for the training consists of segments of 512 token. Of these tokens, 15% are selected for masking, 80% are being replaced by a special masking word, 10% are being replaced by a random word and 10% are being replaced by themselves. This division seems arbitrary at first, but it can be explained by the fact that when the target word is masked 100%, the model does not learn its own representation for non-masked tokens, but only uses them to contextualize the masking. If the remaining 20% is completely replaced by random tokens, the model could not learn anymore, as any customization due to the masked tokens would prove wrong. It is maintaining the target word as an alternative to masking results a context-less prediction, only on token embedding. The model is left in the dark about which token has been replaced, so each token must have its own contextualized representation. It emerges from this task that the LSTMs architecture would be extremely time-consuming, since in this way, for each token in the segment, anticipatory and past information would have to be provided simultaneously. That's why Bert does not use LSTMs, but Transformers.

### 2.2.9.1. Transformer

The Transformer layer introduced by [Vaswani et al., 2017] is based on the concept of attention. Attention solves a problem that occurs in recurrent neural networks in connection with far past inputs. LSTMs generate their output from the last hidden state and the current input. The long-term memory, that is, the hidden state, must provide all previous information needed for the processing of the current input and without knowing the input in advance. As a result, LSTMs tend to forget about long-lost information because it is impossible to predict if they will be needed. Attention mechanisms accelerate this problem by allowing access to all past hidden states while learning to filter the information in response to the input.

The hidden states of the recurrent layers are passed onto the next higher layer by means of a filter taking into account the input and the available information from the hidden states [Bahdanau et al., 2014]. However, this architecture still includes recurring blocks that are dependent on all their predecessors. Therefore, this architecture is not suitable for parallelization. The Transformer layer offers the possibility to replace these recurring parts completely by attention. It consists of an encoder and a decoder component. Each of these components is subdivided into several layers; in the case of Bert 6 layers are used. The encoder layers consist of a self-attention mechanism and a feed-forward network. The decoder layers have the same structure, supplemented by a further attention mechanism between self-attention and feed-forward network. Before a sequence of words passes the first encoder, it is converted into a vector by an embedding. Then follows the first self-attention layer. Self-attention differs from the attention discussed in the previous paragraph in that it does not focus on whether a word is relevant to understanding a sentence or any other task. Instead, it is determined which words of the sentence are relevant in relation to the currently processed word.

This information is passed along with the embedding vector to the feed-forward layer. This then creates a new representation and passes it to the next encoding block. In [Vaswani et al., 2017], in addition to self-attention, multi-head attention is also used. This form of attention divides the vector space of the embedding into subspaces and then determines self-attention in each of these subspaces. In this way, a transformer can recognize and process structures and aspects of language, such as dependency for which parsers are otherwise used [Goldberg, 2019]. In addition to the language model,

a prediction of the next segment is trained. In this case, the network receives an additional segment, which is 50% randomly selected from the corpus. Thus, the recognition of semantic similarity is learned over a large context. To use Bert Embeddings as a feature, each sequence of tokens is placed in the previously trained network. The tokens are then represented by the attention values of each transformer and its attention heads.

# 3. INTER-LANGUAGE TECHNIQUES

## 3.1. Alignment

Machine translation models provide a framework for modeling the mapping between languages at the finest level: that of words, or even sub-phrased units. Unlike machine translation, alignment is a probably ill-defined task, which makes it more difficult. Calculating alignments in bilingual texts has various applications: lexicon extraction [Emmanuel and Daille, 2012], cross-language information retrieval [Nie, 2010], automatic language documentation [Anastasopoulos and Chiang, 2017], [Adda et al., 2016], and [Godard et al., 2016], language learning, etc. Alignment applies to collections of translated texts (i.e., aligned corpora) and seeks to match in both languages textual units of lesser grain than the text: paragraph, sentence sequence, and phrase sequence, without linguistic characterization [Véronis, 2013].

Alignment is the key practical issue of learning different word embeddings for different time periods. Specifically, most cost functions for training are invariant to rotations, the learned embeddings across time may not be placed in the same latent space. [Hamilton et al., 2016] imposes the transformation to be orthogonal, and solves a d-dimensional Procrustes problem between every two adjacent time slices. Since word embedding models are non-convex, training them twice on the same data will lead to different results. Thus, embedding vectors at successive times can only be approximately related to each other, and only if the embedding dimension is large.

Alignment work exploits other types of corpora than aligned corpora, including comparable corpora [Zweigenbaum and Habert, 2006] and multimodal corpora as audio and its transcript [Robert-Ribes and Mukhtar, 1997] or a text image and its transcription [Toselli et al., 2011]. For comparable corpora, aligned segments are most often words, simple terms, and complex terms. Alignment from comparable corpora was mainly focused on simple words in the general language domain [Fung, 1998], [Gaussier et al., 2004], [Mikolov et al., 2013b], and [Rapp, 1999], and on simple terms [Chiao and Zweigenbaum, 2002] and [Morin et al., 2007], and Complexes [Emmanuel and Daille, 2012] in a specialty domain. The most recent work in the field is part of the trend of neural network-based approaches [Fung, 1998], [Jakubina and Langlais, 2017], and [Hazem and Morin, 2017] for simple words and terms.

Bilingual lexicons extracted from comparable corpora represent valuable data because they allow access to the original vocabulary of a specialized or technical domain without any bias induced by a translation mechanism. In particular, these lexicons are of interest to professional translators during the revision stage of a text to be translated, especially when the terms are illustrated by contexts allowing them to understand their uses [Delpech, 2014].

One of the challenges ahead is terms alignment of different lengths (for example, the alignment of a simple term with a complex term) from comparable corpora. Concomitantly, the abundance of available resources (corpus as lexicons) is another challenge in selecting the most relevant data to add to existing models.

Another challenge is to align multimodal corpora (e.g., word source and text target). Pioneering work in this area has been done and seems particularly interesting to develop [Duong et al., 2016] and [Anastasopoulos and Chiang, 2017]. The first attempts at the direct alignment between source and target language text are the work of [Godard et al., 2016]. The authors of [Antonios et al., 2016] also propose to use Dynamic Time Warping (DTW) and IBM translation models together to align source speech and target text.

## 3.2. Transfer

Most of the natural language processing systems are based on models trained on large corpus for a given target language. Recently, transfer or projection approaches have appeared. The common goal of these approaches is to find and explore mechanisms that are not expensive to exploit annotated linguistic resources already available for certain languages and parallel or comparable corpora to produce new annotated resources for other weaker languages. Therefore, the transfer consists of identifying morpho-syntactic equivalences [Yarowsky, 2001], [Wisniewski et al., 2014], and [Zennaki et al., 2016], syntactical [Hwa et al., 2005], [Tiedemann, 2014], and [Aufrant et al., 2016] or semantics [Padó and Lapata, 2009] and [Jabaian et al., 2012] from a corpus of parallel or comparable texts. Such approaches seem particularly interesting for building efficient systems in low-resource scenarios where the amount of training data for a given language or group of languages is limited.

More recently, with the ramping up of the end-to-end learning model, we have come closer to build truly multilingual systems (one system for all languages). For example, neural network approaches (notably encoder-decoders) make it easy to model several languages in a single system, provided that a multilingual representation of inputs and outputs (e.g., based on characters, or sub-lexical units [Sennrich et al., 2015]). Different approaches are currently considered. They share the goal of pooling parts of the neural model so that multiple languages enrich the model and make it more robust [He et al., 2016], [Johnson et al., 2017], and [Gu et al., 2018]. Therefore, one of the main challenges is to define a common representation space for all languages. This space could be likened to an interlingua, making it possible to obtain abstract representations independent of the language.

## 3.3. Similarities

The transfer methods mentioned previously are based on representations of words or sentences in a multilingual space learned from parallel corpora [Mikolov et al., 2013b] from dictionaries or without any real resource available in advance [Conneau et al., 2017] and [Alexis and Kiela, 2018]. These methods have largely developed in recent years, thanks in particular to the massive deployment of neuronal models. This interest is explained in particular by the ability of these models to learn the data representation in a completely generic way, which opens the field to the development of different techniques of pairing at the word level or even document. Such work also finds applications in other areas such as translingual plagiarism detection [Ferrero et al., 2017] and multilingual information retrieval [Balikas et al., 2018]. The transfer between languages is favored by the proximity between languages and is simpler when it comes to languages of the same linguistic family or even variants of the same stemming language.

# 4. LANGUAGE MODELING

In general, when we model a natural language, we try to capture, describe, and exploit the regularities and structures intrinsic to this language. At its simplest, a language model can consist of a list of words and sentences allowed by the language. In the context of language modeling, the sequential aspect is present at different levels of granularity (sequence of characters, words, etc.).

In the case of such sequential data, each event depends, in most cases, on events that precede it. Some approaches are more appropriate than others to exploit this type of relationship.

Models such as SVMs, KNNs, and Decision Trees have shown their effectiveness in various automatic classification applications. Indeed, these models (called “classical methods” in the literature) learn to assign a label to a new data by exploiting its characteristics that are expressed as a set of values. As a result, classical methods generally take sequential data as a feature vector and often examine each event independently of the others.

Other architectures (such as n-gram models, HMMs and CRFs) are specialized in this type of data thanks to their ability to model sequential dependencies.

In addition, the last few years have witnessed the particular performance of Recurrent Neural Networks (RNNs) in sequential data processing, which have become state-of-the-art approaches in various application fields. Long Short-Term Memory (LSTM) architectures, in particular, based on RNNs, are even more efficient because they minimize the loss of information in the case of long sequences [Hochreiter and Schmidhuber, 1997].

These architectures have had a significant interest in the classification of sequences including word sequences [Sundermeyer et al., 2012], images [Vinyals et al., 2015], etc.

## 4.1. Hidden Markov Model

Like the Naïve Bayes classification algorithm, Hidden Markov Model (HMM) [Eddy, 1996] is a generative classification algorithm; that is, which defines a

probability distribution, for each class, according to the input sequence. HMMs are based on Markov models.

We begin with an introduction of these before presenting the theoretical framework of the HMM. In a Markov model, each observation in a data sequence depends on the previous elements. Consider a system with a set of states $S = \{1, 2, \dots, N\}$. At each discrete time step $t$, the system advances from one state to another according to a set of probabilities of transitions $P$. We denote by $s_t$ the state of the system at a time $t$.

In several application contexts, the prediction of the next state depends only on the current state. This means that the transition probabilities between states do not depend on the entire history of the process.

This framework is referred to as the first-order Markov process. For example, assuming the information on the number of students admitted for the current year is sufficient to predict next year's success rate, then we are not required to take into account the rates in previous years.

According to these properties, the probability of moving to the state $s_k$ is formulated as follows:

$$P(X_{t+1} = s_k | X_1, \dots, X_t) \approx P(X_{t+1} = s_k | X_t) \,. \tag{4.1}$$

The transition matrix between the states is constituted by the cells:

$$a_{ij} = P(X_{t+1} = s_j | X_t = s_i) \,. \tag{4.2}$$

We note here that the sum of the exit probabilities of a state $s_i$ is equal to 1 as formulated by the following constraint:

$$\sum_{j=1}^{N} a_{ij} = 1 \,. \tag{4.3}$$

Hidden Markov models represent an extension of the Markov model, distinguished by better abstraction power. Contrary to Naïve Bayes classification, which admits the independence of events, HMM models deal well with the sequential

dependencies. This generative model represents the probability law $P(x, y)$ according to which the $x$ and $y$ sequences are generated.

It is composed of two main parameters: the transition probabilities $P(y_t|y_{t-1})$, which define the degree of connection between two continuous latent variables of $y$, and the prediction probabilities $P(x|y)$, which define how the observed variables of $x$ are related to those of $y$. HMM admit that each event $x_i$ is generated independently conditionally at $y$. This means that the probability of prediction can be considered as:

$$P(x|y) = \prod_{i=1}^{N} P(x_i|y) \, . \tag{4.4}$$

As regards the learning of the different parameters of the model, two algorithms are commonly used, namely, the Viterbi algorithm [Viterbi, 1967] and the Baum-Welch algorithm [Baum et al., 1970].

## 4.2. Conditional Random Fields

Conditional Random Fields (CRF) is the most commonly used variant in the processing of sequential data. The CRFs are discriminative models that represent the conditional probability law of a sequence $y$ of $T$ variables to estimate knowing a sequence $x$ of $T$ observations. This law is defined as follows:

$$P(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x, t)\right) , \tag{4.5}$$

where $Z(x)$ is a normalization function of the form:

$$Z(x) = \sum_{y' \epsilon Y} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(y'_t, y'_{t-1}, x, t)\right) . \tag{4.6}$$

In the two previous equations, $\{f_k\}_{k=1}^{K}$ is a set of characteristic functions explicitly defined. These are usually boolean functions indicating the presence or

absence of a certain characteristic. Each $f_k$ is associated with a coefficient $\lambda_k$, estimated during the learning phase, which determines the weight of the function, if activated, in the calculation of the probability of a sequence $y$. Finally, $Y$ corresponds to the set of possible sequences of variables to estimate. In order to learn these coefficients, the most used approaches are the gradient and Quasi-Newton methods [Dennis and Moré, 1977].

CRFs have been used in various tasks handling sequential data and, in particular, for sequence tagging. In the field of natural language processing, this approach has been applied, for example, in the context of morpho-syntactic tagging and named entity recognition [Sha and Pereira, 2003], [McCallum and Li, 2003] and [Kudo et al., 2004]. Due to its good performance, this approach is still used in sequential data processing [Wang et al., 2016], [Tran et al., 2017] and [Goldman and Goldberger, 2017].

## 4.3. N-gram Model

Since its appearance in the 1970s [Jelinek F. , 1971], the n-gram model has been considered for decades as one of the state-of-the-art language modeling approaches. We will thus introduce, in what follows, the concept of n-grams through the statistical language modeling.

N-gram is a subsequence of n elements of a sequence that, in our case, will be a word sequence. The most common n-grams are those of size 1 (unigrams), those of size 2 (bigrams), and those of size 3 (trigrams). In the literature, we can find systems that successfully train models higher than trigrams. For example, Google has a large corpus in German, Chinese, Spanish, French, Hebrew, English, and Russian of up to 5 grams, created from 8,116,746 books [Lin et al., 2012]. The Google n-gram corpus is available online and has been widely used in various investigations [Divvala et al., 2014].

However, we can go one step further, and instead of using words as a basic element, we can use characters. It is a very common approach in the literature for some problems. If we select the characters within the limits of the words, we are talking about intra-word n-grams. On the other hand, if we use a sliding window, we will

obtain inter-word n-grams; the characters that form the n-grams can belong to more than one word.

Despite the large size of the corpora usually used to learn n-gram models (as in the case of textual data in language modeling), a significant number of n-grams may not appear there. In this case, the model ends up attributing a zero probability to such events.

Smoothing techniques provide a solution to this problem by ensuring non-zero probabilities to absent events. The basic approach of smoothing consists of subtracting a mass of probability from the relatively frequent observed events and then distributing it to unknown or very infrequent events. Several smoothing methods have emerged such as the Laplace methods [Lidstone, 1920], Good-Turing [Good, 1953] and Kneser-Ney [Kneser and Ney, 1995]. These methods differ in the technique according to which the probability masses are subtracted (discounting) and distributed (back-off).

The Kneser-Ney method is considered the state-of-the-art method and is, therefore, the most widely used method. This method carries out the sampling and distribution of the probability masses, taking into account the lower order distributions (the (n-1)-gram model). The combination of the models of different orders is ensured by an original approach, which consists of the use of the marginal distribution.

Natural Language Processing (NLP) is the main area in which the n-gram models have excelled, particularly in the construction of language models. These models served as bases of linguistic knowledge, learned on sequences of words, for systems of automatic speech recognition [Bahl et al., 1983], automatic translation [Brown et al., 1990], information retrieval [Cavnar and Trenkle, 1994], etc. N-gram models were also used to model sequences of characters or graphemes in spelling tasks [Mays et al., 1991], handwriting recognition [Hull and Srihari, 1982], or language identification [Zissman, 1996].

# 5. DATASETS

## 5.1. Turkish Grand National Assembly Dataset

The first dataset we use in this work contains the minutes of the Turkish Grand National Assembly (Türkiye Büyük Millet Meclisi, TBMM) sessions. In parliaments around the world, parliamentary sessions reflect events in society.

In Turkey, the Parliamentary Council was elected every five years from 1920 to 2007, and elections were held every four years after 2007 [Onur Gungor and Sönmez, 2018].

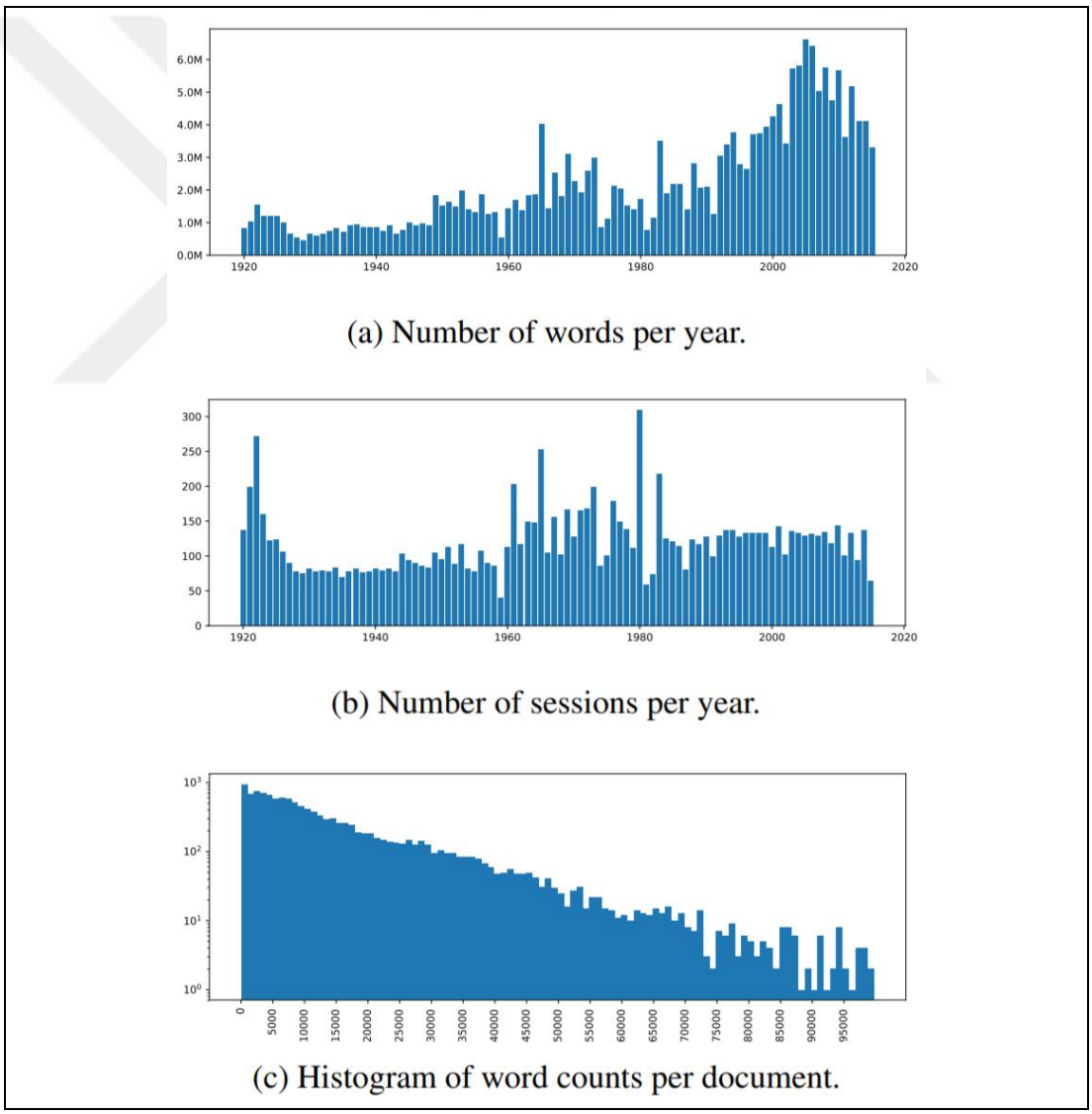



Figure 5.1: Figures that summarize several statistics about TBMM corpus. [Onur Gungor and Sönmez, 2018].

Table 5.1: TBMM corpus statistics. Statistics about periods with a 5-year window split in TBMM corpus from 1920 to 1959 and from 1985 to 2014.

| Period Range | No. of Words | Average Words Length |
|---|---|---|
| 1920_1921_1922_1923_1924 | 7,870,694 | 6.37 |
| 1925_1926_1927_1928_1929 | 6,256,799 | 6.39 |
| 1930_1931_1932_1933_1934 | 5,731,165 | 6.45 |
| 1935_1936_1937_1938_1939 | 6,462,331 | 6.37 |
| 1940_1941_1942_1943_1944 | 6,219,993 | 6.44 |
| 1945_1946_1947_1948_1949 | 9,534,990 | 6.48 |
| 1950_1951_1952_1953_1954 | 13,749,346 | 6.5 |
| 1955_1956_1957_1958_1959 | 11,388,441 | 6.52 |
| 1985_1986_1987_1988_1989 | 19,964,619 | 6.71 |
| 1990_1991_1992_1993_1994 | 28,569,539 | 6.79 |
| 1995_1996_1997_1998_1999 | 34,109,740 | 6.78 |
| 2000_2001_2002_2003_2004 | 45,327,640 | 6.71 |
| 2005_2006_2007_2008_2009 | 53,371,126 | 6.86 |
| 2010_2011_2012_2013_2014 | 43,980,244 | 6.97 |

The dataset we have collected contains the minutes from 1920 to 2014. During the election period, many sessions are held. Fortunately, minutes of these sessions are publicly available in PDF format [Web 3, 2019]. We scrapped them and converted them to text, and then we sorted them according to the chronology of the sessions. While preparing this work, [Onur Gungor and Sönmez, 2018] crawled and processed these documents and provided some statistics about them.

This dataset is valuable because the language spoken during the sessions reflects the spoken language in that time, and thus represents an essential resource for studying Turkish language change during these 95 years. Figure 5.1 (above) shows statistics about the corpus. Note that the number of words increases as we progress in years.

To facilitate the work on the dataset, we have divided it into periods by a non-overlapping 5-year window, starting from 1920. Due to the unbalanced word count between years, we limited the number of words per session to a maximum of 10,000 words, in order to avoid the domination of one period over another. After applying the limit, the dataset contains about 300 million words, distributed over 13,358 sessions.

Table 5.1 shows the start and end year for each period, as well as the words count and the average number of word characters. We exclude the data of years from 1960 to 1984 from our study, due to the lack of data and the changing nature of the data depending on the events occurring at that time. It is also noticeable that the length of words increased as the years progressed, probably due to the nature of the new inserted and removed words in the language.

## 5.2. Wikipedia Dumps Dataset

Wikipedia is an open-source, multilingual, and online encyclopedia. The content was created through the voluntary collaboration of an editors' community. Fortunately, Wikipedia contains articles in most of the Turkic languages. Wikipedia dump [Web 5, 2019] is a complete copy of Wikipedia content in the XML format. We extracted the articles by the WikiExtractor tool [Web 4, 2019]. Figure 5.2 (below) shows Wikipedia dumps we used and the number of articles extracted from each one. We can see the imbalance between the number of articles in different dumps, so we randomly selected 1000 articles from each of them.

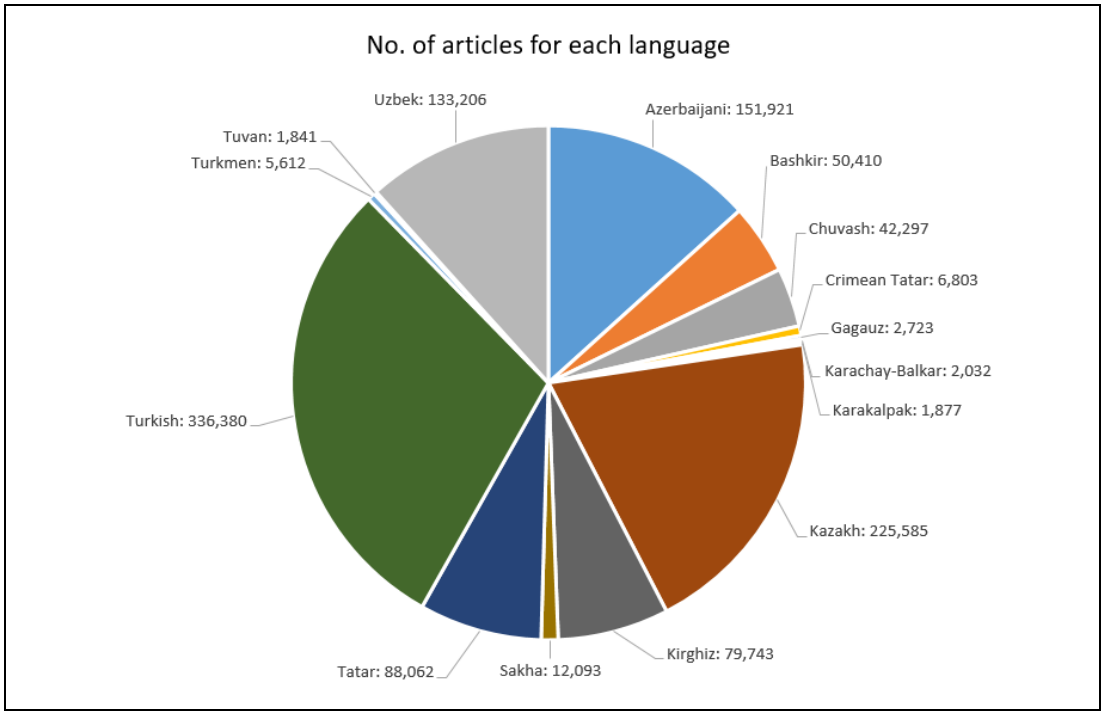



Figure 5.2: No. of articles for each language in the Wikipedia dumps corpus.

Table 5.2: Statistics about the Wikipedia dumps dataset.

| **Language (with local name)** | **Code - Script** | **No. of words** | **Avg. words length** |
|---|---|---|---|
| Azerbaijani (Azərbaycanca) | az - Latin | 49192 | 6.46 |
| Bashkir (башҡортса) | ba - Cyrillic | 36147 | 6.33 |
| Chuvash (Чăвашла) | cv - Latin | 26233 | 6.12 |
| Crimean Tatar (Qırımtatarca) | crh - Latin | 26070 | 6.51 |
| Gagauz (Gagauz) | gag - Latin | 37912 | 6.23 |
| Karachay-Balkar (Къарачай-малкъар) | krc - Cyrillic | 47656 | 5.46 |
| Karakalpak (Qaraqalpaqsha) | kaa - Latin | 53813 | 5.31 |
| Kazakh (Qazaq/Қазақша) | kk – Latin/Cyrillic | 38400 | 6.56 |
| Kyrgyz (Кыргызча) | ky - Cyrillic | 46973 | 6.53 |
| Sakha (Саха тыла) | sah - Cyrillic | 39123 | 6.45 |
| Tatar (Татарча) | tt - Cyrillic | 31108 | 6.05 |
| Turkish (Türkçe) | tr - Latin | 50417 | 6.33 |
| Turkmen (Türkmençe) | tk - Latin | 58076 | 6.51 |
| Tuvan (Тыва дыл) | tyv - Cyrillic | 41894 | 5.98 |
| Uzbek (O‘zbekcha/ўзбекча) | uz -Latin/Cyrillic | 51592 | 5.93 |

The table shows statistics on the data selected from the dumps, the local names of the selected languages, language codes used in Wikipedia sites, and the script in which this language was written.

## 5.3. MADAR Dataset

In the Arab world, there are many spoken dialects. These dialects are mainly used in social media. In [Bouamor et al., 2019], a dataset of 5 Arabic dialects and Modern Standard Arabic (MSA) was created. These dialects belong to diverse cities in the Arab world. Table 5.3 shows statistics about these dialects, cities to which they belong, as well as the used dialect code.

Table 5.3: Statistics about the MADAR dataset.

| City | Country | Dialect Code | No. of words | Avg. words length |
|---|---|---|---|---|
| - | - | msa | 65590 | 4.15 |
| Beirut | Lebanon | bei | 52890 | 4.25 |
| Cairo | Egypt | cai | 58019 | 4.21 |
| Doha | Qatar | doh | 53168 | 4.15 |
| Rabat | Morocco | rab | 60040 | 4.41 |
| Tunis | Tunisia | tun | 54849 | 4.26 |

# 6. METHODOLOGY

To study language change, we should use quantitative methods to measure the changes. In this work, we focused on studying language development in time as well as geographic terms, and we proposed models for studying these changes quantitatively.

We used the TBMM dataset in our proposed model to study the change of the Turkish language during its last development stage. i.e., the last 100 years.

On the other hand, we used the Wikipedia Dump dataset and MADAR dataset in our proposed models to study the change of Turkic languages and Arabic dialects geographically.

In our proposed model, we try to find words that have been replaced during successive time periods during the stage of language development, while we try to find the distance between languages in the case of study changing languages geographically.

Unlike other methods of finding the distance between languages or searching for word substitution over time, our proposed methods do not depend on any parallel data. All we need is monolingual data in the case of searching for the distance between languages or data, which is not necessarily parallel, from every period of time in the case of searching for word substitution.

In the rest of this section, we show the details of the proposed models for the study of language change in quantitative terms.

## 6.1. Time-based Model

In this section, we describe the first proposed model's steps. After splitting the dataset into periods, we train local time-dependent (i.e., for each period data) and global time-independent (i.e., all periods data together) word embeddings. After that, we align each period embeddings space to global space. Then, for an input word, we calculate the energy function, which is an overtime-similarity score for candidate words. Finally, and as a validation metric, we use a simple dictionary-based procedure. We detail the process in the following subsections.

### 6.1.1. Periods' and Global Embeddings

For each period, i.e., time-dependent spaces, we use fastText to train word embeddings. Also, we train a global time-independent word embeddings on all periods' texts together.

In this section, we talk about fastText embeddings. As we deal with morphologically-rich languages like Turkic and Arabic languages, we choose fastText because it takes morphology information into account. This is done by representing the word by the sum of its n-grams.

FastText is derived from the Skip-Gram model. In Skip-Gram, we train word representation to be able to predict the appropriate context word based on its surrounding words. For a given vocabulary of size $M$ and sequence of words $w_1, \dots, w_N$, the objective function is to maximize the following log-likelihood:

$$\sum_{n=1}^{N} \sum_{c \in C_n} \log(P(w_c, w_n)) , \tag{6.1}$$

where $C_n$ is the indices list of context words surrounding the word $w_n$. $P(w_c, w_n)$ is the probability of a context word occurrence $w_c$ given a word $w_n$ calculated as a softmax function:

$$P(w_c, w_n) = \frac{e^{s(w_n, w_c)}}{\sum_{i=1}^{M} e^{s(w_n, i)}} , \tag{6.2}$$

where $s(w_n, w_c)$ is the scalar product between context and word vectors.

$$s(w_n, w_c) = x_{w_n}^{\top} . y_{w_c} . \quad (6.3)$$

#### 6.1.1.1. Subword Information

In the Skip-Gram model, we notice that we use one vector per word. Thus we ignore the internal structure of the word. In fastText, we use a different scoring function, so that we take into account the internal structure of the word.

In order to learn word representation, we create a set of character n-grams, in addition to adding the word itself to the set.

For example, for $n = 3$, the vector for the word "green" is composed of the sum of the n-grams vectors "<gr, gre, ree, een, en>, <green>". The characters "<" and ">" are introduced to mark the beginning and the end of a word and thus to better recognize prefixes and suffixes. In practice, we use $n$ greater or equal to 3 and smaller or equal to 6. We can form various groups by choosing different values for $n$, e.g., taking all prefixed and suffixes.

Given a word $w$, let $G$ is the set of n-grams, the new scoring function is:

$$s(w, c) = \sum_{g \in G} z_g^{\top} . y_c , \quad (6.4)$$

where $z_g$ is the vector representation for the n-gram $g$.

Using this scoring function, n-grams vectors are shared between words. Thus we got a more realistic representation of rare words.

### 6.1.2. Period-Global Alignments

We need to align global and period spaces to ensure that the positions of common words between each of the period spaces and global space as close as possible in order to calculate the similarities over global and period spaces.

In this section, we focus on learning a mapping between global space $G$ and a $i^{th}$ period space $P^i$. We adapt the method described in [Conneau et al., 2017], which uses a domain-adversarial approach for learning a mapping $W$.

Let $P^i = \{p_1^i, \dots, p_n^i\}$ be the set of $n$ word embeddings of period space $i$ and $G = \{g_1, .., g_m\}$ be the set of $m$ word embeddings of global space. The discriminator model has to discriminate between words sampled from $W^iP^i = \{W^ip_1^i, \dots, W^ip_n^i\}$ and $G$. $W^i$ is trained to deceive the discriminator from making the correct decision. The discriminator tries to maximize its capability to recognize the origin of an embedding, and $W^i$ tries to prevent the discriminator from doing its job by making $W^iP^i$ and $G$ as similar as possible. Figure 6.1 shows the network architecture where the source is a period space and the target is the global space.

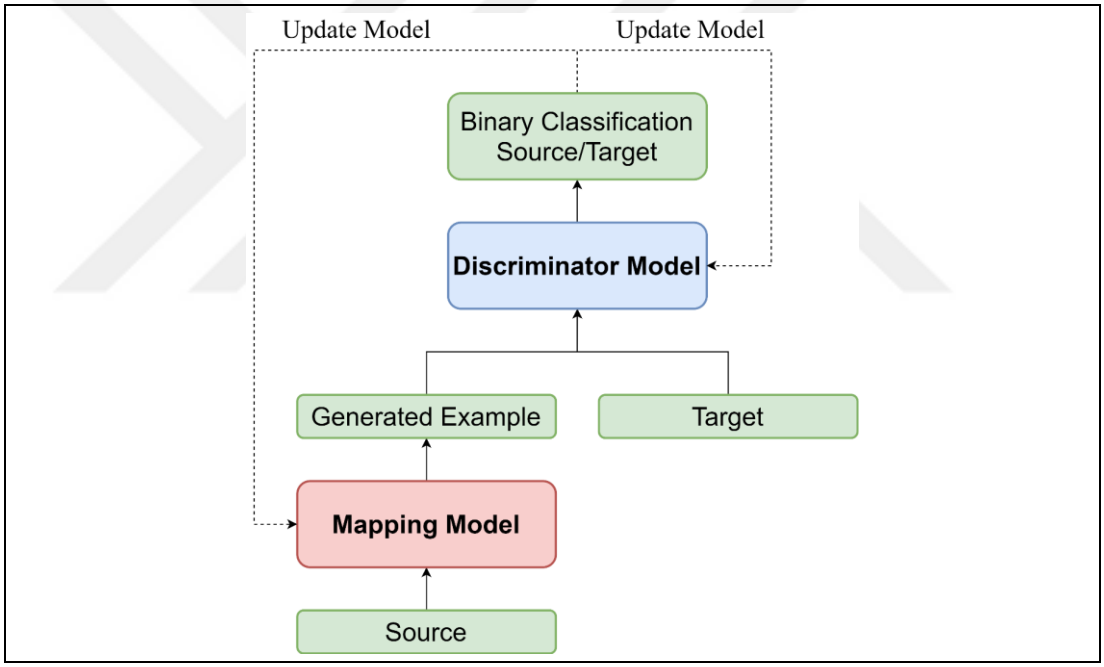



Figure 6.1: Generative adversarial network architecture and training.

### 6.1.2.1. Discriminator Objective

Let $\theta_D$ refers to the discriminator parameters, and according to the discriminator, we consider the probability $Prob_{\theta_D}\ (global = 1|z)$ where $z$ is the mapping vector of the global space (as opposed to period $i$ space). We can write the discriminator loss as:

$$\mathcal{L}(\theta_D|W^i) = -\frac{1}{n}\sum_{j=1}^{n}\log Prob_{\theta_D}(global = 1|G_j) - \frac{1}{m}\sum_{j=1}^{m}\log Prob_{\theta_D}(global = 0|W^i p_j^i)\,. \quad (6.5)$$

### 6.1.2.2. Mapping objective

We train the discriminator and the mapping matrix $W^i$ using the deep adversarial network procedure detailed in [Goodfellow et al., 2014]; by applying stochastic gradient updates to minimize losses, respectively:

$$\mathcal{L}(W^i|\theta_D) = -\frac{1}{n}\sum_{j=1}^{n} log\ Prob_{\theta_D}(global = 0|G_j) - \frac{1}{m}\sum_{j=1}^{m} log\ Prob_{\theta_D}(global = 1|W^i p_j^i)\,. \quad (6.6)$$

### 6.1.2.3. Learning algorithm

We use the training procedure of deep adversarial networks described in [Goodfellow et al., 2014]; the discriminator and the mapping matrix $W^i$ trained successively with stochastic gradient updates to respectively minimize losses.

## 6.1.3. Energy Function

For an input word $w$, we apply an energy function to get the similarity score for the candidate similar words for each period $i$ since the high value means a high probability that the candidate word has replaced this word.

Let $w^i$ be the word vector in period space $P^i$, we obtain $C = \{c_1, \ldots, c_k\}$ as the most $k$ similar words of $w^i$ in $P^i$.

The energy function for the given word $w$ and a candidate similar word $c_j$ in period $i$ is given as:

$$E(w, c_j, i) = \alpha . S^p(w, c_j, i) + (1 - \alpha) . S^g(w, c_j) \quad ; 0 < \alpha < 1 , \tag{6.7}$$

where $S^p$ is a time-dependent similarity function between given two words in a specific period given as:

$$S^p(w, c_j, i) = Cos(e_i(w), e_i(c_j)) , \tag{6.8}$$

where $e_i(w)$ is the lookup table of word embedding in period $i$, and $S^g$ is a time-independent similarity function between given two words given as:

$$S^g(w, c_j) = Cos(e_g(w), e_g(c_j)) , \tag{6.9}$$

where $e_g(w)$ is the lookup table of word embedding in global space.

**Algorithm 1:** Detection of similar words across periods

**Input:** $w$ is an input word.
$e_i$ is the lookup table of word embedding in $i_{th}$ period.
$e_g$ is the lookup table of word embedding in global space.
$k$ is the number of candidate words in each period.
$\alpha$ is a coefficient parameter; $0 < \alpha < 1$
$\theta_E$ is the energy function threshold.
**Output:** *similar_words* is a list of similar words with their scores over periods.

```
similar_words = new list()
for i in periods do
    C = get_most_similar(w, e_i, k) // Get the most k similar words {c_1, .., c_k} in i_th period space.
    for c_j in C do
        // S^p is a time-dependent similarity function between given two words in a specific period
        S^p = Cos(e_i(w), e_i(c_j))
        // S^g is a time-independent similarity function between given two words
        S^g = Cos(e_g(w), e_g(c_j))
        // E is the energy function for the given word w and a candidate similar word c_j in period i
        E_score = αS^p + (1 − α)S^g
        if E_score > θ_E then
            similar_words.append((i, c_j, E_score))
        end
    end
end
```

Figure 6.2: Similar words across periods detection algorithm.

Figure 6.2 shows the algorithm of detecting similar words across periods. We loop over period spaces and find similar local words for a given word. After that, we calculate the energy score for them and keep the words that are above a threshold. We

say about two words that they replace each other if there is only one similar word with an acceptable energy score during the periods.

### 6.1.4. Dictionary-based Validation Metric

To make the validation process easy and automatically done, we propose a simple but effective technique to validate the results. We use Turkish Dictionary of Turkish Language Society (Türk Dil Kurumu) [Web 2, 2019] as a validation dictionary. For an input word $w_i$, the system output a candidate word $w_o$ that replaced the input word. We search for $w_o$ in the description of $w_i$ provided by the dictionary. Using this technique helps in tuning the hyper-parameters while manual verification is time-costly and needs experts in historical terminology.

## 6.2. Location-based Model

In this section, we describe the steps of the second proposed model, which means studying the change of languages geographically. First, we train a language model for each of the languages we have in the dataset. Then, we test the trained language models on the texts of all languages one by one. We measure the distance between two languages by calculating the perplexity when testing language models. Then we create the distance matrix between each pair of languages. We describe the algorithm in Figure 6.3. It takes the languages, the monolingual corpora, and the language model parameter as inputs, and returns a language-pair distance matrix as an output. We use the n-gram models as language models and perplexity value as a distance score between two languages. We detail the process in the following subsections.

**Algorithm 2:** The distance matrix

**Input:** *langs* is the languages list.
*corpora* is the list of monolingual corpus sorted according to *langs* list.
$n$ is the n-gram model parameter
**Output:** *distance_matrix* is the distance matrix between all pairs of languages.

```
distance_matrix = new matrix of size |langs|X|langs|
for i in langs do
    for j in langs do
        // Define a new n-gram language model with n parameter
        model = newNGramModel(n)
        // Train the language model on the first language corpus
        model.fit(corpora[i])
        // Evaluate the language model on the second language corpus using perplexity score
        ppl = model.evaluate(corpora[j])
        // Save the perplexity in the distance matrix
        distance_matrix[i, j] = ppl
    end
end
```

Figure 6.3: The distance matrix creation algorithm.

### 6.2.1. N-gram Model

In a language model, the probability of an event (or word) $w_i$ is determined from the sequence $h_i$ containing the history of all previous events. This probability is expressed by:

$$P(w_i, h_i) = P(w_i|\, w_1^{i-1}) = P(w_i|w_1 w_2 \dots w_{i-1})\,. \tag{6.10}$$

For example, taking $h$ as a sequence of history, the sequence of words “This morning , he took his”. We then want to predict the word following this sequence. Therefore, we must compare the probabilities of occurrence of each category of events (i.e. each word of the lexicon) as in the examples:

$$\begin{matrix} P(car|This\ morning\ ,he\ took\ his)\ , \\ P(jacket|This\ morning\ ,he\ took\ his)\ , \\ P(shower|This\ morning\ ,he\ took\ his)\ . \end{matrix} \qquad (6.11)$$

and choose the one with the highest probability.

The intuition behind the n-gram models is that, instead of calculating the probability of an event knowing its entire history, we can make an approximation that consists in considering only the last few events. The tri-gram model, for example, considers that the probability $P(w_i|w_1^{i-1})$ is equivalent to the probability conditioned on the last 2 events of the sequence.

$$P\left(w_i\middle|w_1^{i-1}\right) \quad \approx \quad P\left(w_i\middle|w_{i-2}^{i-1}\right) = P(w_i|w_{i-2}w_{i-1})\ . \qquad (6.12)$$

In the example of formula (6.11), we get the following approximation:

$$P(car|This\ morning\ ,he\ took\ his) \approx P(car|took\ his)\ . \qquad (6.13)$$

This approximation according to which we consider that the probability of an event depends only on the last 2 events is called the Markov hypothesis. By generalizing the tri-grams, we get the concept of n-grams that look at $n-1$ events in the past. The probability of a new event based on this assumption is therefore estimated as follows:

$$P\left(w_i\middle|w_1^{i-1}\right) \quad \approx \quad P\left(w_i\middle|w_{i-n+1}^{i-1}\right)\ . \qquad (6.14)$$

The most commonly used method for estimating these n-grams probabilities is the Maximum Likelihood Estimation (MLE). This estimate is obtained by counting the number of occurrences, in a learning corpus, of the entire sequence $w_{i-n+1}^{i-1}w_i$ (i.e.,

$w_{i-2}w_{i-1}w_i$ in the case of tri-grams) by normalizing it by the number of occurrences of the history $w_{i-n+1}^{i-1}$ (i.e., $w_{i-2}w_{i-1}$ in the case of tri-grams) in order to obtain a result between 0 and 1. For example, to obtain the tri-gram probability of our example, we are interested in the formula:

$$P(car|took\ his) = \frac{count(took\ his\ car)}{count(took\ his)}. \tag{6.15}$$

In the general case of n-grams, the estimation of the probability of an event by MLE is formulated as follows:

$$P\left(w_i \middle| w_{i-n+1}^{i-1}\right) = \frac{count(w_{i-n+1}^{i-1} w_i)}{count(w_{i-n+1}^{i-1})}. \tag{6.16}$$

### 6.2.2. Language Models Evaluation

The evaluation of the language models is usually done using two main measures; the perplexity and the error rate of the words. Perplexity, closely related to entropy, gives us a clue to the ability of an LM to predict a (or a set of) corpora. The subsections below introduce these essential concepts in a little more detail.

#### 6.2.2.1. Perplexity

Perplexity is the most commonly used means for rapid assessment of language models [Jelinek et al., 1977]. Perplexity, a measure based on cross-entropy, allows us to measure the ability of the language model to predict a given test corpus. On the other hand, the better its prediction ability (a high probability will be attributed), the more the language model is considered good. Moreover, perplexity is often used as an objective function when optimizing language models.

Formally, the perplexity, $PP(T)$, of a language model on a test set is a function of the probability that the language model will assign to the test set. For a test set $T = w_1, \dots, w_N$ of a length $N$:

$$PP(T) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}. \quad (6.17)$$

We can use the chain rule to expand the probability of $T$:

$$PP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \dots w_{i-1})}}. \quad (6.18)$$

Thus, if we are computing the perplexity of $T$ with a trigram language model, we get:

$$PP(T) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-2} w_{i-1})}}. \quad (6.19)$$

An exciting property of perplexity is that by taking its logarithm, we obtain the entropy of the language model, a property familiar to any practitioner of information theory. Moreover, the reduction of entropy was the original goal of the Witten-Bell smoothing, which was intended for text compression.

### 6.2.2.2. Word Error Rate

Word Error Rate (WER) is the application of Levenshtein distance to words: a tool that allows us to measure the similarity between two strings [Klakow and Peters, 2002]. To do this, we try to align the two strings, and we calculate the minimum number of characters (or words in other cases) that must be removed, inserted, or replaced to move from a string to the other. On the other hand, unlike the Levenshtein distance where we summon the errors, WER will count the number of deletions, substitutions, and insertions such as:

$$WER = \frac{S + I + D}{N}. \quad (6.20)$$

where $N$ is the number of words in the reference sentence, $S$ is the number of substitutions in relation to the sentence of reference, $D$ is the number of deletions (omitted words) with respect to the reference sentence, and $I$ is the number of insertions (words added) to the reference sentence.

# 7. EXPERIMENTS

In this section, we review the results that we obtained by applying the proposed models in measuring language change in time and geography. We divided the experiments into two parts; time-based and location-based experiments. We detail them in the rest of this section.

## 7.1. Time-based Experiments

We use the TBMM dataset detailed in Section 5.1. We divided the data set into 5-year non-overlapping time periods starting from 1920 until 2015; each time period contains texts belonging to five years. We use 200-dimensional fastText vectors as global word embedding and for each period word embedding.

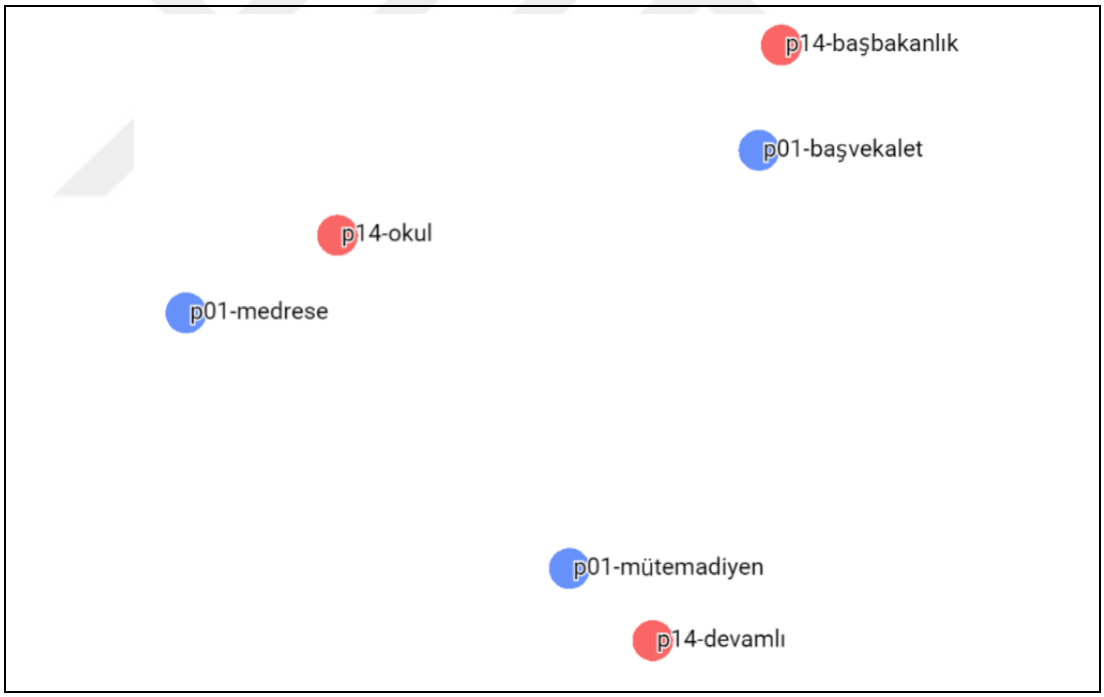



Figure 7.1: Word embedding visualization. Two-dimensional PCA projection of the 200-dimensional fastText vectors that belong to two different periods. Points that appear in blue belong to word vectors from the first period (1920-1924), while red points represent word vectors from the last period (2010-2014). Before we apply PCA to them, we align both word vectors spaces with the global word vectors space. “okul” and “medrese” mean “school”, “başbakanlık” and “başvekalet” mean “premiership”, “devamlı” and “mütemadiyen” mean “continuous”.

During the alignment process, we did not provide any supervised information about word meanings. The mapping $W$ has size 200x200. For the discriminator, we use a multilayer perceptron with two hidden layers of size 2048, and Leaky-ReLU activation functions. The input to the discriminator is corrupted with dropout noise with a rate of 0.1. We use the energy score threshold $\theta_E = 0.4$ and $\alpha = 0.75$ in the energy function shown in formula (6.7) since high $\alpha$ value means giving more importance to the time-dependent similarity.

Figure 7.2, Figure 7.3, and Figure 7.4 provide examples of true predicted words by our system, the usage probabilities of these Turkish words and how they change across periods. Moreover, figures show the relatively high energy scores for these substituted words. The words vary in the period in which they were replaced, and some words were replaced more than once. Interestingly, some words were replaced by another word, and then the original word reused again, then the replacement happened again, as shown in Figure 7.3 and Figure 7.4.

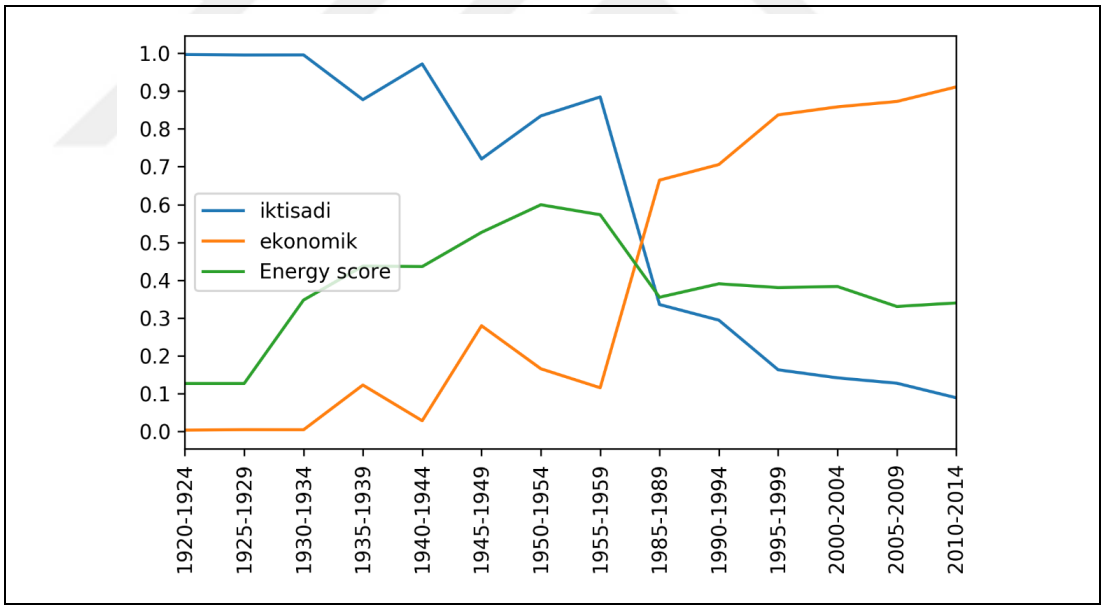



Figure 7.2: “ekonomik” and “iktisadi” probability and energy score chart. The figure shows an example of two substituted Turkish words. It shows their usage probabilities and energy score across periods. Both words mean “economic”.

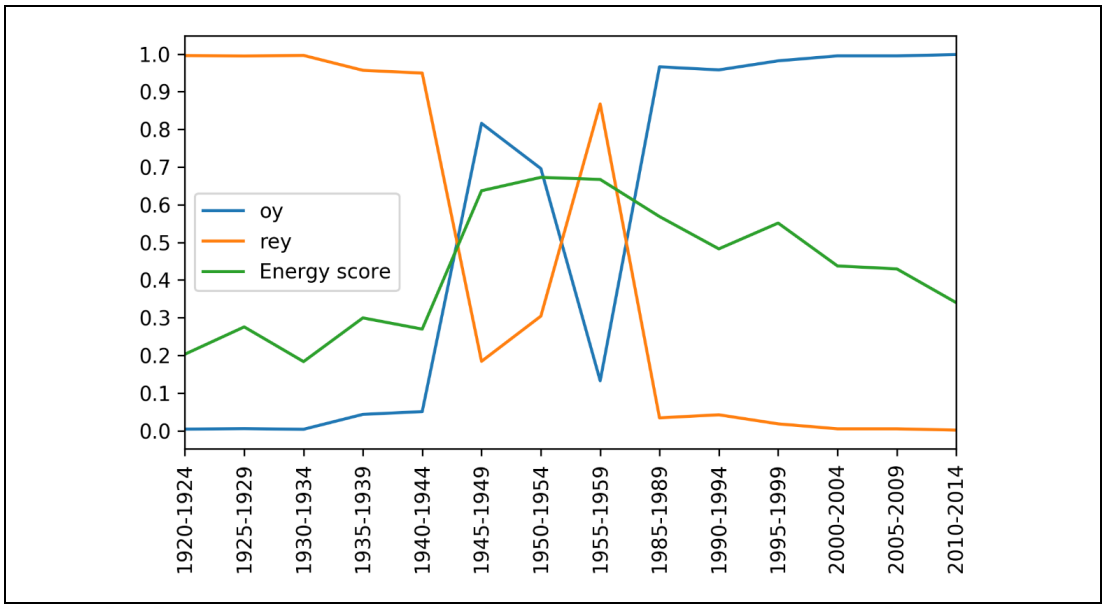



Figure 7.3: “oy” and “rey” probability and energy score chart. The figure shows an example of two substituted Turkish words. It shows their usage probabilities and energy score across periods. Both words mean “vote”.

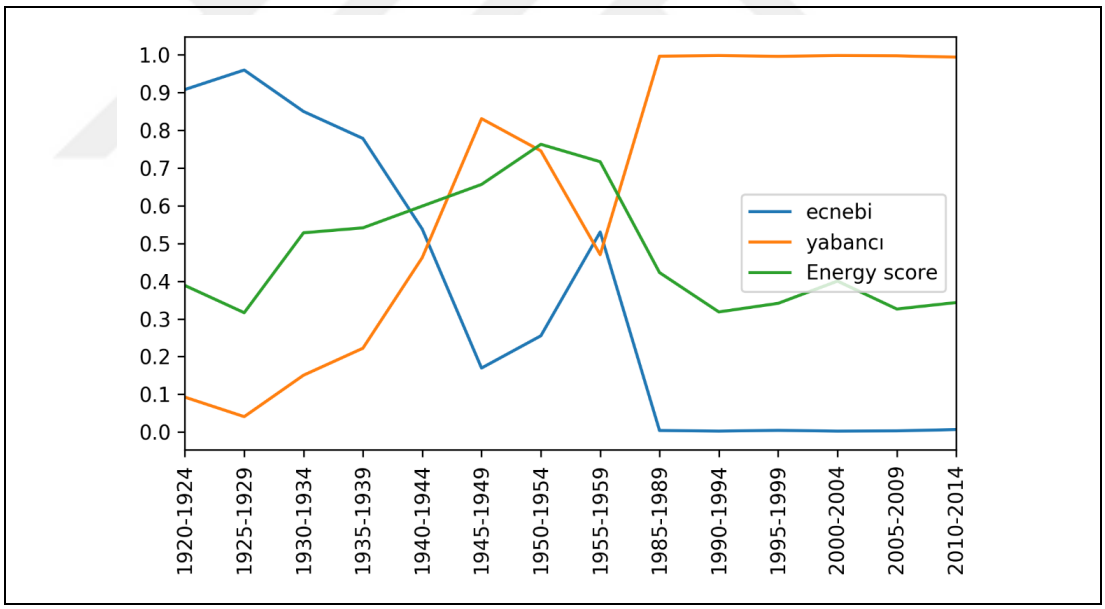



Figure 7.4: “ecnebi” and “yabancı” probability and energy score chart. The figure shows an example of two substituted Turkish words. It shows their usage probabilities and energy score across periods. Both words mean “foreigner”.

Figure 7.5, Figure 7.6, and Figure 7.7 show the usage probabilities and energy scores for non-substituted Turkish words. Although their usage probabilities switch across periods, the energy scores keep at relatively low values.

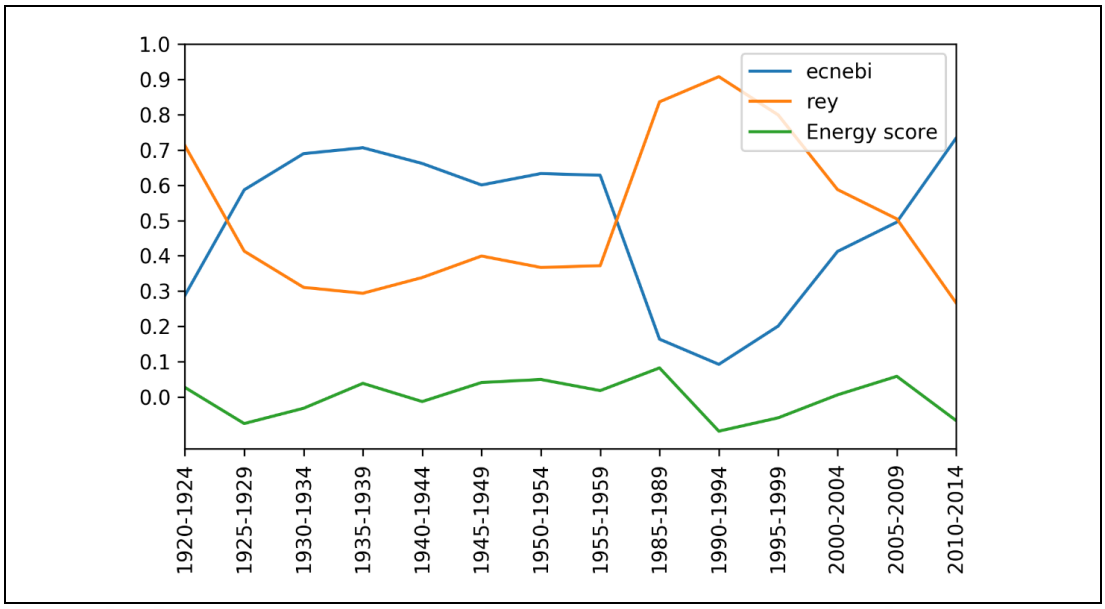



Figure 7.5: “ecnebi” and “rey” probability and energy score chart. The figure shows an example of two non-substituted Turkish words. It shows their usage probabilities and energy score across periods. “ecnebi” means “foreigner” and “rey” means “vote”.

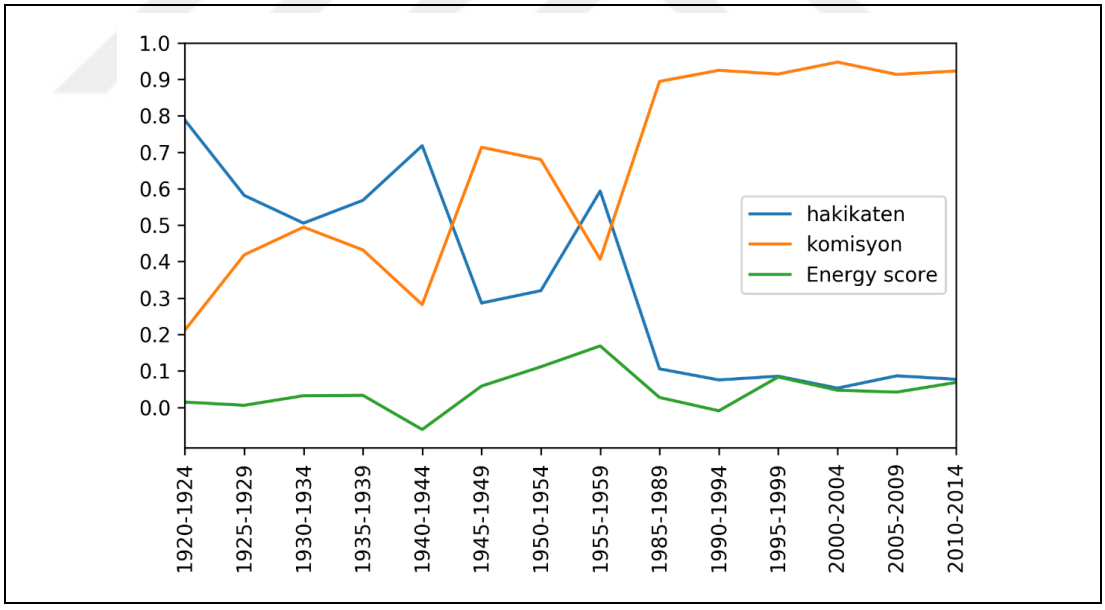



Figure 7.6: “hakikaten” and “komisyon” probability and energy score chart. The figure shows an example of two non-substituted Turkish words. It shows their usage probabilities and energy score across periods. “hakikaten” means “really” and “komisyon” means “committee”.

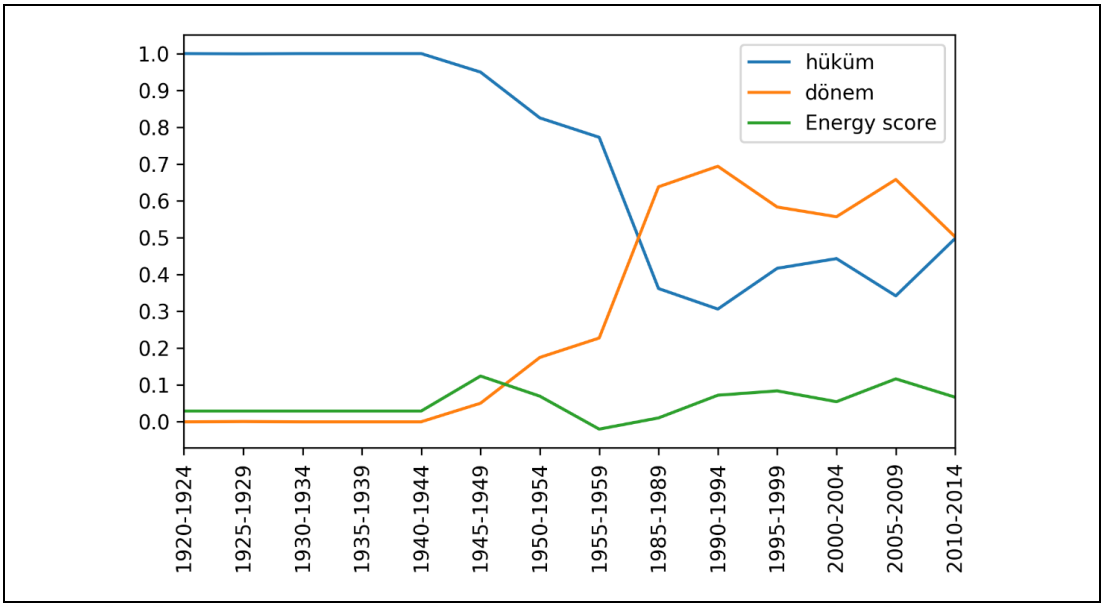



Figure 7.7: “hüküm” and “dönem” probability and energy score chart. The figure shows an example of two non-substituted Turkish words. It shows their usage probabilities and energy score across periods. “hüküm” means “decision” and “dönem” means “period”.

Table 7.1: Examples of randomly selected words with their substitutes. Examples of randomly selected words used in the last period with their meaning from a Turkish dictionary, the best candidate word output from our system, and its maximum energy value across periods.

| **Word** | **Meaning from the dictionary** | **Replaced word** | **Meaning in English** | **Max. Energy Score** |
|---|---|---|---|---|
| adına | . . . , bir şeyin veya bir kimsenin namına, hesabına, yerine, . . . | namına | in the name of | 47 |
| başbakanlık | . . . , başbakanın yaptığı iş, başvekâlet, başbakan ve görevlilerinin çalıştığı daire. | başvekalet | premiership | 41 |
| süre | . . . , zaman aralığı, zaman bölümü, müddet. | müddet | duration | 55 |
| özel | . . . , kişiye ait olan, hususi, resmî karşıtı. | hususi | special | 45 |
| ödenek | . . . , bir iş için ayrılan belli para, tahsisat, parlamento üyelerine, . . . | tahsisat | allowance | 58 |
| ekonomik | . . . , ekonomi ile ilgili olan, iktisadi, az masraflı, hesaplı, . . . | iktisadi | economic | 43 |

We achieve 54.76% accuracy on words with at least 0.01% probability using our dictionary-based validation method.

Table 7.1 (above) shows examples of randomly selected words used in the last period given to our system and the best candidate word to be the most similar word in previous periods. It shows the similarity values based on the energy function and illustrates the interest in using the dictionary-based validation metric. Although we were able to capture many correct word substitutions, there are still some cases that the system was unable to predict correctly. Table 7.2 shows some of these examples. We can classify the wrong predictions into three groups. The first group contains multi-word names of famous characters or events that appeared in a certain period, such as “kemal=halim” and “cemil-koçak”. The second group includes different meaning words used in the same context, such as “kısmen-tamamen” and “cenaze-tedavi”. We can explain this because we generate word vectors based on their context. The third group has misspelled words, especially during the first five periods, perhaps because of the optical character recognition errors, such as the word “kati-katı” and “köy-koy”.

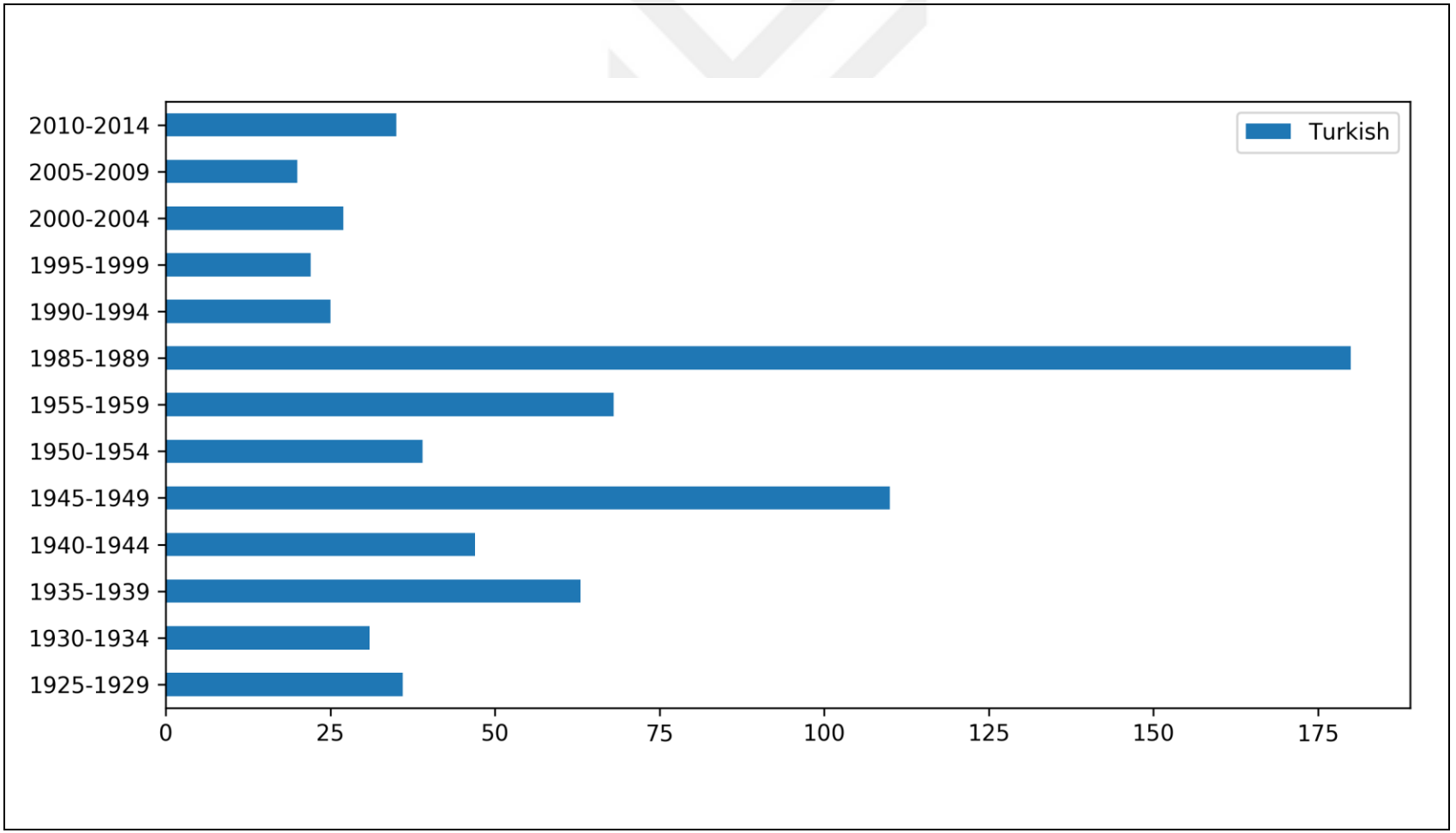
Table 7.2: Examples of false detected word substitutions.

| **Word** | **Word Meaning in English** | **Replaced word** | **Replaced Word Meaning in English** |
|---|---|---|---|
| kemal | person name | halim | person last name |
| cemil | person name | koçak | person last name |
| kısmen | partially | tamamen | completely |
| cenaze | funeral | tedavi | treatment |
| kati | absolute | katı | solid |
| köy | village | koy | bay |

If we go toward more generalization, it leads to the ability to discover the essential changing dates in the vocabulary within the language. Figure 7.8 shows the number of words switches happened in each period for a random set of words substitutions. “namına=adına (in the name of)”, “vazife=görev (job)”, and “tahsisat=ödenek (grant)” are examples of switches in the fifth period.

“iktisadi=ekonomik (economic)”, “başvekâlet=başbakanlık (premiership)”, and “vekalet=bakanlık (ministry)” are examples of switches in the eighth period.

Figure 7.8: Words switches histogram. The histogram shows the count of words switches that happened in each period for a random set of words substitutions.



## 7.2. Location-based Experiments

In this section, we show the results of the proposed model for measuring the distance between languages and dialects geographically. We have made several illustrations to enable us to analyze, compare, and evaluate results. In the rest of this section, we present details of the experiments for both Turkish languages and Arabic dialects.

### 7.2.1. Turkic Languages

We use Wikipedia dumps dataset to measure distances between Turkic languages. As shown in Table 5.2, Turkic languages are written using various script systems, so we first converted Cyrillic characters into Latin using ISO 9:1995 transliteration system, see **Error! Reference source not found.** for details. Then we t rain the n-gram model where $n = 7$. Table7.3 shows the perplexity-based rounded distance values each pair of Turkic languages. Figure 7.11 shows the distances heatmap of these distances.

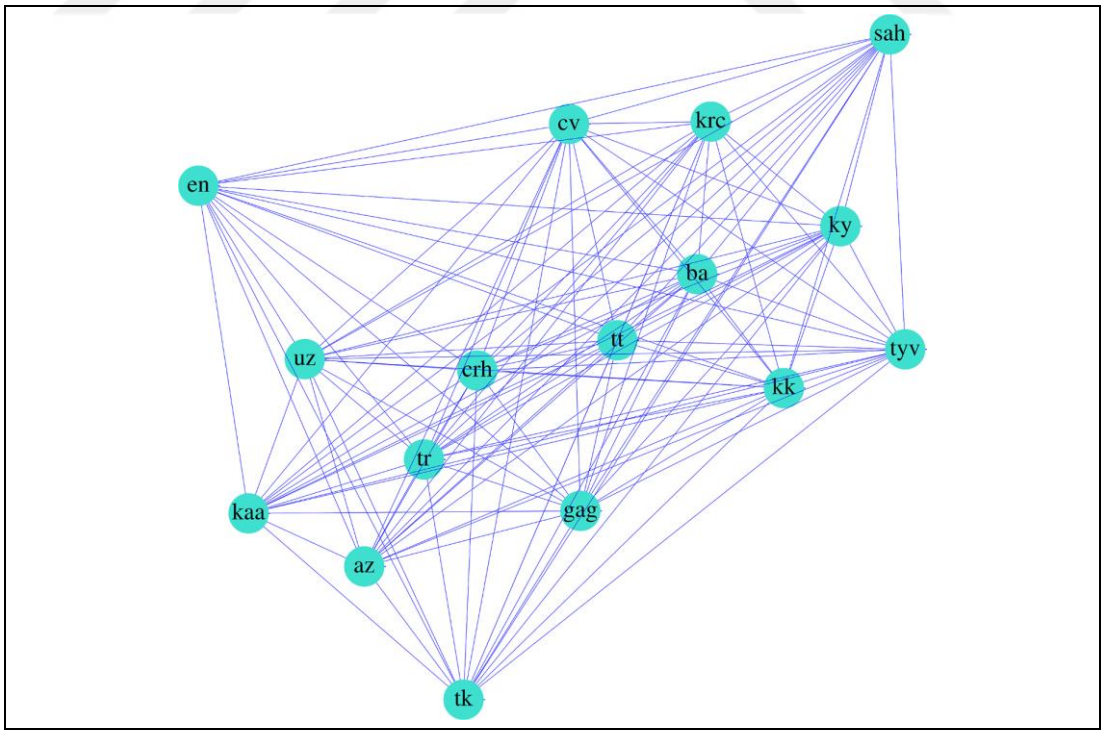



Figure 7.9: Turkic and English languages' distance graph. Edge length represents the distance value between two languages on both sides.

In order to clarify the outputs of the work, we have created a fully connected graph that expresses the distances between languages, where the length of the edge represents the distance between the two languages associated with it on both sides, as shown in Figure 7.9. To draw the graphs, we use NEATO utility following the approach proposed by [Kamada and Kawai, 1989]. NEATO draws a graph by constructing a virtual physical model and running an iterative solver to find a low-energy configuration. An ideal spring is placed between every pair of nodes such that its length is set to the shortest path distance between the endpoints. The springs push the nodes so their geometric distance in the layout approximates their path distance in the graph. This often yields reasonable layouts [Eades, 1984]. The error in drawing the graph increases when it is fully-connected because of the high number of constraints, while the error keeps low when there are few edges only such as the case of the graph in Figure 7.10. We use NetworkX python library as an interface for NEATO tool.

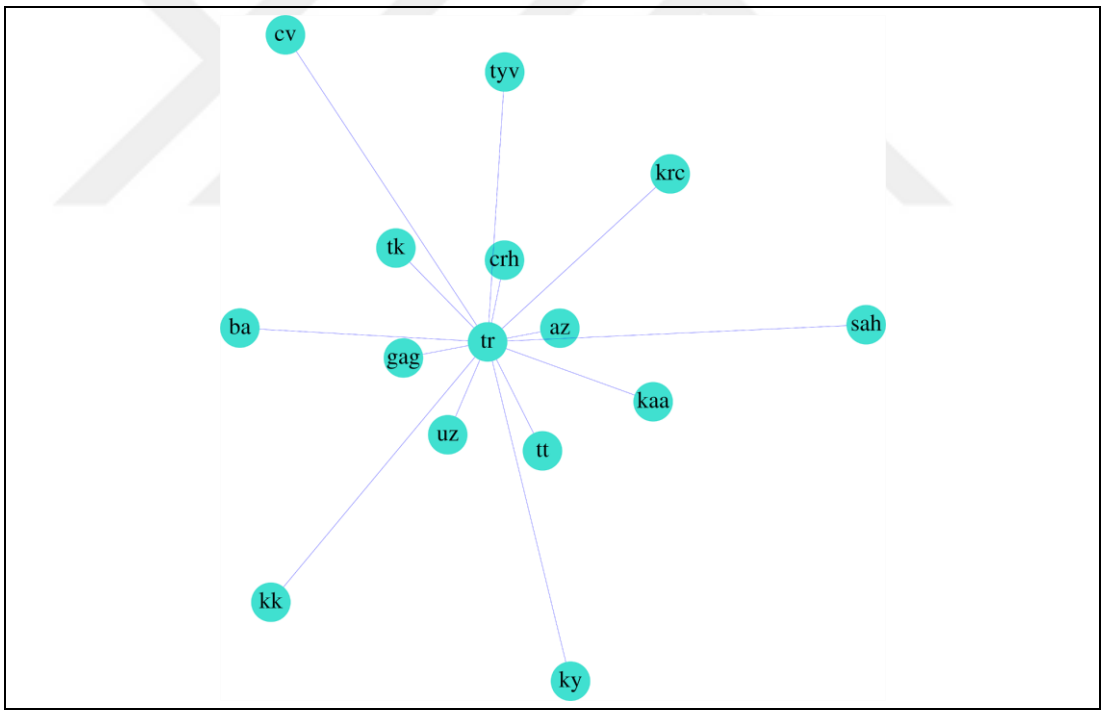



Figure 7.10: Turkic languages' MST distance graph. Edge length represents the distance value between two languages on both sides. We use Kruskal's algorithm [Kruskal, 1956] to generate the minimum spanning tree.

We added English language as a baseline. It is notable that the English, despite its distance from many Turkic languages, is closer to some Turkic languages than

others. We can explain this with the widespread of English, in addition to the closeness of cities where some Turkic languages are spoken to the European continent.

We notice from the graph that Turkish and Tatar languages are almost in the middle of the graph, and therefore it is interesting to conclude that these two languages are the center of the Turkic languages. To further clarify the previous point, we apply the Minimum Spanning Tree (MST) algorithm on this graph and concluded the graph shown in Figure 7.10.

To evaluate our results, we drew a map showing the location of the cities in which Turkic languages are spoken, and thus easy to estimate the geographical distance between these cities, as well as we created links showing the similarity value between languages according to color and thickness. These maps are shown in Figure 7.12, Figure 7.13, Figure 7.14, and Figure 7.15. The distance in kilometers between cities is shown in Table 7.4.

Table7.3: Distances between Turkic and English languages.

| | **az** | **ba** | **crh** | **cv** | **en** | **gag** | **kaa** | **kk** | **krc** | **ky** | **sah** | **tk** | **tr** | **tt** | **tyv** | **uz** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **az** | 16 | 69 | 41 | 126 | 87 | 48 | 51 | 118 | 90 | 136 | 141 | 49 | 40 | 53 | 102 | 32 |
| **ba** | 71 | 15 | 69 | 47 | 208 | 85 | 81 | 40 | 47 | 53 | 64 | 64 | 62 | 28 | 47 | 62 |
| **crh** | 28 | 66 | 14 | 107 | 123 | 29 | 44 | 98 | 68 | 114 | 109 | 51 | 28 | 36 | 78 | 31 |
| **cv** | 61 | 39 | 58 | 14 | 362 | 63 | 76 | 76 | 51 | 48 | 63 | 81 | 58 | 35 | 53 | 48 |
| **en** | 86 | 104 | 69 | 84 | 23 | 72 | 88 | 103 | 117 | 97 | 102 | 76 | 70 | 84 | 105 | 95 |
| **gag** | 28 | 56 | 27 | 94 | 111 | 15 | 47 | 87 | 66 | 92 | 87 | 55 | 27 | 36 | 65 | 37 |
| **kaa** | 64 | 160 | 85 | 311 | 68 | 102 | 15 | 168 | 101 | 236 | 190 | 42 | 53 | 57 | 107 | 72 |
| **Kk** | 67 | 31 | 71 | 58 | 146 | 82 | 94 | 16 | 40 | 29 | 57 | 60 | 61 | 31 | 42 | 62 |
| **krc** | 68 | 40 | 67 | 60 | 87 | 74 | 84 | 50 | 14 | 46 | 58 | 77 | 74 | 35 | 47 | 60 |
| **ky** | 69 | 36 | 66 | 60 | 211 | 74 | 63 | 36 | 38 | 15 | 52 | 86 | 65 | 32 | 38 | 66 |
| **sah** | 78 | 56 | 75 | 73 | 139 | 87 | 107 | 51 | 57 | 74 | 14 | 115 | 78 | 58 | 54 | 80 |
| **tk** | 50 | 160 | 64 | 308 | 83 | 62 | 46 | 146 | 96 | 182 | 133 | 15 | 35 | 59 | 97 | 53 |
| **tr** | 28 | 70 | 30 | 103 | 57 | 31 | 51 | 97 | 70 | 99 | 105 | 42 | 16 | 39 | 76 | 34 |
| **tt** | 62 | 26 | 54 | 50 | 73 | 72 | 74 | 38 | 38 | 39 | 56 | 59 | 54 | 14 | 39 | 52 |
| **tyv** | 78 | 46 | 67 | 58 | 91 | 77 | 117 | 56 | 55 | 40 | 56 | 94 | 62 | 38 | 15 | 81 |
| **uz** | 39 | 70 | 45 | 108 | 81 | 61 | 50 | 120 | 80 | 142 | 122 | 59 | 47 | 48 | 85 | 17 |

Figure 7.11: Heatmap showing similarities between Turkic languages.

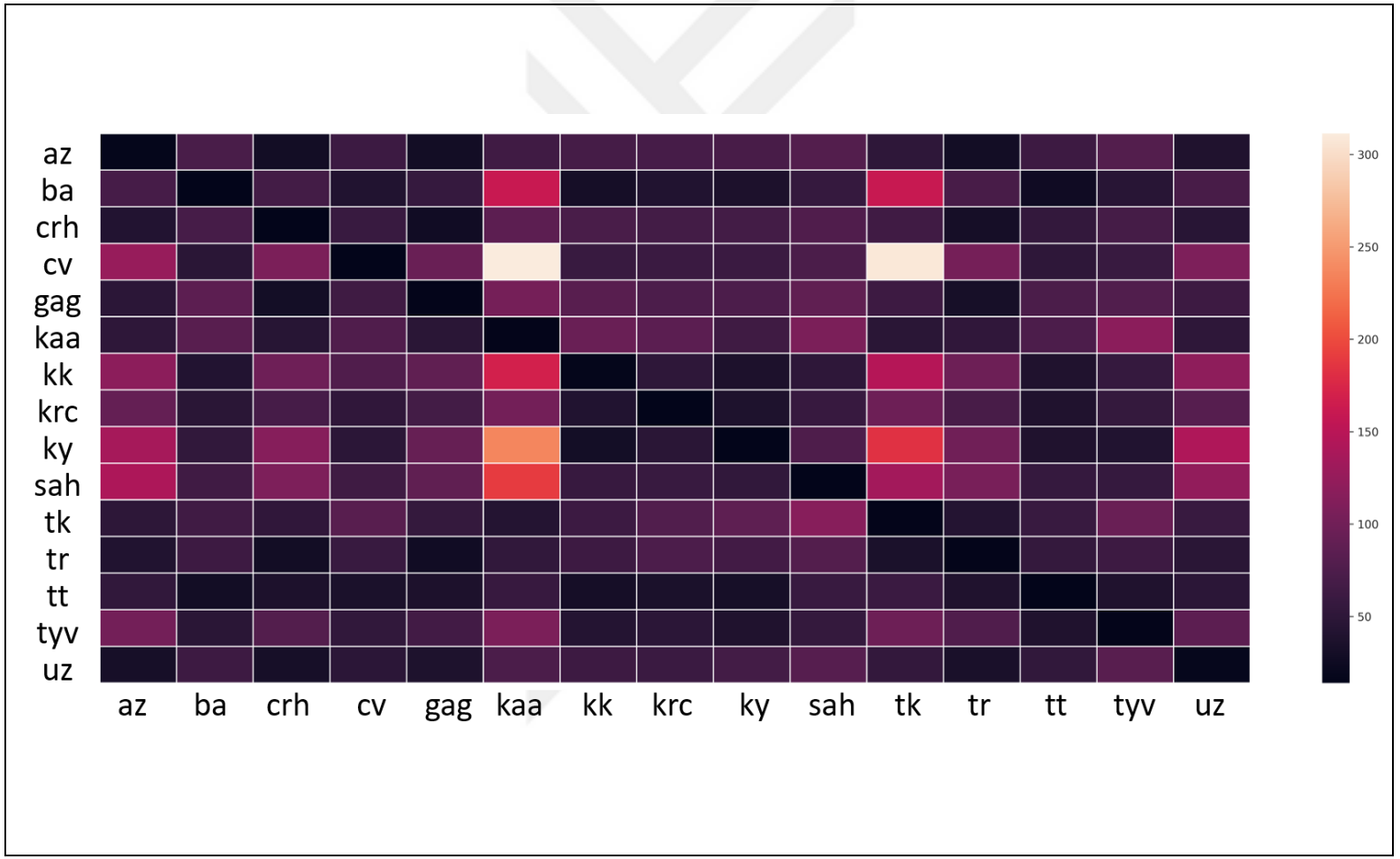


Figure 7.12: Very low similar Turkic language map. Map showing very low similar Turkic language and the geographical distance between cities where the languages are spoken. Yellow links mean less than 25% similarity.

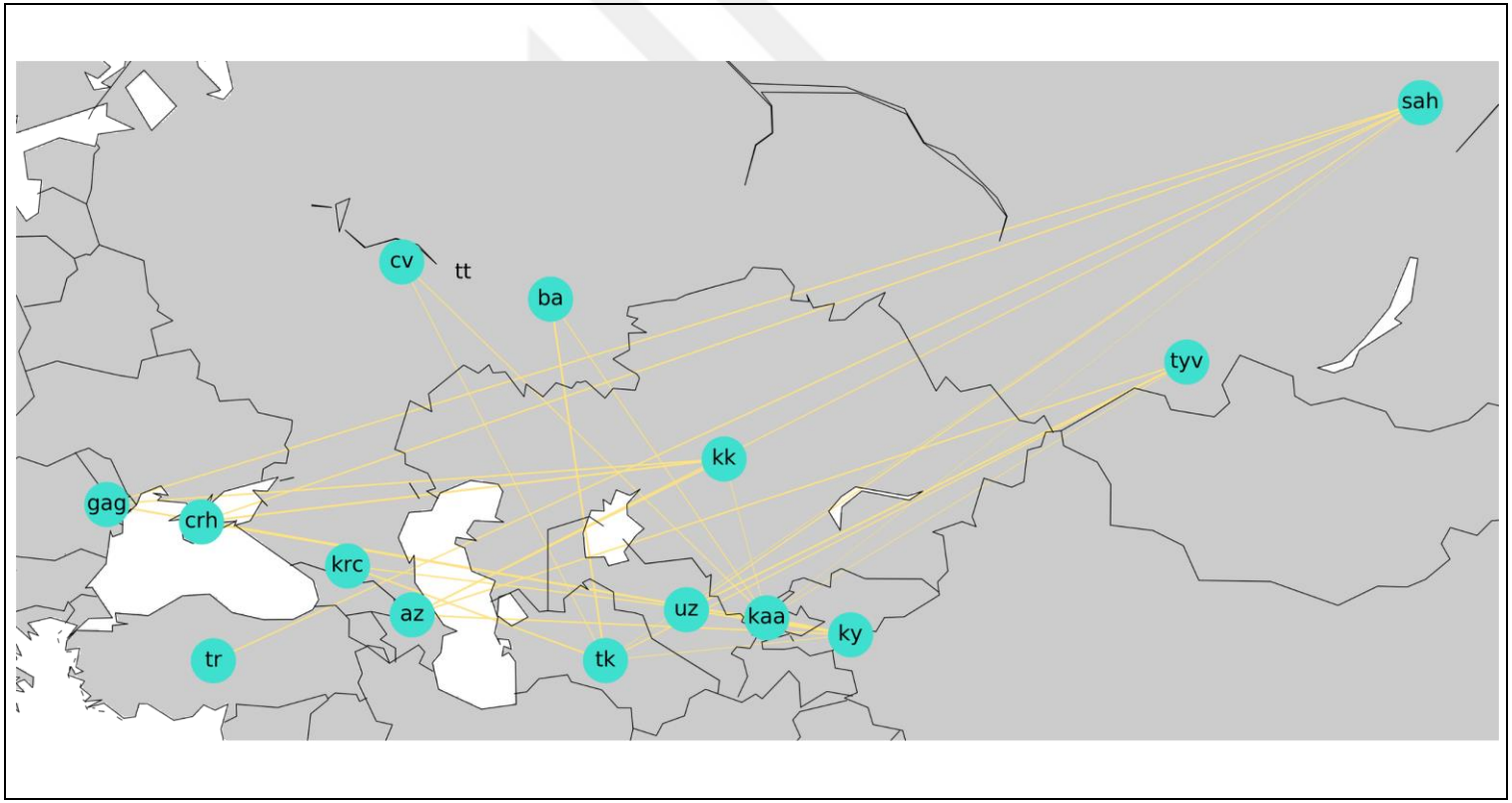


Figure 7.13: Low similar Turkic language map. Map showing low similar Turkic language and the geographical distance between cities where the languages are spoken. Green links mean similarity between 25% and 50%.

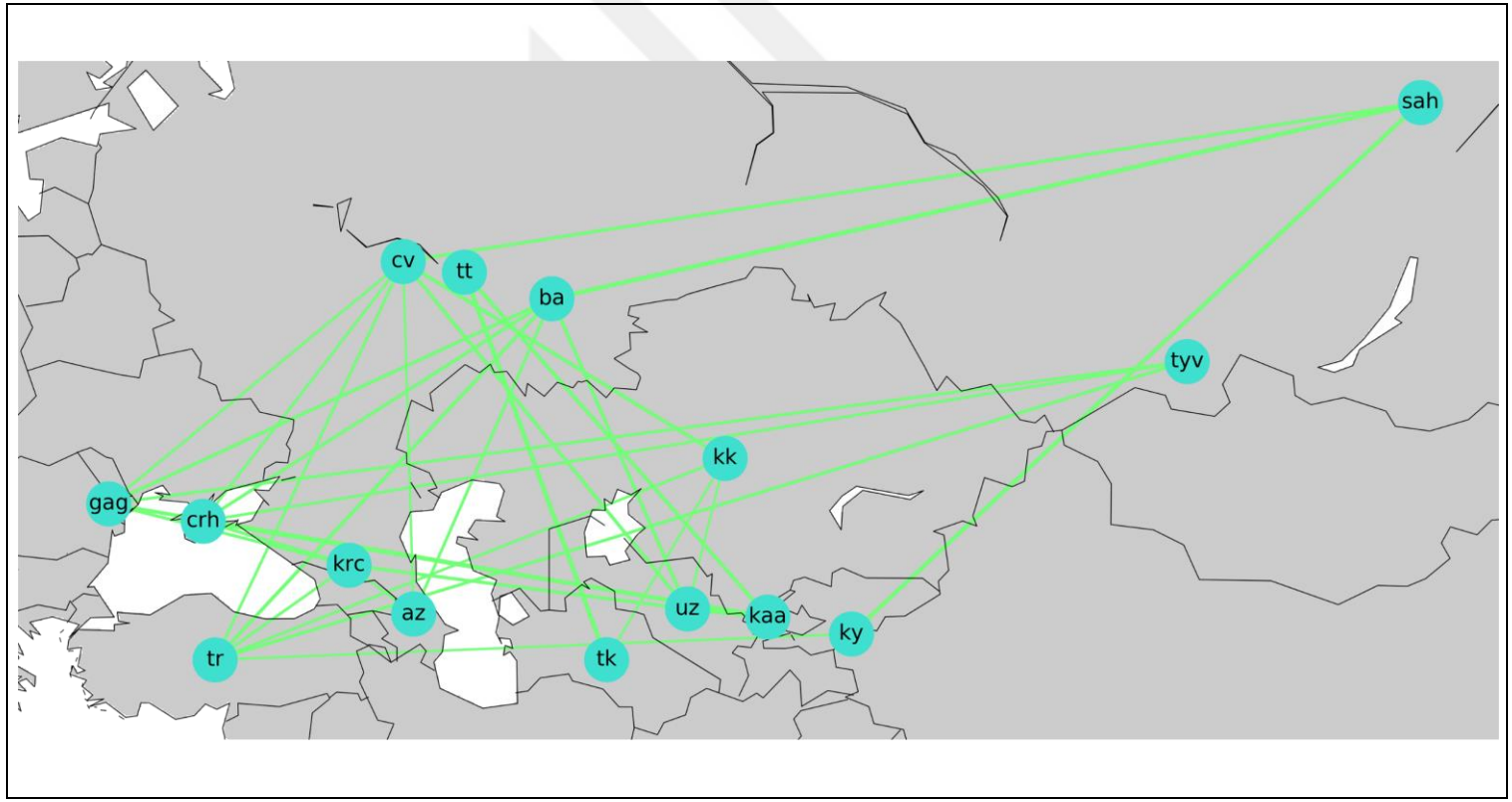


Figure 7.14: High similar Turkic language map. Map showing high similar Turkic language and the geographical distance between cities where the languages are spoken. Blue links mean similarity between 50% and 75%.

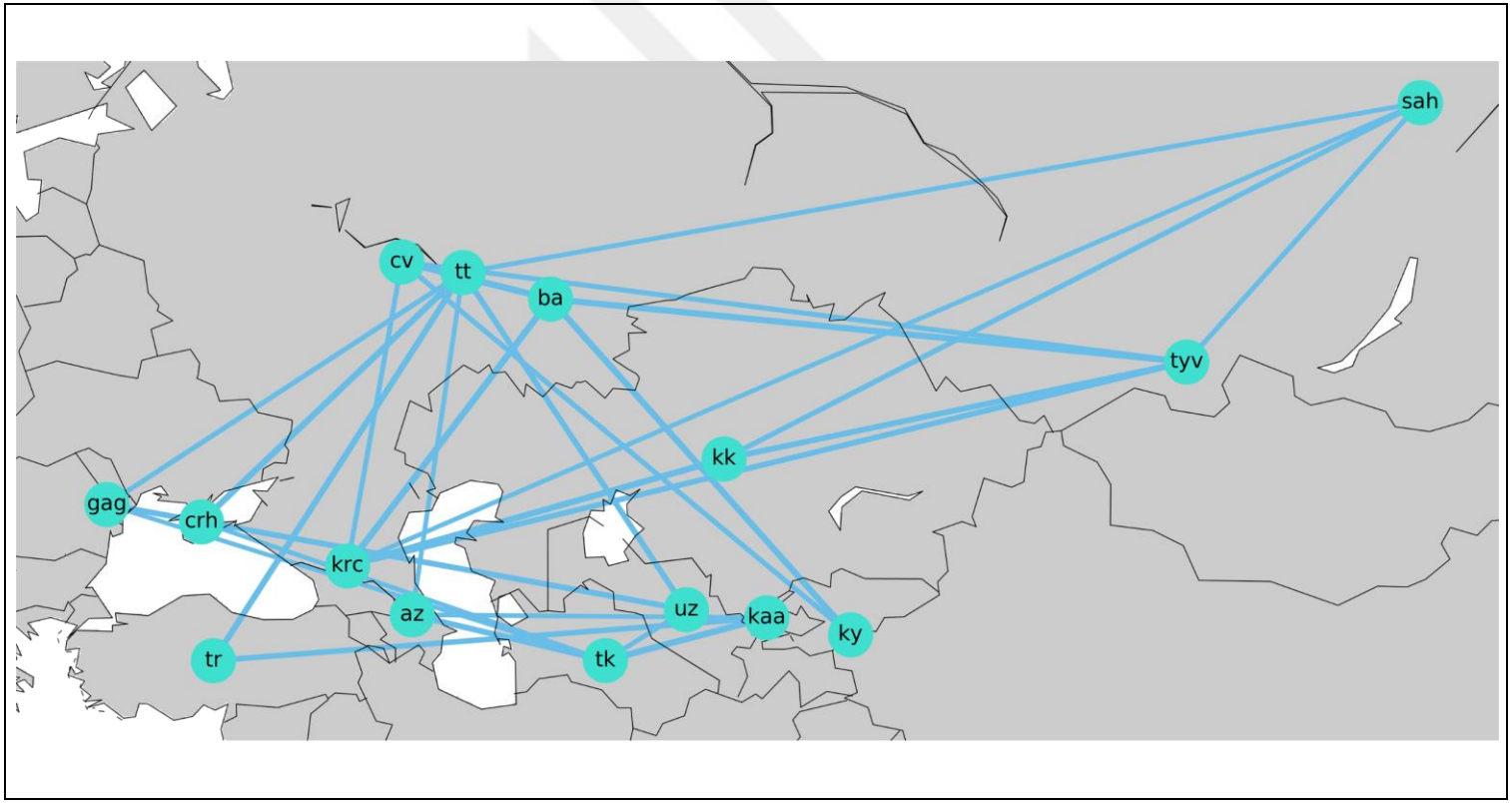


Figure 7.15: Very high similar Turkic language map. Map showing very high similar Turkic language and the geographical distance between cities where the languages are spoken. Red links mean more than 75% similarity.

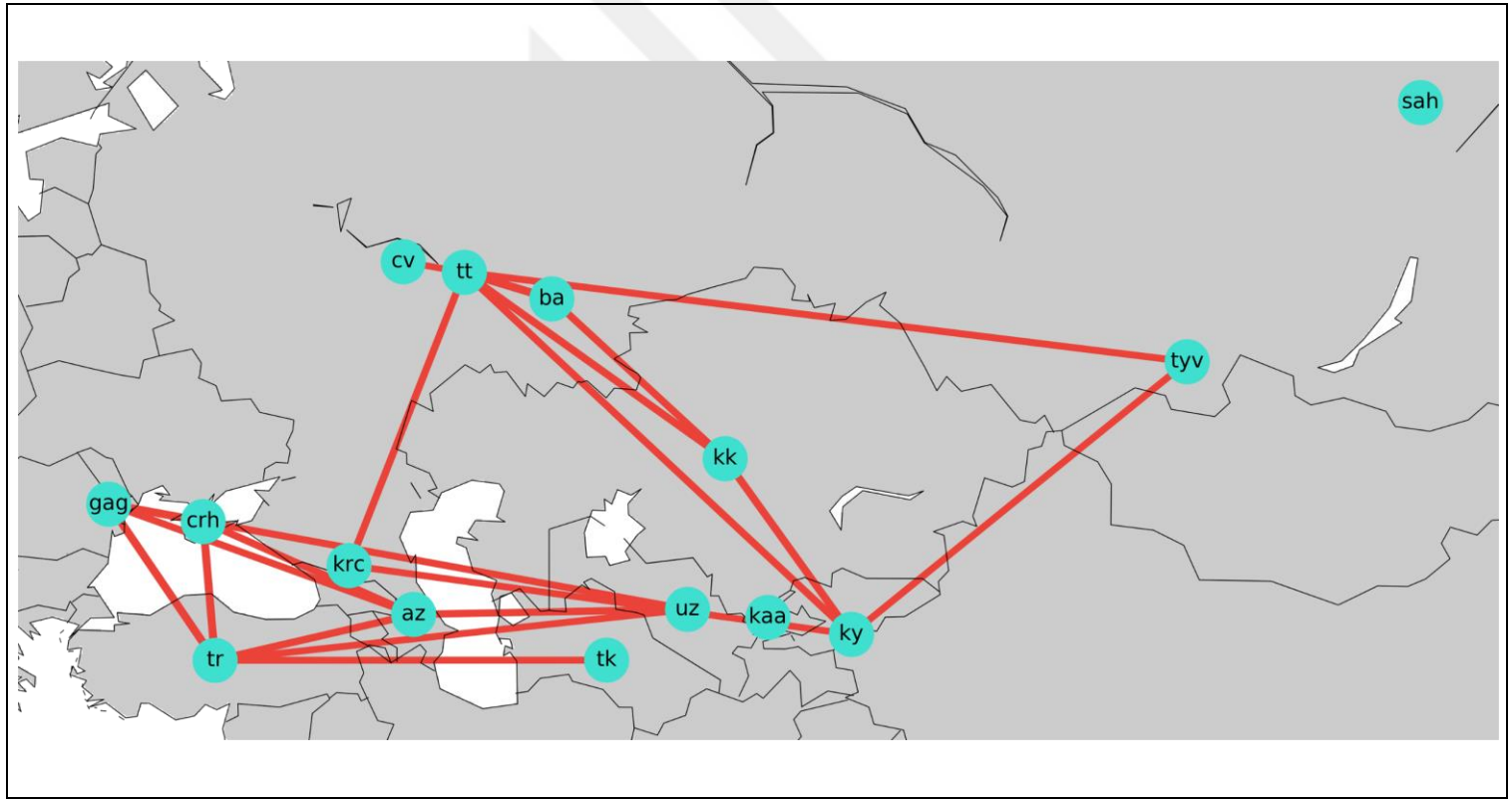


Table 7.4: Turkic languages geographical distance. The geographical distance between cities where the Turkic languages are spoken given by kilometers.

| | **az** | **ba** | **crh** | **cv** | **gag** | **kaa** | **kk** | **krc** | **ky** | **sah** | **tk** | **tr** | **tt** | **tyv** | **uz** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **az** | 0 | 1693 | 1215 | 1715 | 1672 | 1858 | 1772 | 492 | 2292 | 5407 | 1037 | 1067 | 1688 | 3862 | 1440 |
| **ba** | 1693 | 0 | 1835 | 610 | 2147 | 1747 | 1020 | 1515 | 1999 | 3758 | 1716 | 2317 | 366 | 2624 | 1560 |
| **crh** | 1215 | 1835 | 0 | 1434 | 464 | 2857 | 2480 | 754 | 3266 | 5470 | 2178 | 711 | 1585 | 4430 | 2463 |
| **cv** | 1715 | 610 | 1434 | 0 | 1655 | 2287 | 1606 | 1375 | 2578 | 4036 | 2070 | 2040 | 244 | 3173 | 2035 |
| **gag** | 1672 | 2147 | 464 | 1655 | 0 | 3305 | 2886 | 1218 | 3708 | 5634 | 2641 | 960 | 1848 | 4769 | 2917 |
| **kaa** | 1858 | 1747 | 2857 | 2287 | 3305 | 0 | 775 | 2147 | 435 | 4333 | 892 | 2919 | 2070 | 2298 | 418 |
| **kk** | 1772 | 1020 | 2480 | 1606 | 2886 | 775 | 0 | 1882 | 979 | 3826 | 1168 | 2729 | 1371 | 2090 | 761 |
| **krc** | 492 | 1515 | 754 | 1375 | 1218 | 2147 | 1882 | 0 | 2569 | 5273 | 1426 | 853 | 1409 | 3935 | 1741 |
| **ky** | 2292 | 1999 | 3266 | 2578 | 3708 | 435 | 979 | 2569 | 0 | 4123 | 1319 | 3351 | 2347 | 1981 | 853 |
| **sah** | 5407 | 3758 | 5470 | 4036 | 5634 | 4333 | 3826 | 5273 | 4123 | 0 | 4992 | 6051 | 3910 | 2280 | 4543 |
| **tk** | 1037 | 1716 | 2178 | 2070 | 2641 | 892 | 1168 | 1426 | 1319 | 4992 | 0 | 2101 | 1919 | 3123 | 505 |
| **tr** | 1067 | 2317 | 711 | 2040 | 960 | 2919 | 2729 | 853 | 3351 | 6051 | 2101 | 0 | 2142 | 4788 | 2502 |
| **tt** | 1688 | 366 | 1585 | 244 | 1848 | 2070 | 1371 | 1409 | 2347 | 3910 | 1919 | 2142 | 0 | 2947 | 1840 |
| **tyv** | 3862 | 2624 | 4430 | 3173 | 4769 | 2298 | 2090 | 3935 | 1981 | 2280 | 3123 | 4788 | 2947 | 0 | 2621 |
| **uz** | 1440 | 1560 | 2463 | 2035 | 2917 | 418 | 761 | 1741 | 853 | 4543 | 505 | 2502 | 1840 | 2621 | 0 |

### 7.2.2. Arabic Dialects

Using the same proposed model for Turkic languages, we use the MADAR dataset to measure distances between Arabic dialects. We train the n-gram model where n=5. Table 7.5 shows the perplexity-based rounded distance values each pair of Arabic dialects. We added Persian language as a baseline. Figure 7.18 shows the distances heatmap of these distances.

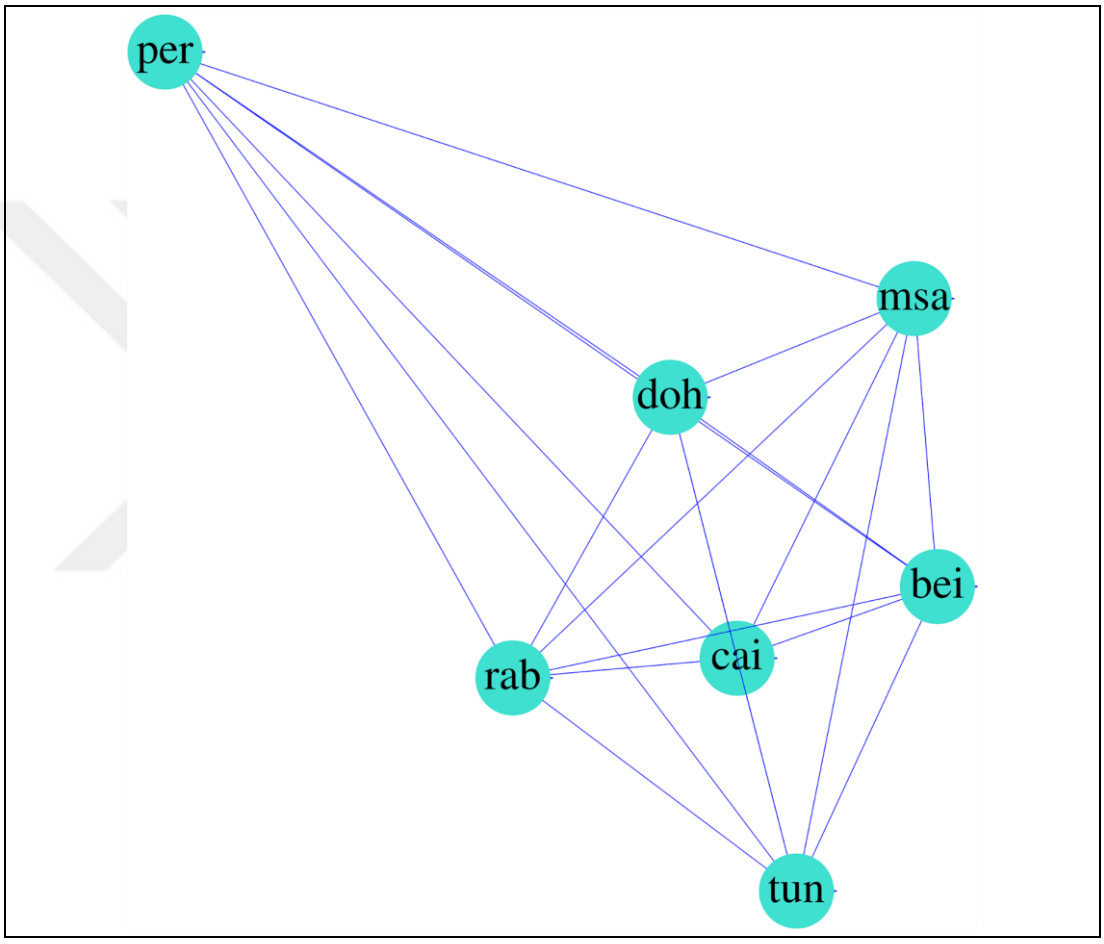



Figure 7.16: Arabic dialects and Persian language distance graph. Edge length represents the distance value between two languages on both sides.

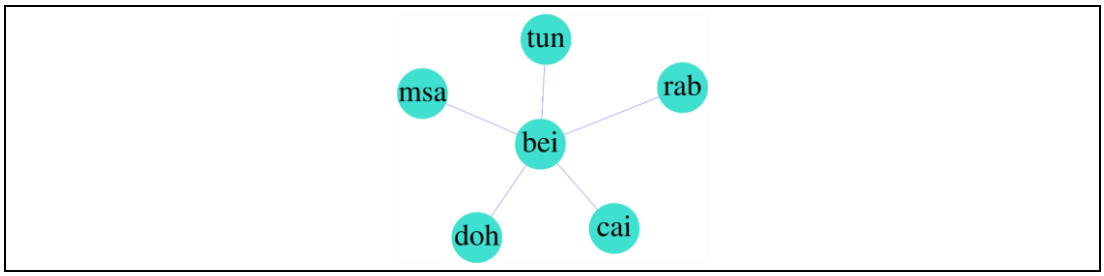



Figure 7.17: Arabic dialects' MST distance graph. Edge length represents the distance value between two languages on both sides. We use Kruskal's algorithm to generate the minimum spanning tree.

Table 7.5: Distances between Arabic dialects and Persian.

| | bei | cai | doh | msa | rab | tun | per |
|---|---|---|---|---|---|---|---|
| **bei** | 26.67 | 32.36 | 37.5 | 52.92 | 32.27 | 46.58 | 228.36 |
| **cai** | 28.82 | 20.99 | 41.49 | 44.56 | 39.73 | 40.85 | 818.71 |
| **doh** | 28.16 | 27.29 | 24.36 | 39.18 | 29.65 | 36.45 | 152.14 |
| **msa** | 31.97 | 37.71 | 31.52 | 29.26 | 33.35 | 32.04 | 612.92 |
| **rab** | 36.57 | 31.62 | 40.15 | 43.51 | 20.83 | 38.53 | 1311.62 |
| **tun** | 27.81 | 34.21 | 33.12 | 34.13 | 31.87 | 22.11 | 874.31 |
| **per** | 175.57 | 166.35 | 199.76 | 239.64 | 131.39 | 215.5 | 51.13 |

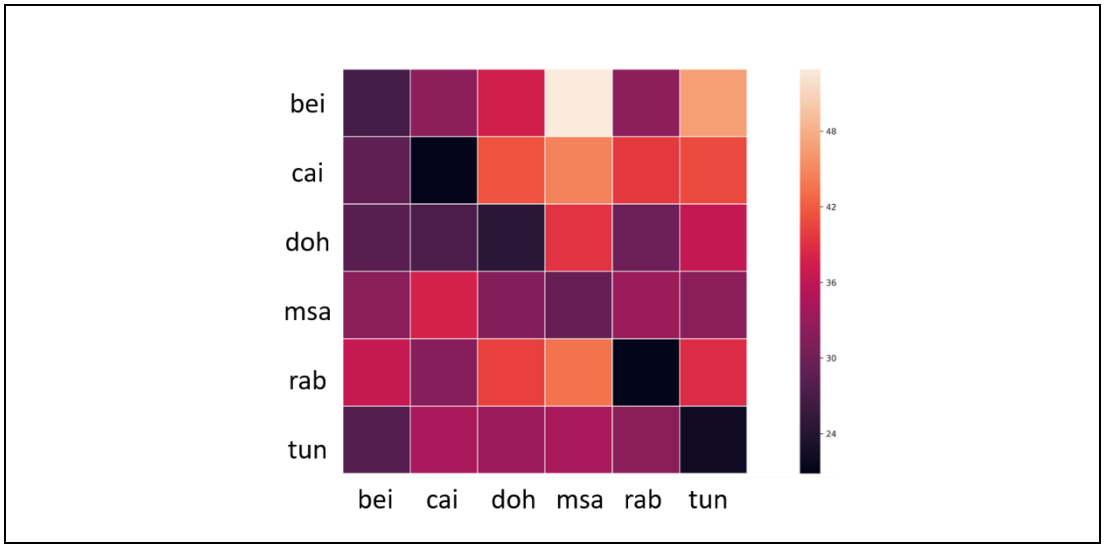
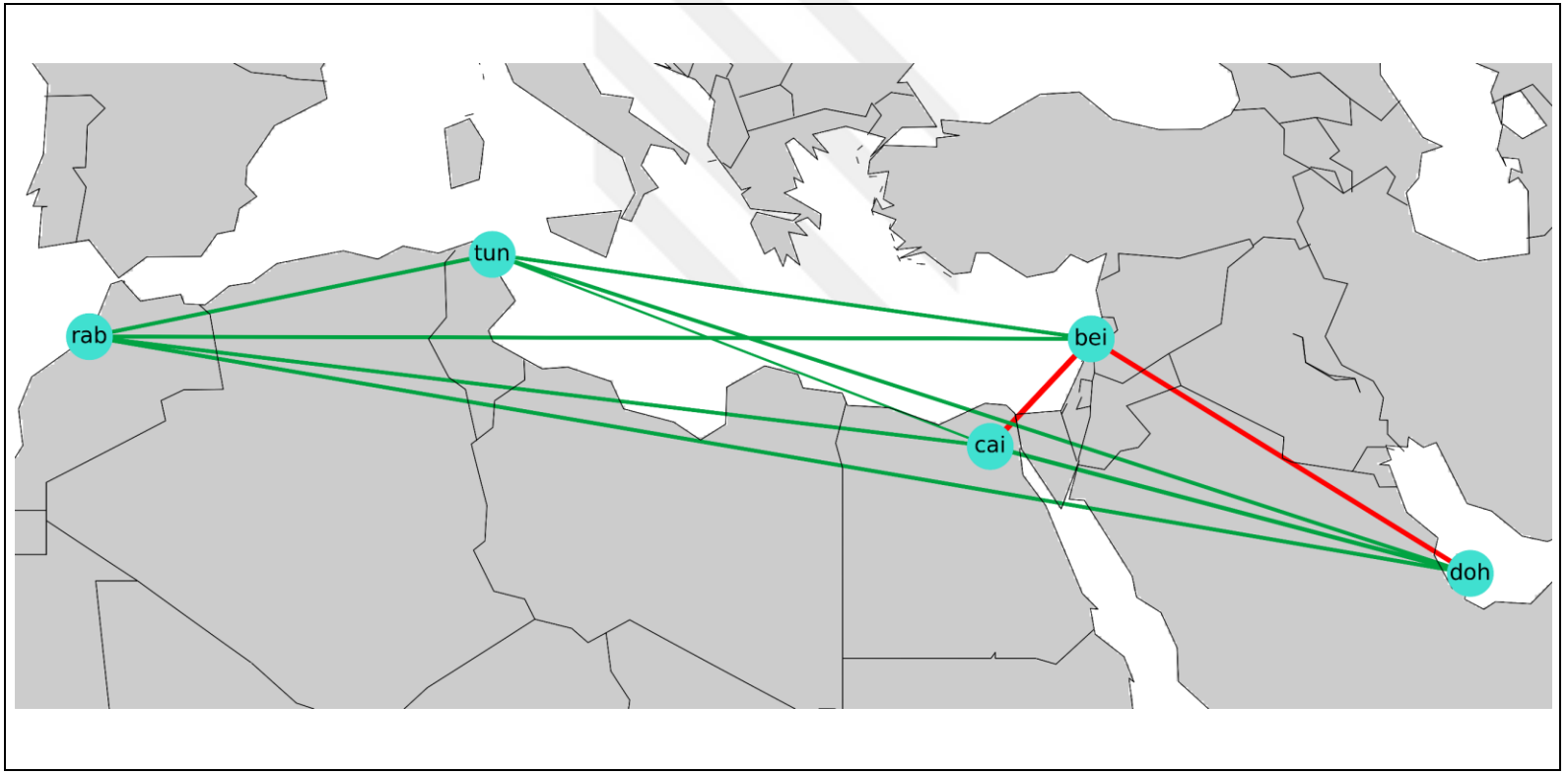



Figure 7.18: Heatmap showing similarities between Arabic dialects.

Figure 7.19: Arabic dialects similarities map. Map showing similarities between Arabic dialects and the geographical distance between cities where the dialects are spoken. The thickness and color of the links express how similar the language is. Blue color means over 50% similarity, while green color means less than 50% similarity.



To evaluate our results, we drew a map showing the location of the cities in which Arabic dialects are spoken, and thus easy to estimate the geographical distance between these cities, as well as we created links showing the similarity value between languages according to color and thickness. The map is shown in Figure 7.19. The distance in kilometers between cities is shown in Table 7.6.

Table 7.6: Arabic dialects geographical distance. The geographical distance between cities where the Arabic dialects are spoken given by kilometers.

| | **bei** | **cai** | **doh** | **rab** | **tun** |
|---|---|---|---|---|---|
| **bei** | 0 | 587 | 1819 | 3887 | 2317 |
| **cai** | 587 | 0 | 2067 | 3604 | 2091 |
| **doh** | 1819 | 2067 | 0 | 5661 | 4113 |
| **rab** | 3887 | 3604 | 5661 | 0 | 1577 |
| **tun** | 2317 | 2091 | 4113 | 1577 | 0 |

# 8. CONCLUSION

This study examines the change of natural languages through time and location by employing quantitative models using the techniques of natural language processing.

Through the two proposed models, we measure, in an unsupervised fashion, the change of natural languages. We achieve that by measuring the inter-language similarity, and by searching for word substitutions.

By utilizing word embeddings, adversarial training, and space alignment techniques, we study the development of the Turkish language during the past 100 years. We extract a list of word replacements in addition to infer the periods in which the language has changed significantly. By employing language modeling and its evaluation methods, we create the similarity matrix for Turkic languages as well as the Arabic dialects. We visualize the similarity matrices via heatmaps. Moreover, we draw the similarities in geographical maps that show the distances between the cities where these languages are spoken.

Our work confirms what linguists have inferred, where they mentioned about a high degree of mutual intelligibility among the various Turkic languages. Although linguistics methods of classification vary, the Turkic languages are usually considered to be divided equally into two branches: Oghur, the only surviving member of which is Chuvash and Common Turkic, which includes all other Turkic languages including the Oghuz sub-branch.

This study helps linguistics in analyzing natural languages and their relationships. As future work, we will utilize and develop the models proposed in this study to improve machine translation models between under-resourced languages or dialects.

# REFERENCES


[1] Adda G., Stüker S., Adda-Decker M., Ambouroue O., Besacier L., Blachon D., Bonneau-Maynard H., Godard P., Hamlaoui F., Idiatov D., (2016), "Breaking the unwritten language barrier: The BULB project", Procedia Computer Science, 81 (1), 8-14.

[2] Alexandre S., Aline V., (2018), "Incorporating subword information into matrix factorization word embeddings", arXiv e-prints, 1805.03710 (1), 1-6.

[3] Alexis C., Kiela D., (2018), "Senteval: An evaluation toolkit for universal sentence representations", arXiv e-prints, 1803.05449 (1), 1-6.

[4] Almahdi M. E., Akgül Y. S., (2019), "Automatic Detection of Word Substitutions within a Language over Periods of Time", 2019 4th International Conference on Computer Science and Engineering (UBMK), 478-481, Samsun, Turkey, 11-15, September.

[5] Anastasopoulos A., Chiang D., (2017), "A case study on using speech-to-translation alignments for language documentation", arXiv e-prints, 1702.04372 (1), 1-10.

[6] Antonios A., Chiang D., Duong L., (2016), "An unsupervised probability model for speech-to-translation alignment of low-resource languages", arXiv e-prints, 1609.08139 (1), 1-10.

[7] Artetxe M., Labaka G., Agirre E., Cho K., (2017), "Unsupervised neural machine translation", arXiv e-prints, 1710.11041 (1), 1-12.

[8] Aufrant L., Wisniewski G., Yvon F., (2016), "Zero-resource dependency parsing: Boosting delexicalized cross-lingual transfer with linguistic knowledge", Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 119-130, Osaka, Japan, December.

[9] Bahdanau D., Cho K., Bengio Y., (2014), "Neural machine translation by jointly learning to align and translate", arXiv e-prints, 1409.0473 (1), 1-15.

[10] Bahl L. R., Jelinek F., Mercer R. L., (1983), "A maximum likelihood approach to continuous speech recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1 (1), 179-190.

[11] Balikas G., Laclau C., Redko I., Amini M.-R., (2018), "Cross-lingual document retrieval using regularized wasserstein distance", European

Conference on Information Retrieval, 398-410, Grenoble, France, 26-29, March.

[12] Baum L. E., Petrie T., Soules G., Weiss N., (1970), "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains", The annals of mathematical statistics, 41 (1), 164-171.

[13] Bengio Y., Ducharme R., Vincent P., Jauvin C., (2003), "A neural probabilistic language model", Journal of machine learning research, 3 (1), 1137-1155.

[14] Bojanowski P., Grave E., Joulin A., Mikolov T., (2017), "Enriching word vectors with subword information", Transactions of the Association for Computational Linguistics, 5 (1), 135-146.

[15] Bouamor H., Hassan S., Habash N., (2019), "The MADAR shared task on Arabic fine-grained dialect identification", Proceedings of the Fourth Arabic Natural Language Processing Workshop, 199-207, Florence, Italy, 28-30, July.

[16] Britz D., Goldie A., Luong M.-T., Le Q., (2017), "Massive exploration of neural machine translation architectures", arXiv e-prints, 1703.03906 (1), 1-9.

[17] Brown P. F., Cocke J., Della Pietra S. A., Della Pietra V. J., Jelinek F., Lafferty J. D., Mercer R. L., Roossin P. S., (1990), "A statistical approach to machine translation", Computational linguistics, 16 (2), 79-85.

[18] Bynon T., (1977), "Historical linguistics", 1st Edition, Cambridge University Press.

[19] Campbell G. L., (2003), "Concise compendium of the world's languages", 1st Edition, Routledge.

[20] Cavnar W. B., Trenkle J. M., (1994), "N-gram-based text categorization", SDAIR-94, 161175 (1), 1-14.

[21] Cer D., Yang Y., Kong S.-y., Hua N., Limtiaco N., John R. S., Constant N., Guajardo-Cespedes M., Yuan S., Tar C., (2018), "Universal sentence encoder", arXiv e-prints, 1803.11175 (1), 1-7.

[22] Chiao Y.-C., Zweigenbaum P., (2002), "Looking for candidate translational equivalents in specialized, comparable corpora", Proceedings of the 19th international conference on Computational linguistics, 1-5, PA, United States, August.

[23] Cho K., Van Merriënboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., Bengio Y., (2014), "Learning phrase representations using RNN

encoder-decoder for statistical machine translation", arXiv e-prints, 1406.1078 (1), 1-15.

[24] Collobert R., Weston J., (2008), "A unified architecture for natural language processing: Deep neural networks with multitask learning", Proceedings of the 25th international conference on Machine learning, 160-167, NY, United States, July.

[25] Conneau A., Kiela D., Schwenk H., Barrault L., Bordes A., (2017), "Supervised learning of universal sentence representations from natural language inference data", arXiv e-prints, 1705.02364 (1), 1-12.

[26] Conneau A., Lample G., Ranzato M., Denoyer L., Jégou H., (2017), "Word Translation Without Parallel Data", arXiv e-prints, 1710.04087 (1), 1-14.

[27] Delpech E. M., (2014), "Comparable corpora and computer-assisted translation", 1st Edition, John Wiley and Sons.

[28] Dennis J. E., Moré J. J., (1977), "Quasi-Newton methods, motivation and theory", SIAM review, 19 (1), 46-89.

[29] Devlin J., Chang M.-W., Lee K., Toutanova K., (2018), "Bert: Pre-training of deep bidirectional transformers for language understanding", arXiv preprint arXiv:1810.04805, 1810.04805 (1), 1-16.

[30] Divvala S. K., Farhadi A., Guestrin C., (2014), "Learning everything about anything: Webly-supervised visual concept learning", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3270-3277, Columbus, OH, USA, 23-28, June.

[31] Dos Santos C., Gatti M., (2014), "Deep convolutional neural networks for sentiment analysis of short texts", Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 69-78, Dublin, Ireland, August.

[32] Duong L., Anastasopoulos A., Chiang D., Bird S., Cohn T., (2016), "An attentional model for speech translation without transcription", Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 949-959, San Diego, California, June.

[33] Dybo A. V., (2007), "Chronology of Türkic languages and linguistic contacts of early Türks", 1st Edition, Moscow.

[34] Eades P., (1984), "A heuristic for graph drawing", Congressus numerantium, 42 (1), 149-160.

[35] Eddy S. R., (1996), “Hidden markov models”, Current opinion in structural biology, 6 (3), 361-365.

[36] Emmanuel M., Daille B., (2012), “Revising the compositional method for terminology acquisition from comparable corpora”, Proceedings of COLING 2012, 1797-1810, Mumbai, India, December.

[37] Ferrero J., Agnes F., Besacier L., Schwab D., (2017), “UsingWord Embedding for Cross-Language Plagiarism Detection”, arXiv e-prints, 1702.03082 (1), 1-7.

[38] Fung P., (1998), “A statistical view on bilingual lexicon extraction: from parallel corpora to non-parallel corpora”, Conference of the Association for Machine Translation in the Americas, 1-17, DC, USA, September.

[39] Gaussier E., Renders J.-M., Matveeva I., Goutte C., Déjean H., (2004), “A geometric view on bilingual lexicon extraction from comparable corpora”, Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), 526-533, Barcelona, Spain, July.

[40] Godard P., Adda G., Adda-Decker M., Allauzen A., Besacier L., Bonneau-Maynard H., Kouarata G.-N., Löser K., Rialland A., Yvon F., (2016), “Preliminary experiments on unsupervised word discovery in mboshi”, Interspeech 2016, 1-6, San-Francisco, United States, September.

[41] Goldberg Y., (2019), “Assessing BERT's Syntactic Abilities”, arXiv e-prints, 1901.05287 (1), 1-4.

[42] Goldman E., Goldberger J., (2017), “Structured image classification from conditional random field with deep class embedding”, arXiv e-prints, 1705.07420 (1), 1-10.

[43] Good I. J., (1953), “The population frequencies of species and the estimation of population parameters”, Biometrika, 40 (3), 237-264.

[44] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., (2014), “Generative adversarial nets”, Advances in neural information processing systems, 2672-2680, Montreal, Canada, December.

[45] Gu J., Hassan H., Devlin J., Li V. O., (2018), “Universal neural machine translation for extremely low resource languages”, arXiv e-prints, 1802.05368 (1), 1-11.

[46] Hamilton W. L., Leskovec J., Jurafsky D., (2016), “Diachronic word embeddings reveal statistical laws of semantic change”, arXiv e-prints, 1605.09096 (1), 1-13.

[47] Hazem A., Morin E., (2017), “Bilingual word embeddings for bilingual terminology extraction from specialized comparable corpora”, Proceedings of the Eighth International Joint Conference on Natural Language Processing, 685-693, Taipei, Taiwan, November.

[48] He D., Xia Y., Qin T., Wang L., Yu N., Liu T.-Y., Ma W.-Y., (2016), “Dual learning for machine translation”, Advances in Neural Information Processing Systems, 820-828, Barcelona, Spain,December.

[49] Hochreiter S., Schmidhuber J., (1997), “Long short-term memory”, Neural computation, 9 (8), 1735-1780.

[50] Hull J. J., Srihari S. N., (1982), “Experiments in text recognition with binary n-gram and viterbi algorithms”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1 (1), 520-530.

[51] Hwa R., Resnik P., Weinberg A., Cabezas C., Kolak O., (2005), “Bootstrapping parsers via syntactic projection across parallel texts”, Natural language engineering, 11 (3), 311-325.

[52] Iyyer M., Manjunatha V., Boyd-Graber J., Daumé III H., (2015), “Deep unordered composition rivals syntactic methods for text classification”, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 1, 1681-1691, Beijing, China, July.

[53] Jabaian B., Besacier L., Lefevre F., (2012), “Comparison and combination of lightly supervised approaches for language portability of a spoken language understanding system”, IEEE Transactions on Audio, Speech, and Language Processing, 21 (3), 636-648.

[54] Jakubina L., Langlais P., (2017), “Reranking translation candidates produced by several bilingual word similarity sources”, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, 605-611, Valencia, Spain, April.

[55] Jelinek F., (1971), “Markov Models and Linguistic Theory: An Experimental Study of a Model for Eng-lish”, Mouton De Gruyter, The Hague, 1 (1), 3.

[56] Jelinek F., Mercer R. L., Bahl L. R., Baker J. K., (1977), “Perplexity—a measure of the difficulty of speech recognition tasks”, The Journal of the Acoustical Society of America, 62 (1), 63-63.

[57] Johnson M., Schuster M., Le Q. V., Krikun M., Wu Y., Chen Z., Thorat N., Viégas F., Wattenberg M., Corrado G., (2017), “Google’s multilingual neural machine translation system: Enabling zero-shot translation”, Transactions of the Association for Computational Linguistics, 5 (1), 339-351.

[58] Kamada T., Kawai S., (1989), “An algorithm for drawing general undirected graphs”, Information processing letters, 31 (1), 7-15.

[59] Kamusella T., (2017), “The Arabic language: a Latin of modernity?”, Journal of Nationalism, Memory and Language Politics, 11 (2), 117-145.

[60] Klakow D., Peters J., (2002), “Testing the correlation of word error rate and perplexity”, Speech Communication, 38 (1), 19-28.

[61] Kneser R., Ney H., (1995), “Improved backing-off for m-gram language modeling”, 1995 International Conference on Acoustics, Speech, and Signal Processing, 1, 181-184, Michigan, USA, 9-12, May.

[62] Kruskal J. B., (1956), “On the shortest spanning subtree of a graph and the traveling salesman problem”, Proceedings of the American Mathematical society, 7 (1), 48-50.

[63] Kudo T., Yamamoto K., Matsumoto Y., (2004), “Applying conditional random fields to Japanese morphological analysis”, Proceedings of the 2004 conference on empirical methods in natural language processing, 230-237, Barcelona, Spain, July.

[64] Lample G., Conneau A., Denoyer L., Ranzato M., (2017), “Unsupervised machine translation using monolingual corpora only”, arXiv e-prints, 1711.00043 (1), 1-14.

[65] Landauer T. K., McNamara D. S., Dennis S., Kintsch W., (2013), “Handbook of latent semantic analysis”, 1st Edition, Psychology Press.

[66] Levy O., Goldberg Y., (2014), “Linguistic regularities in sparse and explicit word representations”, Proceedings of the eighteenth conference on computational natural language learning, 171-180, Baltimore, Maryland, June.

[67] Lidstone G. J., (1920), “Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities”, Transactions of the Faculty of Actuaries, 8 (13), 182-192.

[68] Lin Y., Michel J.-B., Aiden E. L., Orwant J., Brockman W., Petrov S., (2012), “Syntactic annotations for the google books ngram corpus”, Proceedings of the ACL 2012 system demonstrations, 169-174, Jeju Island, Korea, july.

[69] Mays E., Damerau F. J., Mercer R. L., (1991), “Context based spelling correction”, Information Processing and Management, 27 (5), 517-522.

[70] McCallum A., Li W., (2003), “Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons”,

Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003, 188-191, PA, United States.

[71] Melamud O., Goldberger J., Dagan I., (2016), "context2vec: Learning generic context embedding with bidirectional lstm", Proceedings of the 20th SIGNLL conference on computational natural language learning, 51-61, Berlin, Germany, August.

[72] Mikolov T., Chen K., Corrado G., Dean J., (2013a), "Efficient estimation of word representations in vector space", arXiv e-prints, 1301.3781 (1), 1-12.

[73] Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J., (2013b), "Distributed representations of words and phrases and their compositionality", Advances in neural information processing systems, 3111-3119, NV, United States, December.

[74] Morin E., Daille B., Takeuchi K., Kageura K., (2007), "Bilingual terminology mining-using brain, not brawn comparable corpora", Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 664-671, Prague, Czech Republic, June.

[75] Nie J.-Y., (2010), "Cross-language information retrieval", Synthesis Lectures on Human Language Technologies, 3 (1), 1-125.

[76] Onur Gungor M. T., Sönmez Ç., (2018, 5), "A Corpus of Grand National Assembly of Turkish Parliament's Transcripts", In Fišer D., Eskevich M., Jong F. (Ed.), Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 1-5, Miyazaki, Japan, 7-12, May.

[77] Padó S., Lapata M., (2009), "Cross-lingual annotation projection for semantic roles", Journal of Artificial Intelligence Research, 36 (1), 307-340.

[78] Pennington J., Socher R., Manning C., (2014), "Glove: Global vectors for word representation", Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 1532-1543, Doha, Qatar, October.

[79] Perone C. S., Silveira R., Paula T. S., (2018), "Evaluation of sentence embeddings in downstream and linguistic probing tasks", arXiv e-prints, 1806.06259 (1), 1-15.

[80] Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., (2018), "Deep contextualized word representations", arXiv e-prints, 1802.05365 (1), 1-15.

[81] Peterson D. J., (2015), "The art of language invention: From Horse-Lords to Dark Elves, the words behind world-building", 1st Edition, Penguin.

[82] Petrolito R., Dell'Orletta F., (2018), "Word Embeddings in Sentiment Analysis", In Proceedings of 5th Italian Conference on Computational Linguistics, 330-334, Turin, Italy, 10-12, December.

[83] Rapp R., (1999), "Automatic identification of word translations from unrelated English and German corpora", Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, 519-526, College Park, Maryland, USA, June.

[84] Robert-Ribes J., Mukhtar R. G., (1997), "Automatic generation of hyperlinks between audio and transcript", Fifth European Conference on Speech Communication and Technology, 1-4, Rhodes, Greece, 22-25 September.

[85] Salle A., Idiart M., Aline V., (2016), "Matrix factorization using window sampling and negative sampling for improved word representations", arXiv e-prints, 1606.00819 (1), 1-6.

[86] Sennrich R., Haddow B., Birch A., (2015), "Neural machine translation of rare words with subword units", arXiv e-prints, 1508.07909 (1), 1-11.

[87] Seymore K., McCallum A., Rosenfeld R., (1999), "Learning hidden Markov model structure for information extraction", AAAI-99 workshop on machine learning for information extraction, 37-42, IL, United States, July.

[88] Sha F., Pereira F., (2003), "Shallow parsing with conditional random fields", Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, 134-141, PA, United States.

[89] Sun F., Guo J., Lan Y., Xu J., Cheng X., (2015), "Learning word representations by jointly modeling syntagmatic and paradigmatic relations", Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 136-145, Beijing, China, July.

[90] Sundermeyer M., Schlüter R., Ney H., (2012), "LSTM neural networks for language modeling", Thirteenth annual conference of the international speech communication association, 1-4, Portland, Oregon, 9-13, September.

[91] Tekin T., (1978), "Türk Dilleri Ailesi I", Türk Dili, 37 (318), 173-183.

[92] Tiedemann J., (2014), "Rediscovering annotation projection for cross-lingual parser induction", Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics, 1854-1864, Dublin, Ireland, August.

[93] Toselli A. H., Romero V., Vidal E., (2011), “Alignment between text images and their transcripts for handwritten documents”, 1st Edition, Springer.

[94] Tran T., Phung D., Bui H., Venkatesh S., (2017), “Hierarchical semi-Markov conditional random fields for deep recursive sequential data”, Artificial Intelligence, 246 (1), 53-85.

[95] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I., (2017), “Attention is all you need”, Advances in neural information processing systems, 5998-6008, CA, United States, December.

[96] Véronis J., (2013), “Parallel Text Processing: Alignment and use of translation corpora”, 1st Edition, , Springer Science and Business Media.

[97] Verwimp L., Bellegarda J. R., (2019), “Reverse Transfer Learning: Can Word Embeddings Trained for Different NLP Tasks Improve Neural Language Models?”, arXiv e-prints, 1909.04130 (1), 1-5.

[98] Vinyals O., Toshev A., Bengio S., Erhan D., (2015), “Show and tell: A neural image caption generator”, Proceedings of the IEEE conference on computer vision and pattern recognition, 3156-3164, Boston, MA, USA, 7-12, June.

[99] Viterbi A., (1967), “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”, IEEE transactions on Information Theory, 13 (2), 260-269.

[100] Wang W., Pan S. J., Dahlmeier D., Xiao X., (2016), “Recursive neural conditional random fields for aspect-based sentiment analysis”, arXiv e-prints, 1603.06679 (1), 1-11.

[101] Web 1, (2019), https://www.ethnologue.com, (Retrieved 30/12/2019).

[102] Web 2, (2019), https://sozluk.gov.tr, (Retrieved 30/12/2019).

[103] Web 3, (2019), https://www.tbmm.gov.tr/develop/owa/tutanak_dergisi_pdfler.meclis_donemleri?v_meclisdonem=0, (Retrieved 30/12/2019).

[104] Web 4, (2019), https://github.com/attardi/wikiextractor, (Retrieved 30/12/2019).

[105] Web 5, (2019), https://dumps.wikimedia.org, (Retrieved 30/12/2019).

[106] Wisniewski G., Pécheux N., Gahbiche-Braham S., Yvon F., (2014), “Cross-lingual part-of-speech tagging through ambiguous learning”, Proceedings of

the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1779-1785, Doha, Qatar, October.

[107] Wu Y., Schuster M., Chen Z., Le Q. V., Norouzi M., Macherey W., Krikun M., Cao Y., Gao Q., Macherey K., (2016), "Google's neural machine translation system: Bridging the gap between human and machine translation", arXiv e-prints, 1609.08144 (1), 1-23.

[108] Yarowsky D., (2001), "Induce, Multilingual POS Tagger and NP bracketer via projection on aligned corpora", Proceedings of NAACL-01, 2001, 104 (485), 31-36.

[109] Zennaki O., Semmar N., Besacier L., (2016), "Inducing multilingual text analysis tools using bidirectional recurrent neural networks", arXiv e-prints, 1609.09382 (1), 1-12.

[110] Zissman M. A., (1996), "Comparison of four approaches to automatic language identification of telephone speech", IEEE Transactions on speech and audio processing, 4 (1), 31.

[111] Zweigenbaum P., Habert B., (2006), "Faire se rencontrer les parallèles: regards croisés sur l'acquisition lexicale monolingue et multilingue", Revue de sociolinguistique en ligne GLOTTOPOL, 8 (1), 22-44.

# BIOGRAPHY

Muhammed Enes ALMAHDI was born in Damascus, Syria in 1993. He graduated from the Department of Artificial Intelligence and Natural Language Processing, Faculty of Information Technology Engineering, Damascus University in 2015. He started his Master's at the Department of Computer Engineering, Engineering Faculty, Gebze Technical University at 2016. During his Master's, he worked on a project about social media analysis and anomaly detection. His research interests include natural language processing, machine learning, and deep learning.

# APPENDICES

## Appendix A: Publications on the thesis

Almahdi M. E., Akgül Y. S., (2019), "Automatic Detection of Word Substitutions within a Language over Periods of Time", In 2019 4th International Conference on Computer Science and Engineering (UBMK), 478-481.

## Appendix B: Word Substitution List

Table B.1: List of word substitutions extracted by the proposed model.

| Word | Replaced Word | Substitution Period(s) |
|---|---|---|
| ahkam | hüküm | 1985-1989 |
| arzuhal | dilekçe | 1985-1989 |
| aza | üye | 1945-1949, 1955-1959, 1985-1989 |
| başvekalet | başbakanlık | 1945-1949, 1955-1959, 1985-1989 |
| baytar | veteriner | 1940-1944 |
| bilahare | sonradan | 1945-1949 |
| bilhassa | özellikle | 1985-1989 |
| cürüm | suç | 1930-1934 |
| devamlı | sürekli | 1925-1929, 1945-1949, 1955-1959, 1985-1989 |
| devre | dönem | 1985-1989 |
| ecnebi | yabancı | 1945-1949, 1955-1959, 1985-1989 |
| ehemmiyet | önem | 1985-1989 |
| elbette | bittabi | 1930-1934 |
| elbette | muhakkak | 1985-1989 |
| encümen | komisyon | 1945-1949, 1955-1959, 1985-1989 |

Table B.1: continuation.

| Word | Replaced Word | Substitution Period(s) |
|---|---|---|
| evvelki | önceki | 1985-1989 |
| eylemek | etmek | 1925-1929 |
| fayda | yarar | 1985-1989, 2010-2014 |
| fena | kötü | 1945-1949 |
| fikir | düşünce | 1945-1949, 1955-1959, 1985-1989 |
| gayrimenkul | taşınmaz | 1985-1989, 1990-1994, 1995-1999 |
| hadise | olay | 1985-1989 |
| hakikaten | gerçekten | 1985-1989 |
| halbuki | oysa | 1985-1989 |
| harcırah | yolluk | 1945-1949, 1955-1959, 2010-2014 |
| hasebiyle | sebebiyle | 1935-1939, 1940-1944, 1945-1949 |
| hisse | pay | 1990-1994 |
| hitam | nihayet | 1985-1989 |
| hudut | sınır | 1985-1989 |
| hükmi | tüzel | 1945-1949, 1955-1959, 9 |
| icar | kira | 1940-1944 |
| içtima | toplantı | 1945-1949, 1955-1959, 1985-1989 |
| ihtiyat | yedek | 1940-1944 |
| iktisadi | ekonomik | 1985-1989 |
| ilim | bilim | 1985-1989 |
| ilmi | bilimsel | 1985-1989 |
| imkan | olanak | 1925-1929 |
| inkişaf | gelişme | 1985-1989 |
| inzibat | disiplin | 1945-1949, 1950-1954, 1955-1959 |
| iştigal | meşgul | 1985-1989 |

Table B.1: continuation.

| Word | Replaced Word | Substitution Period(s) |
|---|---|---|
| istihlak | tüketim | 1985-1989 |
| istihsal | üretim | 1985-1989 |
| istikraz | borçlanma | 1985-1989 |
| istimlak | kamulaştırma | 1945-1949, 1950-1954, 1985-1989 |
| istinaden | göre | 1940-1944 |
| istinat | dayalı | 1985-1989 |
| izah | ifade | 1925-1929 |
| izahat | cevap | 1935-1939, 1940-1944 |
| kabil | mümkün | 1935-1939, 1940-1944 |
| kafi | yeterli | 1985-1989 |
| kati | kesin | 1985-1989 |
| mahsul | müstahsil | 1985-1989, 1995-1999, 2000-2004 |
| malik | sahip | 1925-1929, 1945-1949 |
| malumat | bilgi | 1985-1989 |
| mani | engel | 1985-1989 |
| marifetiyle | tarafından | 1985-1989 |
| masarif | masraf | 1925-1929 |
| matbuat | basın | 1945-1949 |
| mecburi | zorunlu | 1985-1989 |
| meccanen | parasız | 1935-1939 |
| mektep | okul | 1935-1939 |
| menfaat | fayda | 1930-1934, 1935-1939 |
| mesele | konu | 1985-1989 |
| mesul | sorumlu | 1985-1989 |
| mesuliyet | sorumluluk | 1985-1989 |
| misal | örnek | 1985-1989 |
| modern | çağdaş | 1985-1989, 2010-2014 |
| muallim | öğretmen | 1940-1944 |

Table B.1: continuation.

| Word | Replaced Word | Substitution Period(s) |
|---|---|---|
| muamele | işlem | 1985-1989 |
| muayyen | belirli | 1985-1989 |
| mucibince | gereğince | 1945-1949 |
| mugayir | aykırı | 1935-1939 |
| muhakeme | yargılama | 1985-1989 |
| muhalif | aykırı | 1985-1989 |
| mühim | önemli | 1985-1989 |
| muhtelit | karma | 1945-1949, 1955-1959, 1985-1989 |
| muhtevi | ihtiva | 1985-1989 |
| mukavele | sözleşme | 1945-1949,1955-1959, 1985-1989 |
| mukayyet | kayıtlı | 1945-1949 |
| münakale | aktarma | 1945-1949, 1955-1959, 1985-1989 |
| münakaşa | müzakere | 1985-1989 |
| münasip | muvafık | 1985-1989 |
| müracaat | başvuru | 2000-2004 |
| müspet | olumlu | 1985-1989 |
| müstesna | hariç | 1935-1939 |
| mutabakat | uygunluk | 1945-1949, 1950-1954, 1985-1989, 2005-2009 |
| muteber | geçerli | 1985-1989 |
| mütehassıs | uzman | 1945-1949 |
| mütevellit | dolayı | 1985-1989 |
| muvaffak | başarılı | 1985-1989 |
| muvafık | uygun | 1935-1939, 1940-1944, 1945-1949 |
| muvafık | şayan | 1985-1989 |
| müzakere | görüşme | 1985-1989 |
| nakliye | taşıma | 1945-1949 |

Table B.1: continuation.

| Word | Replaced Word | Substitution Period(s) |
|---|---|---|
| namına | adına | 1945-1949 |
| neşriyat | yayın | 1945-1949 |
| netice | sonuç | 1985-1989 |
| numaralı | sayılı | 1935-1939 |
| politika | siyaset | 1945-1949, 1995-1999 |
| reisicumhur | cumhurbaşkanı | 1945-1949, 1955-1959, 1985-1989 |
| rey | oy | 1945-1949, 1955-1959, 1985-1989 |
| rica | istirham | 1985-1989 |
| şahıs | kişi | 1925-1929, 1945-1949, 1955-1959, 1985-1989 |
| şahsi | kişisel | 1985-1989 |
| salahiyet | yetki | 1945-1949, 1950-1954, 1985-1989 |
| sarih | açıkça | 1945-1949, 1950-1954, 1985-1989 |
| sebebiyle | nedeniyle | 1985-1989 |
| şifahi | tahriri | 1940-1944, 1950-1954, 1985-1989 |
| tahkikat | soruşturma | 1985-1989 |
| tahsisat | ödenek | 1945-1949, 1955-1959, 1985-1989 |
| takibat | soruşturma | 1945-1949, 1950-1954, 1985-1989 |
| takrir | önerge | 1945-1949, 1950-1954, 1985-1989 |
| talebe | öğrenci | 1945-1949,1955-1959, 1985-1989 |
| tamir | onarım | 1985-1989 |
| tasrih | ifade | 1985-1989 |
| tatbikat | uygulama | 1985-1989 |

Table B.1: continuation.

| Word | Replaced Word | Substitution Period(s) |
|---|---|---|
| tatbiki | uygulama | 1985-1989 |
| tediye | ödeme | 1945-1949, 1985-1989 |
| temettü | kazanç | 1925-1929 |
| tenvir | aydınlatma | 1945-1949 |
| teshin | yakacak | 1945-1949 |
| tespit | tayin | 1985-1989 |
| teşrii | yasama | 1985-1989 |
| tetkikat | inceleme | 1945-1949, 1955-1959, 1985-1989 |
| ücretli | sözleşmeli | 1985-1989 |
| ulaştırma | münakalat | 1945-1949, 1955-1959, 1985-1989 |
| ulvi | eşref | 1950-1954, 1955-1959, 1985-1989 |
| umum | genel | 1945-1949, 1955-1959, 1985-1989 |
| vazife | görev | 1945-1949 |
| vaziyet | durum | 1955-1959 |
| vekalet | bakanlık | 1945-1949, 1955-1959, 1985-1989, 2000-2004 |
| vekil | bakan | 1935-1939, 1940-1944, 1945-1949, 1985-1989 |
| vesaire | benzeri | 1945-1949 |
| yakıt | akaryakıt | 1940-1944, 1950-1954, 1985-1989 |
| yolsuzluk | usulsüzlük | 1985-1989 |
| yürürlük | meriyet | 1945-1949, 1955-1959, 1985-1989 |
| zabıt | tutanak | 1945-1949, 1955-1959, 1985-1989 |
| zevat | zat | 1985-1989 |
| zirai | tarımsal | 2000-2004 |

## Appendix C: ISO 9:1995 Transliteration Table

Table C.1: ISO 9:1995 Transliteration table.

| Cyrillic | Latin | Cyrillic cont. | Latin cont. |
|---|---|---|---|
| А а | A a | П п | P p |
| Б б | B b | Р р | R r |
| В в | V v | С с | S s |
| Г г | G g | Т т | T t |
| Ґ ґ | G̀ g̀ | Ќ ќ | Ḱ ḱ |
| Д д | D d | Ћ ћ | Ć ć |
| Ѓ ѓ | Ǵ ǵ | У у | U u |
| Ђ ђ | Đ đ | Ў ў | Ŭ ŭ |
| Е е | E e | Ф ф | F f |
| Ё ё | Ë ë | Х х | H h |
| Є є | Ê ê | Ц ц | C c |
| Ж ж | Ž ž | Ч ч | Č č |
| З з | Z z | Џ џ | D̂ d̂ |
| S s | Ẑ ẑ | Ш ш | Š š |
| И и | I i | Щ щ | Ŝ ŝ |
| І і | Ì ì | Ъ ъ | ″ |
| Ї ї | Ï ï | Ы ы | Y y |
| Й й | J j | Ь ь | ′ |
| J j | J̌ ǰ | Ѣ ѣ | Ě ě |
| К к | K k | Э э | È è |
| Л л | L l | Ю ю | Û û |
| Љ љ | L̂ l̂ | Я я | Â â |
| М м | M m | ’ | ’ |
| Н н | N n | Ӂ ӂ | Ă ă |
| Њ њ | N̂ n̂ | Ѳ ѳ | F̀ f̀ |
| О о | O o | Ѵ ѵ | Ỳ ỳ |