AUTOMATIC GENERATION OF SCIENTIFIC TERMINOLOGY WITH DEEP LEARNING

DERİN ÖĞRENME İLE OTOMATİK BİLİM TERİMLERİ SÖZLÜĞÜ OLUŞTURULMASI

İPEK NUR KARAMAN

PROF. DR İLYAS ÇİÇEKLİ

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

fort he Award of the Degree of Master of Science

In Computer Engineering.

2021

ÖZET

DERİN ÖĞRENME İLE OTOMATİK BİLİM TERİMLERİ SÖZLÜĞÜ OLUŞTURULMASI

İpek Nur KARAMAN

Yüksek Lisans, Bilgisayar Mühendisliği

Tez Danışmanı: Prof. Dr. İlyas ÇİÇEKLİ

Eş danışman: Dr. Gönenç ERCAN

Haziran 2021, 62 sayfa

Otomatik terim çıkarımı, doğal dil işlemede önemli bir görevdir. Bu tezde, otomatik terminoloji çıkarımı üzerinde iki amaç için çalışılmıştır. Birinci amacımız, farklı bilim alanları için bilimsel terminolojinin tutarsızlığını ölçmektir. Bilimsel yazılarda terminoloji tutarlılığı, bilimsel bilginin araştırmacılar arasında yayılması açısından önemlidir. Bu tezde, terminoloji tutarsızlığını ölçen bir metrik önerilmekte ve otomatik terim çıkarımı ile istatistiksel makine çevirisi kullanılarak farklı bilim alanları için terminoloji tutarsızlığı ölçülmektedir. Sonuçlarımız, terminolojideki tutarsızlığa göre bilimsel grupların sıralamasının: PHY (Fizik Bilimleri ve Mühendislik) $>$ SOC (Sosyal ve Davranış Bilimleri) $>$ LIF (Yaşam Bilimleri) olduğunu göstermiştir. Ayrıca sonuçların doğrulanması için anket çalışması yapılmış ve anket sonuçları elde ettiğimiz sonuçları desteklemiştir. Bu tezin ikinci amacı, dizi etiketleme ile derin öğrenme yöntemlerine dayalı ortak çok dilli öğrenme ile çok dillilikten faydalanmak ve İngilizce verileri kullanarak Türkçe terminoloji çıkarımı performansını iyileştirmektir. Derin öğrenme ile otomatik terim çıkarımı, yeterli eğitim verisi bulunduğunda umut verici sonuçlar elde etmektedir. Ne yazık ki, bazı diller için

bazı bilim alanları eğitim verisi için gerekli olan kaynaklardan yoksun olabilir ve veri eksikliği yetersiz uyum nedeniyle otomatik terim çıkarımında düşük performansa neden olabilmektedir. Bu tez çalışmasında, metinlerden otomatik olarak terimleri çıkarmak için dizi etiketlemeli, çok dilli veriler üzerinde eğitilmiş ve bu sorunu çözmek için hizalanmış kelime temsilleri ile ortak çok dilli derin öğrenme modeli önerilmektedir. Değerlendirme sonuçlarımıza göre, çok dilli bir model, sınırlı eğitim verileriyle eğitilmiş tek dilli bir modelle karşılaştırıldığında, otomatik terim çıkarımında performans iyileştirmesi sağlamıştır. İyileştirme oranı bilim alanı ve verinin boyutuna göre değişmekle birlikte, değerlendirmemiz F1 puanındaki en yüksek gelişmenin Bilgisayar Bilimleri alanında 10,1 %, en az iyileştirmenin ise Elektronik Mühendisliği alanında 7,6 % olduğunu göstermektedir. Ayrıca çok dilli modelimiz, yeterli eğitim verisi ile eğitilmiş tek dilli bir modelle karşılaştırıldığında rekabetçi sonuçlar elde etmektedir.

Anahtar Kelimeler: Otomatik terim çıkarımı, terminoloji çevirisi, birlikte çok dilli öğrenme, sinirsel dizi etiketleme, derin öğrenme, iki yönlü uzun-kısa vadeli bellek, istatistiksel makine çevirisi

ABSTRACT

AUTOMATIC GENERATION OF SCIENTIFIC TERMINOLOGY WITH DEEP LEARNING

İpek Nur KARAMAN

Master of Science, Department of Computer Engineering

Supervisor: Prof. Dr. İlyas ÇİÇEKLİ

Co-Supervisor: Dr. Gönenç ERCAN

June 2021, 62 pages

Automatic term extraction is an essential task in natural language processing. In this thesis, we work on terminology extraction for two purposes. The first aim is to measure inconsistency of scientific terminology for different scientific disciplines. Terminology consistency in scientific writing is important for the dissemination of scientific information among researchers. In this thesis, we propose a metric that measures terminology inconsistency and we measure terminology inconsistency for different scientific disciplines by using automatic term extraction and statistical machine translation. Our results showed that the order of scientific groups by inconsistency in terminology is: PHY (Physical Sciences and Engineering) $>$ SOC (Social and Behavioral Sciences) $>$ LIF (Life Sciences). We also survey for verification of the results and survey results support our study. The second aim of this thesis is to leverage multilinguality with joint multilingual learning based on deep learning methods with sequence labeling and improve terminology extraction performance in Turkish by utilizing English data. Automatic term extraction using deep learning achieves promising results if sufficient training data exists. Unfortunately,

some languages may lack these resources in some scientific domains causing poor performance due to under-fitting. In this thesis, we propose a joint multilingual deep learning model with sequence labeling to extract terms, trained on multilingual data and aligned word embeddings to tackle this problem. Our evaluation results demonstrate that the multilingual model provides an improvement for automatic term extraction task when it is compared with a monolingual model trained with limited training data. Although the improvement rate varies according to domain and the size of the data, our evaluation shows that the highest improvement in F1-score is 10.1 % in the domain of Computer Science, the least improvement is 7.6 % in the domain of Electronic Engineering. Our multilingual model also achieves competitive results when it is compared with a monolingual model trained with sufficient training data.

Keywords: Automatic term extraction, terminology translation, joint multilingual learning, neural sequence labeling, deep learning, bidirectional long short-term memory, statistical machine translation

ACKNOWLEDGEMENT

I would like to thank my thesis supervisor Prof. Dr. İlyas ÇİÇEKLİ and my co-advisor Dr. Gönenç ERCAN for their guidance, efforts and continued support throughout this thesis.

I would finally like to thank my family for their endless love, support, help and patience.

CONTENTS

FIGURES

TABLES

ABBREVIATIONS

Abbreviations

NLP	Natural Language Processing
TF	Term Frequency
ATF	Average Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
CBOW	Continuous Bag of Words
POS	Part Of Speech
RNN	Recurrent Neural Network
LSTM	Long Short-term Memory
BiLSTM	BiDirectional Long Short-term Memory
HMM	Hidden Markov Model
CRF	Conditional Random Fields
ReLU	Rectified Linear Unit
LIF	Life Sciences
PHY	Physical Sciences and Engineering
SOC	Social and Behavioral Sciences
SMT	Statistical Machine Translation

1. INTRODUCTION

Terms are domain-specific words or word groups referring to domain specific entities or concepts. Automatic terminology extraction (also known as term extraction, term recognition) is an automatic text analysis method for identifying word or phrases that meet the criteria for terms. The goal of automatic terminology extraction is to extract terms from a given corpus automatically. Extracted terms can be used in other natural language processing tasks such as ontology learning, machine translation [1, 2]. Due to the importance of handling terminologies, different extraction methods have been proposed.

In this thesis, we work on terminology extraction with different problems and methods. We handle two different problems. The first problem is to measure inconsistency of scientific terminology for different scientific disciplines. The second problem is to leverage multilinguality with joint multilingual learning based on deep learning methods with sequence labeling and improve terminology extraction performance in Turkish by utilizing English data.

1.1. Inconsistency of Scientific Terminology

Scientific studies are generally published in English. However, publications in the native language are important for the dissemination of scientific knowledge among researchers. Scientific studies contain a large number of terms and new terms are constantly being added in continuously developing science fields such as Computer science, which are often translated differently by different researchers. This poses a discrepancy problem and makes it difficult to achieve unity in terminology. The most essential cause of the problem is the polyphony in both languages when translating scientific terms from one language to another.

Terminology consistency in scientific writing is important for the dissemination of scientific information among researchers. The reasons for terminology inconsistency in translations have been investigated in many studies for different languages and domains [3, 4, 5, 6]. However, no research has been conducted to measure inconsistency in terminology for scientific fields using bilingual texts and using NLP techniques. Also, to the

best of our knowledge, there is no previous study to compare the degree of terminology inconsistency for different scientific disciplines.

In this thesis, we measure terminology inconsistency for different scientific disciplines by using automatic term extraction and statistical machine translation. We propose a metric for measuring terminology inconsistency. We aim to reveal how terminology inconsistency varies for different scientific disciplines. For this, we apply both automatic terminology extraction and statistical machine translation methods in our bilingual (English-Turkish) source. Also, we support our work with a survey. The results of the survey support our study such that our proposed metric is a valid metric for measuring terminology inconsistency.

1.2. Automatic Term Extraction with Joint Multilingual Learning

Neural sequence labeling is one of the successfully used methods for keyword extraction [7, 8]. Since keyword extraction and term extraction are similar tasks, automatic term extraction task can also be treated as a sequence labeling problem and deep neural models are suitable for use. Although deep neural network models are state-of-the-art techniques for sequence labeling, they require sufficient data for training in order to obtain successful results [9]. However, some languages may have low resources for some domains and the lack of data causes underfitting for deep neural models [9].

Joint multilingual learning is an approach to enable to use natural language processing (NLP) techniques for low-resource languages. Joint multilingual learning enables to train a single model with mixed data in languages with rich resources and low resources. In order to share data and parameters, multilingual embeddings are used. The word vectors of different languages share the same word vector space. If the meanings of two different words in different languages are similar, their word vectors are close to each other in the vector space.

The motivation for our study is the question of whether it is possible to tackle the problem of the lack of data by using data in another language for automatic term extraction task. If it is possible, it is very beneficial for languages with limited resources. So, in this study, we utilize joint multilingual learning for automatic term extraction task. We trained

a single multilingual model on a mixed dataset in English and Turkish. In this way, it enables data and model parameter sharing. We used a shared word vector space as the primary language representation so that two similar words in English and Turkish will be close to each other in the vector space. To the best of our knowledge, there is no previous research that trains a multilingual neural model for automatic term extraction using joint multilingual learning approach. Also, there are no automatic term extraction studies with neural sequence labeling and deep learning in Turkish.

1.3. Thesis Goals

In this thesis, we have several goals:

- To perform bilingual terminology extraction with using the traditional pipeline of the terminology extraction and statistical machine translation methods to generate scientific terminology for different scientific disciplines.
- To measure terminology inconsistency by terminology inconsistency metric for different scientific disciplines and compare them in terms of terminology inconsistency
- To perform terminology extraction using deep learning with sequence labeling.
- To train monolingual and multilingual deep neural models for automatic term extraction task using word embeddings in Turkish and English.
- To improve terminology extraction performance in Turkish with joint multilingual learning by utilizing English data.
- To compare trained monolingual and multilingual deep learning models in terms of performance for different scientific disciplines.
- To discuss the reasons of the performance gap between trained monolingual and multilingual models.

1.4. Thesis Contributions

The main contributions of the thesis are follows:

- We proposed a metric to measure terminology inconsistency for scientific fields by using automatic term extraction and statistical machine translation.
- We revealed how the degree of terminology inconsistency varies for different scientific disciplines for English-Turkish language pairs.
- We proposed automatic term extraction method based on deep joint learning.
- We proved that automatic term extraction in one language can be improved by using a different language data in the same domain.

1.5. Thesis Outline

The structure of the thesis is given as follows:

- Chapter 2 presents the background and related work.
- Chapter 3 explains the dataset.
- Chapter 4 explains the bilingual automatic term extraction methodology in this study.
- Chapter 5 presents the method for measuring terminology inconsistency and discuss the differences between different scientific fields in terms of terminology inconsistency.
- Chapter 6 explains the multilingual term extraction with joint learning methodology in this thesis and discusses the results.
- Chapter 7 concludes this thesis with remarks.

2. BACKGROUND AND RELATED WORK

2.1. Automatic Term Extraction

Terms are domain-specific words or words groups such that they refer to entities that belong to a specific domain. Automatic term extraction is the task of identification of domain-specific words or phrases (i.e., terms) from text documents. Extracted terms can be used for further operations such as ontology learning, machine translation, etc. In literature, lots of term extraction methods are proposed until today. In this chapter, the general pipeline of term extraction is explained and also the proposed term extraction methods are discussed.

2.1.1. Pipeline of Term Extraction

Term extraction methods generally consist of four steps:

- Preprocessing
- Term candidates selection
- Term candidates scoring
- Term candidates ranking

, respectively [10], as shown in Figure 2.1.

Preprocessing Step

In the preprocessing step, some operations are performed on input texts such as sentence segmentation, word segmentation, and POS tagging to make input text ready for further steps. Sentence segmentation is basically converting input text into sentences. Generally, punctuations are used for splitting the input text. Word segmentation (Tokenization) is the process of dividing sentences into their component words. Words are usually separated by spaces for many languages using Latin alphabet.

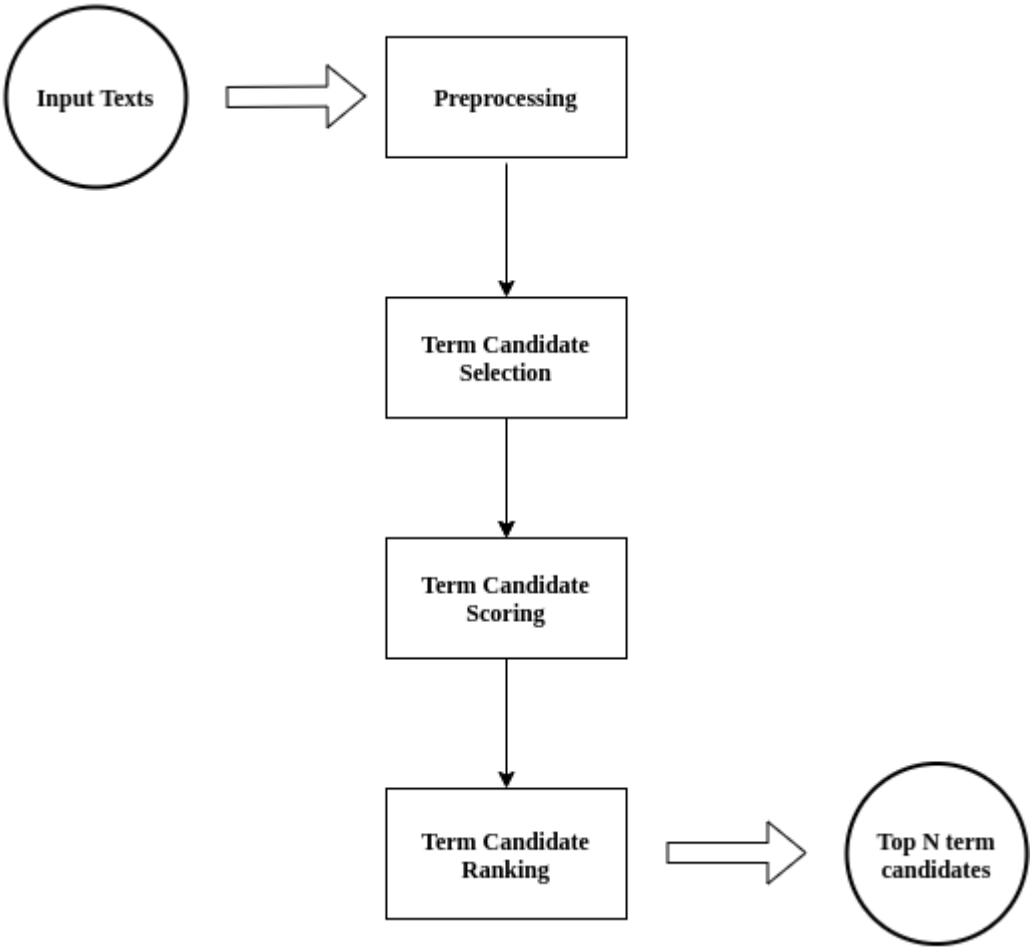

Fig. 2.1. The traditional pipeline of automatic term extraction

Part-of-speech tagging (POS tagging) is the marking operation of words in source texts with considering definition of words and their context. It basically assigns part of speech tags to words. For example, a token may be a verb or a noun. So, each token in a sentence is applied a tag. Predefined POS tag sets are used for POS tagging. For example, the most popular POS tag set for English is the Penn tag set [11]. For English, tag sets have 9 core parts of speech tags: noun, verb, article, adjective, adverb, interjection, preposition, pronoun, and conjunction. However, certainly, there are more categories and subcategories. For example, nouns have part of speech tags as plural, possessive, etc. Also, verbs have tense forms. Widely used POS tags of Penn tag set are given in Table 2.1.

It is useful to determine POS tags of words in a sentence for automatic term extraction task because grammatical properties of words give information about whether they are terms or not. For example, generally, terms are nouns or noun phrases. Therefore, detected POS

Table 2.1. POS tags in English

POS tag	Name	Examples
JJ	Adjective	old, green
NN	Noun	car, apple
NNs	Noun Plural	cars, apples
DT	Determiner	a/an, the
VB	Verb	visit, hide
IN	Preposition	of, on
CC	Conjunction	and, but, yet

tags can be used in the further steps for automatic term extraction.

Term Candidate Selection

Term candidate selection is the selection of term candidates according to predefined rules. These rules are categorized into two groups: linguistic and statistical. For instance, it can be the rule such that only nouns can be term candidates. Such rules are entirely generated based on language linguistic features. Also, POS tags which are determined in preprocessing step are used for applying linguistic rules. Naturally, the rules can be more complicated such that consider the combinations of POS tags. For this purpose, a regex that represents the accepted POS pattern can be used for filtering term candidates.

The other type of rules is statistical. Rare words can be filtered out. The minimum occurrence count is defined, and the term candidates are filtered out by using this predefined value. In addition, stop words, most common words in a language, are filtered out because they do not belong to terminology. The stop word removal is performed by using a predefined set of stop words.

Term Candidate Scoring

After term candidate selection, the next step is candidate scoring which is the most important step. The primary purpose of this step is to assign scores to term candidates. The score of a term candidate simply represents the possibility of being a term. Therefore, a higher score for a term candidate means higher term probability.

There are a variety of techniques for term candidates scoring. Generally, they depend on statistical methods. The statistical methods are given in the next subsection.

Term Candidate Ranking

The final step is the candidate ranking step which simply ranks by the calculated scores in the previous step and extracts the top predefined number of candidates. If only one term scoring method is used, then the candidates can be ranked by that score. Most of the studies use only one scoring method, and they rank the term candidates by the calculated score. In addition, some works use multiple scoring methods and they use linear combination of the multiple scores for ranking. Also, the multiple scores can be used as inputs for a supervised machine learning method [12].

2.1.2. Automatic Term Extraction Methods

Generally, most of the researchers use linguistic and statistical methods for extracting terms from text collections. Scoring methods of term candidates are generally based on statistical approaches. Their essential assumption that terms appear commonly in domain-specific texts [13, 14, 15]. On the other hand, modern approaches use linguistic methods only in the step of selection of term candidates. In addition, neural methods have also been used recently for term extraction. Thus, automatic term extraction methods are grouped as linguistic methods, statistical methods, and neural methods.

Linguistic Methods

Linguistic information is necessary for term extraction task as well as in other NLP tasks. The usage of linguistic methods for term extraction is based on defined rules. For example, terms are mostly noun phrases [16]. Linguistic methods attempt to identify word combinations which match certain syntactical and morphological patterns for identifying terms. For this purpose, morphological analyser and part-of-speech taggers are mostly used.

Linguistic features are totally language-dependent so that it is not possible to generate general rules for all languages. It requires knowledge of morphology and syntax information of the language. Also, word ambiguity, some words have more than one meaning and their meanings depend on the context, is a considerable problem for identification of terms. Although earlier studies applied only linguistic methods, using only linguistic features are not sufficient to identify terms. Generally, modern methods used linguistic features for term candidate selection only.

Statistical Methods

Statistical methods, as the name suggests, identify terms using statistical information. These methods are used for scoring term candidates. We can divide these methods into 3 groups:

- methods based on frequency,
- methods based on context,
- methods based on reference corpora,

Many methods perform terminology extraction by determining frequently used words and phrases from domain-specific texts. Methods based on term frequency contain Term frequency (TF), Average Term Frequency (ATF) [17], TF-IDF [15], Residual IDF (RIDF), C-Value [13], Basic [18], and ComboBasic [19].

Term Frequency is simply the measure of the number of occurrences of term candidates in the text. If term frequency is normalized by the number of documents containing term candidates, then we have ATF. TF-IDF stands for term frequency and inverse document frequency and it is often used for text mining. TF implies that if a word occurs multiple times in a document then it must be more meaningful than other words. However, if a word is a frequent word in all documents then it is not relevant and meaningful. So, IDF part of TF-IDF implies that so common words in documents must not be considered. TF-IDF is

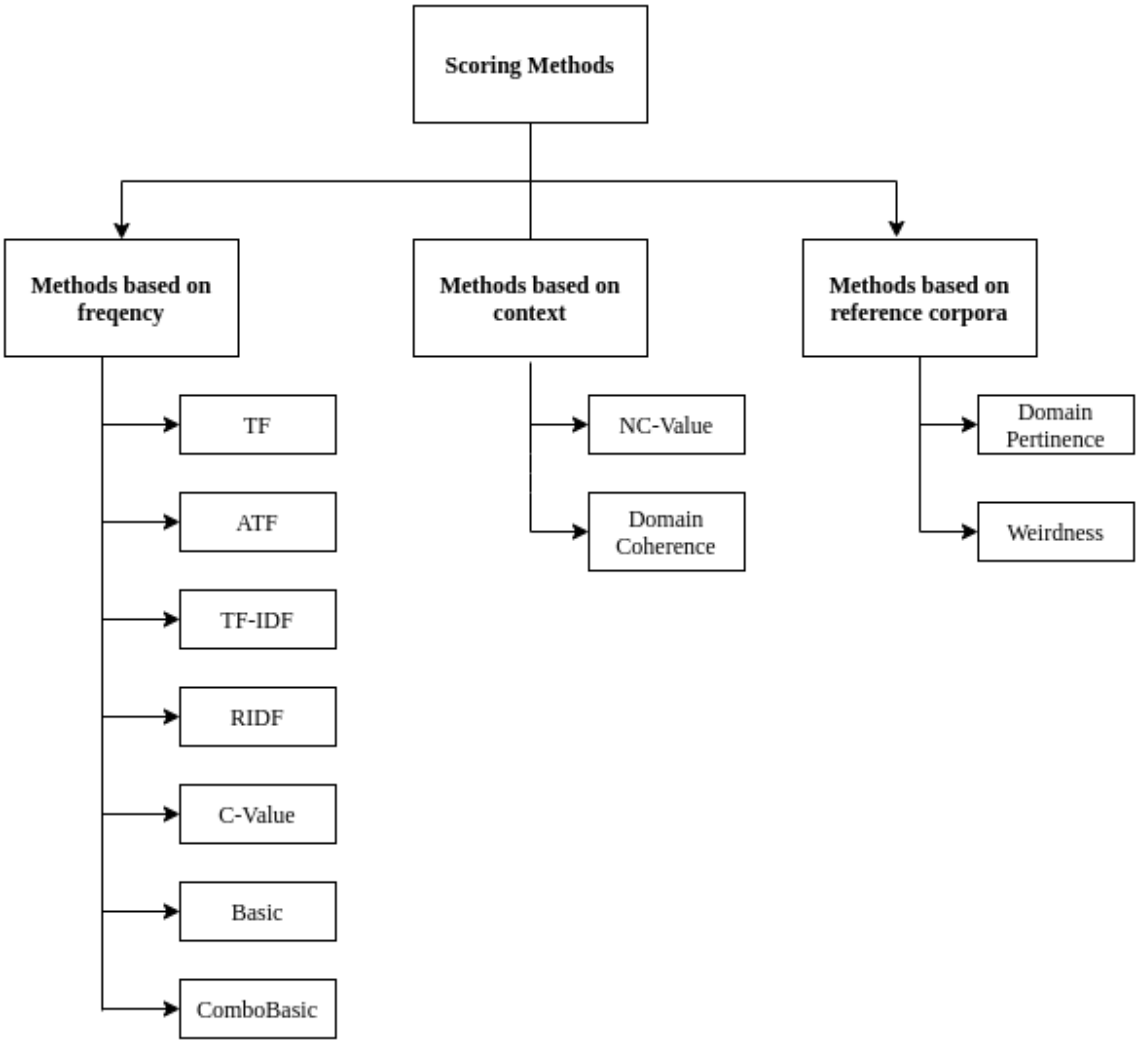

Fig. 2.2. Scoring methods

defined as :

$$TF\text{-}IDF_{t,d} = (tf_t) \cdot \log \frac{N}{df_t} \tag{2.1}$$

where N is a total number of document in the collection, tf_t is term frequency of term candidate t, and df_t is a number of documents containing term candidate t.

Residual IDF is an alternative to IDF and it is defined as the difference between the logs of document frequency and document frequency predicted by Poisson distribution. The other method based on occurrence frequency is C-Value [13]. It promotes term candidates

that have high frequency. At the same time, it promotes term candidates which are not as parts of other term candidates. So, it is sensitive to nested terms. Also, there are term scoring methods such as NC-Value [13] and DomainCoherence [18] which extract terms based on the importance of the context in which the terms are used.

Methods based on Reference Corpus Methods such as DomainPertinence [20] and Weirdness [14] compare general-purpose reference corpora with domain-specific corpora. Weirdness simply compares the frequency information of term candidates in general-purpose reference corpora with domain-specific corpora. The assumption is that terms are domain-specific so they occur frequently in domain-specific texts. In addition, a term candidate appears few in the reference corpora, the more likely it is to be a term. We can measure Weirdness scores of term candidates. The Weirdness score of a term is calculated as:

$$Weirdness(t) = \frac{NTF_{domain}(t)}{NTF_{reference}(t)} \tag{2.2}$$

where $NTF_{domain}(t)$ and $NTF_{reference}(t)$ are frequencies of the term which is normalized by the sizes of domain-specific and general-purpose reference corpora.

Neural Network Methods

As mentioned before, the general pipeline of term extraction consists of pre-processing, selection of term candidates, scoring of term candidates, and ranking of term candidates, respectively [10]. One of the problems of the traditional pipeline is that it has difficulty recognizing rare terms. In fact, words with less than a predefined occurrence frequency are generally eliminated in the term candidate selection step. The other problem is that linguistic features used in the filtering step are language-specific so that it is a problem to generalize linguistic methods for all languages. In order to solve these problems, deep learning methods are recently used for automatic term extraction. In TermEval 2020 shared task, a deep learning methodology achieves the highest scores [21]. The advantage of deep neural network models is that they learn linguistic features from data itself. There is no need to use a filtering step such as stop word removal, POS tag filtering, etc. Also, statistical features such as occurrence count are not necessary.

Recently, automatic term extraction task is treated as a sequence labeling problem with deep learning [22, 23, 24, 25]. Sequence labeling is the task of assigning categorical labels to each element in a sequence. Words are sequence elements for texts and assigning categorical labels to component words is the task of sequence labeling. Sequence labeling algorithms are generally based on statistical inference. Hidden Markov Model (HMM) [26] and conditional random fields (CRFs) [27] are two of the most common models used for sequence labeling. Deep learning methods are recently in use for sequence labeling task and effectively yielding the state-of-the-art performances for tasks such as named entity recognition, text chunking, etc. Also, there are some studies that use neural sequence labeling for automatic term extraction with deep learning. Generally, the studies use Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and Conditional Random Field (CRF) layers.

Kuzca et al. [22] built a monolingual term extraction system to identify terms with sequence labeling with BILOU scheme. They used Glove word embeddings [28] and worked with the following kinds of recurrent neural networks: Unidirectional LSTM, Unidirectional GRU, Bidirectional LSTM, Bidirectional GRU, Multilayer LSTM. They performed their experiments with GENIA [29] and ACL RD-TEC corpus [30] in English. In addition, adding a CRF [27] layer also improved the performance in some studies. For instance, a study proposed a BiLSTM-CRF model with Transformers (BERT) [24]. Their deep neural model is trained on Lithuanian Corpus in cybersecurity domain. Another study proposed a CNN-BiLSTM-CRF model and they used computer field papers in Chinese [23].

In short, there are some studies in the literature that propose a monolingual neural model for automatic term extraction task in different languages with sequence labeling. In this thesis, instead of a monolingual model, we propose a multilingual model with joint multilingual learning. Our method utilizes the sequence labeling and deep learning methods. The multilingual model we propose operates at a performance close to the monolingual model.

2.2. Statistical Machine Translation

Statistical Machine Translation learns how to translate by analyzing manually formed translations. SMT models are trained using a bilingual corpus. Given sentence f in source language, the goal is to find most likely translation e in target language. In statistical machine translation, alignment models are used to determine translations of the words and phrases in two sentences with same meaning in two languages. Alignment can be at sentence level or word level. Sentence alignment can be provided by the corpus or obtained by Gale-Church alignment algorithm [31]. However, in order to learn the translation model, there is need to know which words aligned with each other in source-target sentence pair. In a parallel text, the words in one language are aligned with the words in the other language. The example of word alignment is given in Figure 2.3. In order to align words, one of the solutions is the IBM-Models.

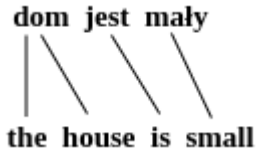

Fig. 2.3. Example of word alignment

2.2.1. Word Alignment Models

IBM alignment models [7] are widely used in Statistical Machine Translation to train an alignment model. There are five IBM alignment models. The sequence of the five models can be summarized as:

- Model 1: only uses lexical translation
- Model 2: additional absolute alignment model
- Model 3: adds fertility model
- Model 4: adds relative alignment model
- Model 5: fixes deficiency problem.

IBM Model 1 is based on only lexical translation. The problem of IBM Model 1 is the weakness of reordering, adding and dropping words. Generally, words in translations have a different order after translation. However, IBM Model 1 handles all types of reordering as equally as possible. IBM Model 2 adds a model of alignment to IBM Model 1. Therefore, after lexical translation step, there is one more step as the alignment step. IBM Model 3 adds fertility model to solve fertility problem. Fertility is the one of the problems while aligning words. Fertility problem means that some words can be translated to multiple words or translated no words at all. IBM Model 4 consider the surrounding words. Therefore, each word is dependent on the word that previously aligned. IBM Model 5 enhancing the alignment model in order to overcome deficiency [32].

2.2.2. Word Alignment Tools

There are some tools that implements IBM alignment models such as GIZA++ [33], FastAlign [34], etc. GIZA++ is an open-source statistical machine translation tool and it is used to train IBM Model 1, IBM Model 2, IBM Model 3, IBM Model 4 and IBM Model 5. Also, it uses an HMM word alignment model.

FastAlign is a simple, fast, unsupervised word aligner tool. It uses log-linear reparameterization of IBM Model 2 [35].

2.3. Word Embeddings

There are various approaches to represent words in NLP. As an example, one-hot encoding and bag-of-words models are widely used. However, they do not capture information about a word's meaning and context. In contrast, word embeddings consider surrounding words of a word so that it captures its meaning and context. Word embeddings are simply vector representations of words in a vector space. They are fundamental and commonly used for various NLP tasks. Each word is mapped to one vector and similar words have similar vector representations.

2.3.1. FastText

FastText is an open-source library for learning of word embeddings developed by Facebook in 2015. It is an extension of word2vec model but it treats each word as a composed of character n-grams. This allows to build vectors even for misspelled words or concatenation of words. Since it consider sub-words, it also generates words embeddings for rare words. Furthermore, it can produce the vectors of out of vocabulary words. Even if a word doesn't appear in corpus, it can construct the vector of the unknown word from its character n-grams.

It is possible to use the FastText library for learning vector representations of words from any dataset. In addition, they distribute pre-trained multilingual word vectors for 157 different languages which are trained on Common Crawl and Wikipedia [36]. They trained these models using CBOW with position-weights. The dimension of these models are 300. Further, they published aligned word vectors for 44 languages. These aligned word vectors are based on pre-trained vectors. These pre-trained vecters are computed on Wikipedia [37, 38]. RCSLS method [39] is used to align word vectors.

2.4. Deep Recurrent Neural Networks

Long short-term memory networks (LSTM) and its variations are broadly used neural network architecture in the field of deep learning. Since LSTM networks are well-suited to time series data, they are used for various NLP tasks such as keyword extraction, language modeling, sequence labeling, etc. In this section, LSTM and BILSTM networks are explained.

2.4.1. Long Short-Term Memory Networks

Long short-term memory (LSTM) is an architecture [40] used in deep learning. It was proposed by Sepp Hochreiter and Jürgen Schmidhuber. It is basically an artificial recurrent neural network (RNN) architecture. It is a speacial kind of RNN architecture such that it solves the vanishing gradient problem of RNNs.

An LSTM unit consists of a cell, an input gate, a forget gate and an output gate. The structure of a LSTM unit is given in Figure 2.4. The equations for output gate, output gate and forget gate are given respectively in following equations.

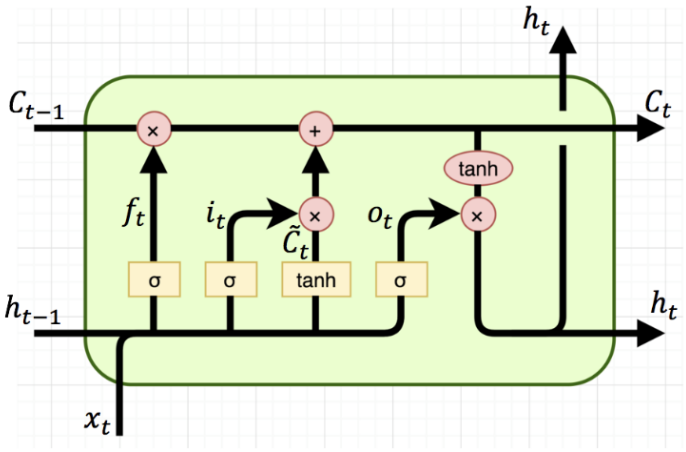

Fig. 2.4. LSTM unit

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{2.3}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2.4}$$

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{2.5}$$

where σ is the sigmoid function, w_x is weight for the gate neurons, h_{t-1} is the output of the previous lstm block, x_t is the input at current timestamp, and b_x is the biases for gates.

The forget gate decides which information needs and which can be ignored. The information from the current input x_t and hidden state h_{t-1} are passed through the sigmoid function and sigmoid fuction generates values between 0 and 1 (Equation 2.4).

The input gate of an LSTM unit determines what new information will be stored in the cell state. The input gate performs the following operations to update the cell status. Firstly, the current state x_t and previously hidden state h(t-1) are passed into the second sigmoid function (Equation 2.3). So, the values are transformed between 0 and 1. Then, the tanh

function creates a vector $\tilde{c_t}$ with all the possible values between -1 and 1. The equation for $\tilde{c_t}$ is given in Equation 2.6.

$$\tilde{c_t} = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.6}$$

It is enough to form new cell state using the information from the forget gate and also from the input gate. Here, the previous cell state c_{t-1} multiplied with forget vector f_t. The outcome can be 0, then values are released in the cell state. Then, the output value of the input vector i_t is taken and performs point by point addition, so the cell state is updated. The new cell state c_t is given in Equation 2.7.

$$c_t = f_t * c_{t-1} + i_t * \tilde{c_t} \tag{2.7}$$

The function of the output gate is to determine the value of the next hidden state. Here, next hidden state contains information about previous inputs.Firstly, the values of the current state and previous hidden state are passed into sigmoid function (Equation 2.5. Then the new cell state generated from the cell state is passed through the tanh function. So, the output of the LSTM at time t is given in following equation.

$$h_t = o_t * tanh(c_t) \tag{2.8}$$

To sum up, the forget gate determines which information from the prior steps is necessary. The input gate decides what information can be added using the current step, and the output gates determines the next hidden state.

2.4.2. Bidirectional Long Short Term Memory Networks

Bi-directional LSTM (BiLSTM) is the bidirectional version of LSTM. Here, bidirectional means the signal propagates backward as well as forward in time. Bidirectional recurrent neural networks basically consist of two independent LSTMs together. In this way, it allows the networks to have both backward and forward information about the sequence. The transmission of information is from the past to the future and also from the future

to the past. Thefore, BiLSTMs can understand the context better than LSTMs. BiLSTM is actually two separate LSTM network layers. Two layers of the BiLSTM generate two different context vectors for each time, and then these vectors are joined to create the final vector.

2.5. Sequence Labeling

Sequence labeling is the operation of assigning categorical labels to each element in a sequence. Words are sequence elements for a sentence and assigning categorical labels to words is a task of sequence labeling. POS tagging is a common example of sequence labeling task. POS tagging operation for an example sentence is given in the Figure 2.5.

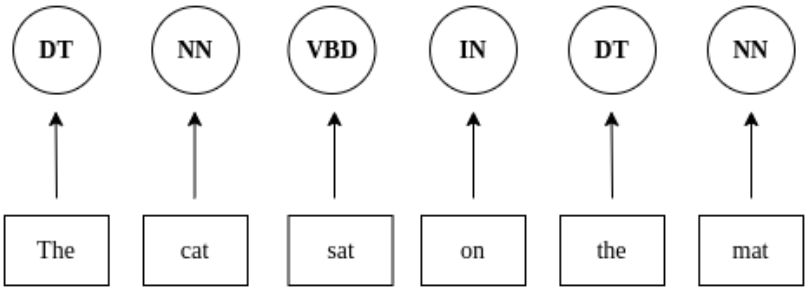

Fig. 2.5. POS tagging

Sequence labeling algorithms are generally based on statistical inference. Hidden Markov Model (HMM) [26] is one of the most common models used for sequence labeling. Also, conditional random fields is in use. Recently, deep learning methods are in use for sequence labeling task and effectively yielding the state-of-the-art performances for tasks such as named entity recognition, text chunking.

2.6. Conditional Random Fields

Conditional Random Fields is a discriminative sequence labeling model which is used for predicting sequences. They use contextual information and it considers the state of the neighbors in the context. To consider neighbors, the prediction is modeled as a graphical model, which implements dependencies between the predictions. CRF has NLP applications such that part of speech tagging, named entity recognition etc.

Lafferty, McCallum and Pereira [27] define a CRF on observations X and random variables Y as follows: It assumes $G = (V, E)$ be a graph and t $Y = (Y_v)_{v \in V}$. Then, (X, Y) is a conditional random field when the random variables Y_v, conditioned on X, obey the Markov property with respect to the graph: $p(Y_v|X, Y_w, w \neq v) = p(Y_v|X, Y_w, w \sim v)$. Here, $w \sim v$ means that w and v are neighbors in the graph G.

2.7. Joint Multilingual Learning

Natural language processing (NLP) techniques have several different layers such that morphological analysis, syntactic analysis, semantic analysis, etc. Languages with low resources suffer from the inability to applied standard NLP techniques because NLP techniques require linguistic knowledge developed by experts. Also, a lot of labeled data is required for NLP techniques and it is expensive to generate labels.

Joint multilingual learning is an approach to enable to use NLP techniques for low-resource languages. In this approach, a single multilingual model is trained on multilingual dataset. In order to share data and parameters, multilingual embeddings are used. The word vectors of different languages share the same word vector space. If the meaning of two words in different languages are similar, their word vectors are close to each other in the vector space. For example, a shared word embedding space between Turkish and English is given in Figure 2.6.

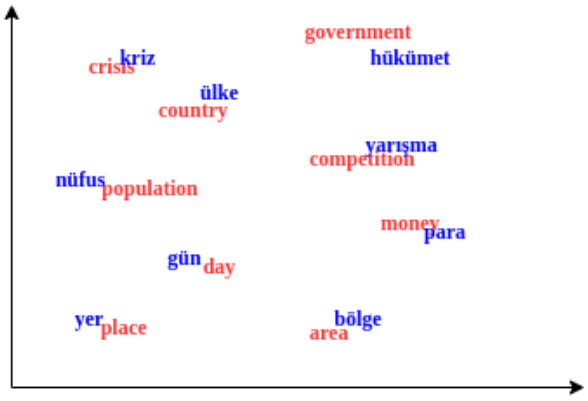

Fig. 2.6. A shared embedding space between Turkish and English

It is possible to learn a linear transformation from one language to another with using dictionaries and monolingual embeddings [41]. In order to learn a shared embedding

space for words in multiple languages, the resources can be :

- Dictionaries
- Word-level aligned data
- Sentence-level aligned data
- Document-level aligned data

There are supervised [37] and unsupervised [42, 43] embedding alignment studies in literature. For example, bilingual dictionaries and monolingual embeddings can be used for learning bilingual mappings [41, 44]. Also, small seed bilingual dictionary can be used in unsupervised methods [45].

3. DATASET

English and Turkish thesis abstracts are collected from YÖK Thesis Center [1] which is a domain-specific bilingual source that includes theses in different scientific disciplines. It includes an English and a Turkish abstract for each thesis.

3.1. Cleaning

Our input texts, thesis abstracts, had accented characters such as the acute (é), grave (è), and circumflex (â, î). They are converted to ASCII format characters. Since we aim to obtain parallel sentences, we choose theses that do not have inconsistency in the number of sentences in the Turkish abstracts and English abstracts of them. Naturally, the number of sentences may not be exactly the same depending on the language. However, we limit the sentence number difference to take up to 5 sentences with the purpose of improving quality of our system.

3.2. Grouping Thesis Abstracts

We grouped the thesis abstracts into three scientific groups using taxonomy of scientific fields: LIF (Life Sciences), PHY (Physical Sciences and Engineering), SOC (Social and Behavioral Sciences) [46]. The numbers of thesis abstracts for each scientific field are indicated in Table 3.1.

3.3. Parallel Corpora Construction

In most of the thesis abstracts, the sentences are parallel. Even so, we apply sentence alignment methods in order to improve translation quality. Sentence alignment is used to identify correspondences between sentences of bilingual sources. Thus, it enables constructing parallel corpora to be obtained from bilingual sources. In this work, LFAligner tool [2], one of the sentence alignment tools, is used to align sentences of our bilingual source

[1] YÖK Tez, (https://tez.yok.gov.tr/UlusalTezMerkezi)
[2] LFAligner, (https://sourceforge.net/p/aligner/wiki)

Table 3.1. Scientific Fields

Scientific Field	Scientific Group	Abstract Count
Biology	LIF	7267
Food Science	LIF	3807
Agriculture	LIF	13243
Electrical and Electronics Engineering	PHY	10686
Chemistry	PHY	14730
Computer Science	PHY	7145
Economics	SOC	7071
History	SOC	5679
Business Administration / Management	SOC	17611

i.e. thesis abstracts. LFAligner is simply based on Hunalign [47] and it uses Gale-Church algorithm which is based on sentence length information [31] in order to align sentences. The number of sentences for each scientific field and language are given in Table 3.2.

Table 3.2. The number of sentences for each scientific field

Scientific Field	Language	Sentences
Biology	TR EN	70767 73839
Food Science	TR EN	42800 44810
Agriculture	TR EN	57025 56923
Electrical and Electronics Engineering	TR EN	102933 105460
Chemistry	TR EN	142533 147008
Computer Science	TR EN	68969 69813
Economics	TR EN	34255 34151
History	TR EN	47452 46154
Business Administration / Management	TR EN	156349 158149

4. BILINGUAL AUTOMATIC TERM EXTRACTION

Bilingual Automatic Term Extraction is the automatic extraction of bilingual term pairs from both source and target languages. In this study, the steps for this task has following two steps:

- Monolingual term extraction for English.
- Terminology translation phase where Turkish translation candidates of extracted English terms are detected.

The above-mentioned steps are performed separately for each scientific field. Therefore, we aim to obtain separate terminology for each scientific field.

4.1. Automatic Terminology Extraction

In this work, automatic term extraction was performed in English. The pipeline of our work consists of four steps:

- Preprocessing
- Term candidates selection
- Term candidates scoring with Weirdness method
- Term candidates ranking

respectively, as shown in Figure 4.1. To perform these steps, we used ATR4S library [10].

4.1.1. Preprocessing

Preproccessing step is the first step in our automatic term extraction pipeline. In the preprocessing step, the following preprocessing tasks are performed, respectively:

- Sentence Segmentation: It is the operation of splitting texts into sentences. Our input texts are segmented into sentences by determining sentence boundaries using

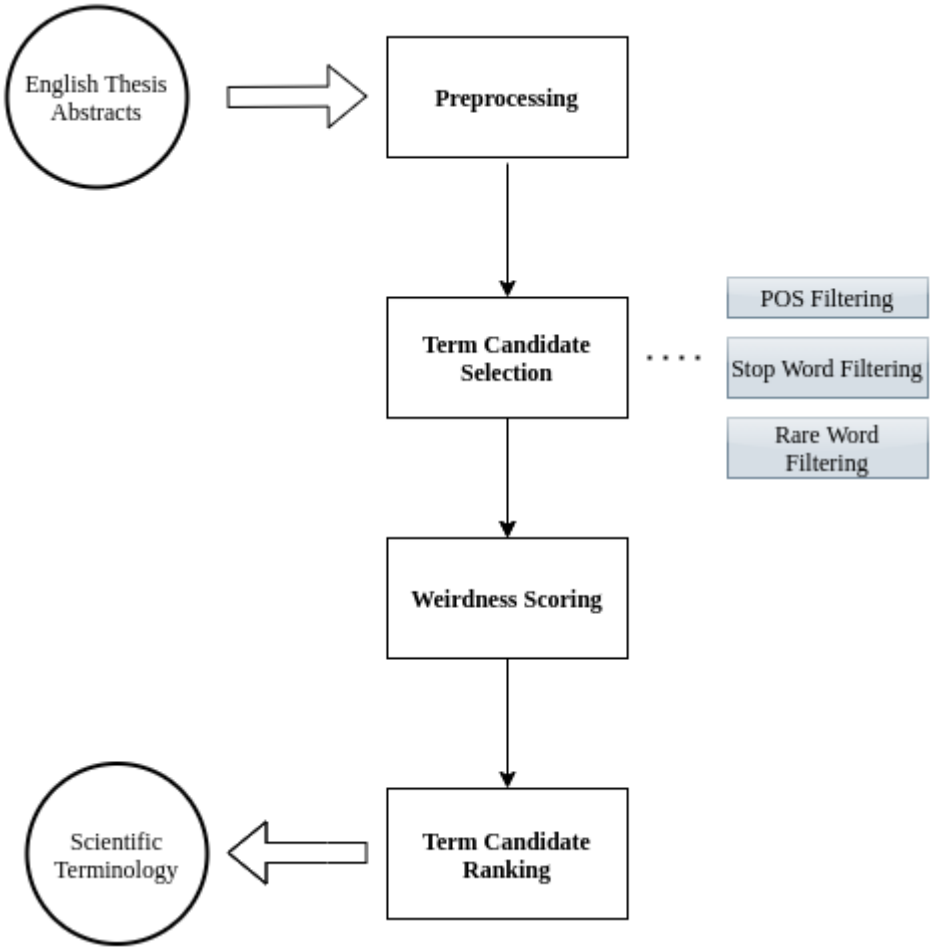

Fig. 4.1. The pipeline of term extraction task

punctuations.

- Tokenization (Word Segmentation): It is basically the operation of splitting sentences into their component words. Our sentences are tokenized into words by white-space characters. Then, punctuations are removed from tokens, and all tokens are converted to lower case in order to normalize words into the same form in case. Word segmentation and case-folding operation for an example sentence is given in the Figure 4.2.

- POS tagging : POS tags are determined for each word in sentences. POS tags of the tokens of the example sentence are given in Table 4.1.

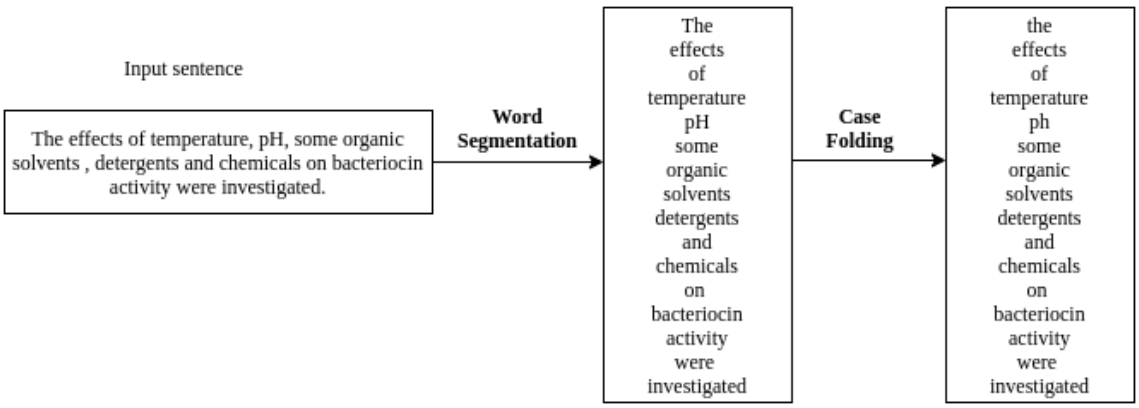

Fig. 4.2. Word Segmentation and Case Folding

Table 4.1. POS tags of the tokens of an example sentence

Token	POS tag
the	Determiner, DT
effects	Noun, NNS
of	Preposition , IN
temperature	Noun, NN
ph	Noun, NNP
some	Determiner, DT
organic	Adjective, JJ
solvents	Noun, NNS
detergents	Noun, NNS
and	Conjunction, CC
chemicals	Noun, NNS
on	Preposition , IN
bacteriocin	Noun, NN
activity	Noun, NN
were	Verb, VBD
investigated	Verb, VBN

4.1.2. Term Candidate Selection

The next step of the pipeline is term candidates selection step. Firstly, n-grams are extracted. N-gram extraction is the operation of extraction of consecutive words from text. In this study, we extracted unigrams, bigrams, trigrams, and four-grams. 2-gram extraction of an example sentence is given in the Figure 4.3.

After n-gram extraction, following filtering operations are performed:

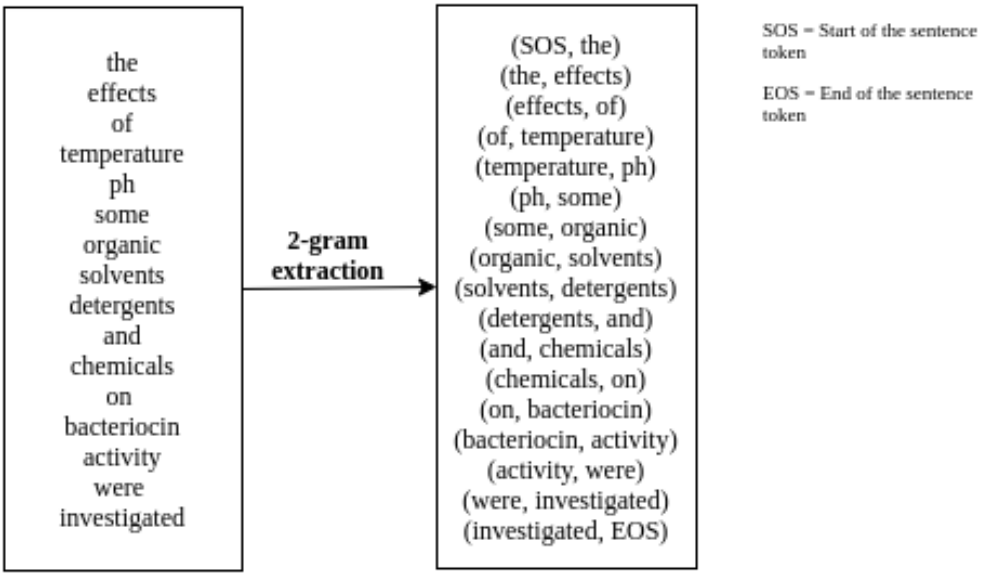

Fig. 4.3. 2-gram extraction.

- **Stop Words Removal:** Most common words such as “a”,”the” etc. are filtered out in order to reduce noise. For this purpose, a predefined stop word list for English is used.

- **Occurence Filter:** Since we aim to reduce noise, candidates occurring rarer than 100 times are filtered out.

- **POS Tags Filter:** Only candidates that match the following POS tags pattern are accepted:

$$(NN(S)?|JJ|NNP|NN(S?)IN) * (NN(S)?) \tag{4.1}$$

A few example of accepted and not expected term candidates are given in the Table 4.2.

Table 4.2. Examples of accepted and not expected term candidates

Candidate	**POS tags**	**Status**
bacteriocin activity	NN NN	Accepted
were investigated	VBD VBN	Not Accepted
chemicals on	NN IN	Not Accepted
effects of temperature	NNS IN NN	Accepted
organic solvents	JJ NNS	Accepted

4.1.3. Term Candidate Scoring

In the next step, we use Weirdness scoring method [14] for term candidate scoring. As we already mentioned in Section 2, Weirdness simply compares the occurrence frequency of words and word groups in general-purpose reference corpora with those in domain-specific corpora. Terms are domain-specific so that terms occur frequently in domain-specific texts. In addition, the less a word or phrase appears in the reference corpora, the more likely it is to be a term. If these two ideas are combined, we can measure weirdness of term candidates. In short, the more frequently a word appears in domain-specific corpora, and the less it occurs in reference corpora, the higher the Weirdness score. The Weirdness score of a term is calculated as:

$$Weirdness(t) = \frac{NTF_{domain}(t)}{NTF_{reference}(t)} \tag{4.2}$$

where $NTF_{domain}(t)$ and $NTF_{reference}(t)$ are frequencies of the term which is normalized by the sizes of domain-specific and general-purpose reference corpora.

Randomly selected 10 term candidates in the field of Computer Science and their Weirdness scores are given in Table 4.3.

Table 4.3. Term candidates and their Weirdness scores

Term candidate	**Weirdness Score**
data mining	0.394
decision tree	0.106
computing	0.293
cloud	0.137
web service	0.206
management system	0.200
authentication	0.185
social network	0.119
semantic web	0.101
data	4.737

4.1.4. Term Candidate Ranking

The final step is the ranking step of term candidates. We rank term candidates by Weirdness score. Then, we take the top 2000 candidates to construct terminology. Top 10 term candidates sorted by Weirdness score in Computer Science field are given in Table 4.4

Table 4.4. Top 10 term candidates in Computer Science

Term candidate	Weirdness Score
system	5.481
data	4.737
method	4.024
algorithm	3.856
network	3.140
application	2.999
model	2.647
information	2.189
software	2.143
user	2.071

4.2. Terminology Translation

The second step of bilingual term extraction is extracting Turkish translations for extracted English terms. This step consists of following substeps:

- Alignment of words
- Generation of translation dictionaries
- Operations on translation candidates

Above mentioned steps are performed to generate scientific terminology for each scientific field.

4.2.1. Word Alignment

We apply statistical machine translation methods on our bilingual source at word level in order to obtain translation candidates. Word alignment is the problem of matching

translational correspondences at the word level for parallel sentences. By using sentences aligned at the previous phase, we used FastAlign tool [34] that is an unsupervised word aligner using log-linear reparameterization of IBM Model 2 [35]. It should be noted that IBM Model 2 is an extension to IBM Model 1 such that IBM Model 2 adds a model of alignment to IBM Model 1.

By using FastAlign tool on our bilingual source, both target-source (Turkish - English) and source-target (English-Turkish) alignments are generated and these alignments are symmetrized with the help of the tool which applies a variety of symmetrization heuristics.

4.2.2. Generation of Translation Dictionaries

After word alignment phase, the next phase is generation of translation dictionaries. The purpose of this step is to obtain Turkish translation candidates for each recognized English term. In this step, we form separate dictionaries for 1-grams, 2-grams, 3-grams, and 4-grams. These dictionaries have terms and their translation candidates.

It is a fact that word alignments may be one-to-one, where each target word is aligned with one source word. In this situation, a term has simply the exact matching translation candidate. In addition, one-to-many alignments are possible where the target word is aligned with more source words. We concatenate consecutive alignments for one-to-many alignments.

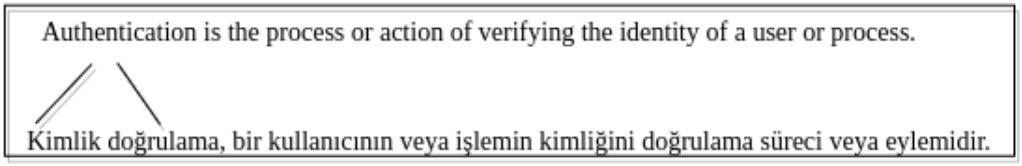

Fig. 4.4. Example of one-to-many alignment

For example, assume a term ”authentication” is aligned with multiple words ”kimlik” and ”doğrulama”. Then, if these alignments are consecutive, then they are concatenated. Thus, ”authentication” term has a translation candidate as ”kimlik doğrulama”. The illustration of the example is given in 4.4.

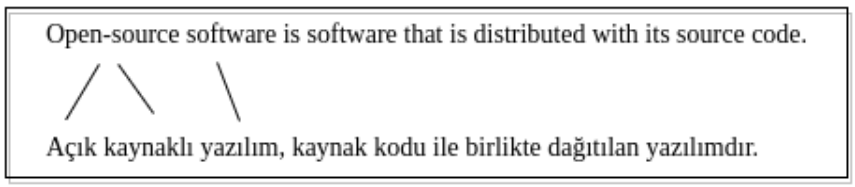

Fig. 4.5. Example of consecutive alignments

Also, if word alignments of 2-gram, 3-gram, and 4-gram target words are consecutive, they are concatenated. As an example, assume that a sentence has the term ”open-source software”. If the word ”open-source” is aligned with ”açık kaynak” and the word ”software” is aligned with ”yazılımı”. Then, since we concatenate consecutive words, entry in the translation dictionary becomes ”open-source software : açık kaynak yazılımı”. The illustration of the example is given in 4.5.

We generate the translation dictionaries for each scientific field separately, by considering the conditions mentioned above. It should also be noted that dictionaries we formed also have occurrence information. Therefore, if a term has multiple translations, then we know about frequency information that indicates how many times the translation candidate occurs.

4.2.3. Operations

After generation of translation dictionaries for each scientific field, we have translation candidates in Turkish for each recognized English term. However, it is not optimal to use translation candidates directly because of the noise factor. For this reason, the following operations are applied on the translation candidates:

Filtering operations

- **Stop Word Removal:** Most common words such as “bir”,”ve” etc. are eliminated. For this purpose, a predefined stop word list for Turkish is used.
- **Occurence Filter:** We only keep translation candidates that occurred more than 2 times.

- **Pos Tag Filtering:** We obtain POS tags by using Zemberek framework [48]. Then, POS tags filtering is applied such that only candidates that match the following POS tags pattern are accepted:

$$NN|((NN|JJ)+(IN)?(JJ)*NN) \tag{4.3}$$

A few examples of term candidates which are accepted or not accepted are given in Table 4.5.

Table 4.5. Sample term candidates with POS tag filtering

Candidate	**POS tags**	**Status**
sistem	NN	Accepted
doğrulanmak	VB	Not Accepted
çoklu	JJ	Not Accepted
sistem tasarımı	NN NN	Accepted
geliştirilen sistem	JJ NN	Accepted
hacimli veri iletişimi	JJ NN NN	Accepted
ile tanımlanmıştır	CC VB	Not Accepted

Inflectional Suffix Removal

Since Turkish is an agglutinative language, translation candidates may have suffixes. For example, the term ”bilgisayar” can occur with suffixed forms as shown in Table 4.6.

Table 4.6. Example suffixes of the word ”bilgisayar”

Suffixed Form	**Suffix**
bilgisayar**lar**	Plural Suffix
bilgisayar**ı**	Accusative Case Ending
bilgisayar**ın**	Genitive Case Ending
bilgisayar**a**	Dative Case Ending
bilgisayar**dan**	Ablative Case Ending
bilgisayar**ların**	Plural Suffix + Genitive Case Ending

We analyze the morphology of translation candidates by using Zemberek framework [48]. As much as possible, we remove noun inflectional suffixes such as plural, case endings, suffix of possession, etc. The examples of the forms of the words after suffix removal are given in Table 4.7.

Table 4.7. Examples of suffix removal

Word with Suffix	**After Suffix Removal**
sistem**i**	sistem
platform**u**	platform
destek sistemi**nin**	destek sistemi
destek sistemi ayrıntıları**yla**	destek sistemi ayrıntıları
haberleşme cihazı**nı**	haberleşme cihazı
işletim dizgesi**nde**	işletim dizgesi

4.3. Scientific Terminology

After monolingual term extraction in English and terminology translation phase, we acquired terms in English and their translations in Turkish. Consequently, we finally obtained scientific terminology for each scientific field. Several terms and their translations in Electrical and Electronics Engineering field are given in the Table 4.8. Several terms and their translations in Computer Science are given in the Table 4.9.

Table 4.8. Sample Electrical and Electronics Engineering terms and translations

Term	**Translations**
sampling	örneklem
power line	güç hattı, enerji hattı
motion estimation	hareket kestirimi, hareket tahmini
beamforming	huzme şekillendirme, huzme oluşturma
integrated circuit	tümdevre, entegre devre, tümleşik devre
detection algorithm	bulma algoritması, tespit algoritması, belirleme algoritması

As seen from the table, a term can have one or multiple translations. Also, it should be noted that the same term may have different translations for different scientific fields because the data is different for each scientific field and we formed translation dictionaries separately for each scientific field.

Table 4.9. Sample Computer Science terms and translations

Term	**Translations**
authentication	kimlik doğrulama
support vector machine	destek vektör makinesi
text processing	metin işleme
relational database	ilişkisel veritabanı
artificial neural network	yapay sinir ağı
cloud computing	bulut bilişim
binary search	ikili arama
sentiment analysis	duygu analizi
natural language processing	doğal dil işleme

5. MEASURING TERMINOLOGY INCONSISTENCY

5.1. Terminology Inconsistency Metric

We use Shannon Entropy [49] intending to measure terminology inconsistency. We assume each term as a random variable. If a term T has N translations, each of translations has a probability $p(n_i)$. Formally,

$$p(n_i) = \frac{Number\ of\ Translations\ as\ n_i}{Total\ Number\ of\ Translations} \tag{5.1}$$

where $\sum_{i=1}^{N} p(n_i) = 1$.

The translation entropy of a term is a measure of inconsistency in translations of a term. Using the probability distribution, the translation entropy for a term can be calculated as :

$$E(T) = -\sum_{i=1}^{N} p(n_i) \log p(n_i) \tag{5.2}$$

It is obvious that if a term has only one translation, then the translation entropy of it is zero.

Since we aim to measure terminology inconsistency for a scientific field, we need to join the entropies of all terms in the scientific field. We use the Joint Entropy in order to measure the uncertainty for the set of terms. Since translation diversity of each term is independent of each other, we can calculate Joint Entropy for a scientific field as the average of individual entropies of the terms in that scientific field. Assume S is a scientific field, and it contains K terms. Then the entropy of S can be calculated as:

$$J(S) = \sum_{i=1}^{K} \frac{E(T_i)}{K} \tag{5.3}$$

When we calculate entropy for a scientific group, we can use the same Joint Entropy metric. If G is a scientific group which has W scientific fields, then the entropy of G can

be calculated as:

$$J(G) = \sum_{i=1}^{W} \frac{E(T_i)}{W} \tag{5.4}$$

5.2. Results

We measured translation inconsistency for scientific fields and groups by our metric. The results are given in Table 5.1.

Table 5.1. Inconsistency values of scientific fields

Scientific Field	Scientific Group	Term Count	Inconsistency	Group Inconsistency
Biology	LIF	854	0.389	0.403
Food Science		707	0.391	
Agriculture		1599	0.428	
Electrical/Electronics Engineering	PHY	1282	0.475	0.432
Chemistry		1800	0.377	
Computer Science		856	0.446	
Economics	SOC	832	0.412	0.414
History		539	0.425	
Business Administration/Management		1668	0.407	

As expected, in the fields of Computer Science and Electrical and Electronics Engineering, where new terms are constantly added, the inconsistency of terminology is higher than the others. On the other hand, the most consistent scientific field in terminology is Chemistry. It is not surprising that the terminology of Chemistry is consistent, as it has a history dating back to ancient times.

According to the results, the terminology of LIF (Life Sciences) is more consistent than SOC (Social and Behavioral Sciences) and PHY (Physical Sciences and Engineering). On the other hand, PHY is the most inconsistent scientific group in terminology according to our results. If we sort scientific groups by terminology inconsistency, it is as follows:

$PHY > SOC > LIF$. It makes sense because it is natural that terminology is inconsistent in relatively new disciplines.

We conduct a survey to verify that our metric is a valid metric for measuring translation inconsistency. The survey consists of randomly selected 50 terms from three scientific field: History, Computer Science and Food Science. It should be noted that the survey for each scientific field is conducted separately. For each English scientific term, we have 4 options such that 3 Turkish translations found by the system and the Other option. We asked people to choose the translation they most prefer to use in the academic language for each scientific term.

We measure Fleiss' kappa [50] for each scientific field. In short, it is a way to measure agreement between raters. It ranges from 0 to 1. 0 means no agreement and 1 is perfect agreement. Interpretation of kappa ranges is given in the Table 5.2.

Table 5.2. Interpretation of kappa ranges

Fleiss' kappa	**Interpretation**
<0	Poor agreement
0.01 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 1.00	Almost perfect agreement

The calculated kappa values for each scientific field are given in the Table 5.3. According to kappa scores for scientific fields, Food science has a higher agreement rate than others. If we sort scientific fields by agreement rate, it is as follows: Food Science > History > Computer Science. It makes sense because Computer Science is a relatively new discipline than others. As a result, survey results support our study such that our metric is a valid metric for measuring terminology inconsistency.

Table 5.3. Fleiss' kappa scores for scientific fields

Scientific Field	**Fleiss' kappa**	**Interpretation**
Computer Science	0.57	Moderate aggreement
Food Science	0.61	Substantial aggrement
History	0.58	Moderate aggreement

6. TERM EXTRACTION WITH JOINT LEARNING

In this part of the thesis, automatic term extraction task is treated as a sequence labeling problem. We utilize joint multilingual learning. Joint multilingual learning enables to train a single model with mixed data in languages with rich resources and low resources. We train a single multilingual model on a mixed dataset in English and Turkish. In this way, it enables data and model parameter sharing.

We collect thesis abstracts from YÖK Thesis Center [1]. The steps in our pipeline are as follows:

- Preprocessing step on dataset.
- Manually labeling our dataset by using terminology dictionaries.
- Train deep multilingual neural models with training data of four domains. For comparison, we also train monolingual models for each domain.
- Tag decoding operation is performed such that converting predicted tags to terms.
- Finally, test the models on test data and evaluate the models.

6.1. Dataset

English and Turkish thesis abstracts are collected from YÖK Thesis Center which is a domain-specific bilingual source that includes theses in different scientific disciplines. It includes an English and a Turkish abstract for each thesis. In this study, without loss of generality we utilize data from four scientific fields: Chemistry, Food Science, Computer Science, and Electronic Engineering.

We randomly select Turkish sentences up to 40,000 and English sentences for up to 20,000 for different training experiments for each domain. For validation, 10% of training sentences are used. For testing, we select 4000 Turkish sentences for each domain.

[1] YÖK Tez, Ulusal Tez Merkezi [online] Website https://tez.yok.gov.tr/UlusalTezMerkezi [accessed 05 2021]

Since each thesis has an abstract in English and an abstract in Turkish, we have parallel texts. While selecting sentences, we ensured that the sentences belonging to the same thesis are included in the same set (training set, validation set or test set) so that no bias occurs.

6.2. Preprocessing

In the preprocessing step, the following tasks are performed, respectively: sentence segmentation, word segmentation, padding operation, and language suffix addition. First, our input texts are segmented into sentences by determining sentence boundaries using punctuations. Then, the sentences are segmented into words such that splitting sentences into their component words. The sentences are tokenized into words by white-space characters. Then, punctuations are removed from tokens. Since some terms can have the hyphen (-) to join words, we do not remove hyphen marks from tokens. Then, all tokens are converted to lower case in order to normalize words into the same form in case.

When sentence and word segmentation are performed, we have a list of words for each sentence. Then, we perform the padding operation. Sequences are padded with a special token to match the maximum sentence length. Then, we added $< SOS >$ tag to the beginning of the sentences and $< EOS >$ tag to the end of the sentences in order to indicate the beginning and end of the sentence. In addition, we added language suffixes $< TR >$ and $< EN >$ at the end of the words to avoid confusing the words written in the same way in both Turkish and English.

6.3. Terminology Labeling

We need to have labeled data for supervised learning. Our dataset does not include annotated terms. Therefore, we need to annotate terms on both Turkish and English data manually by using a terminology dictionary. We used TÜBA Turkish Science Terms Dictionary [2] as the gold term source. TÜBA dictionary contains both Turkish and English terms for various scientific fields. The number of gold terms for each domain used in this

[2]TÜBA Turkish Science Terms Dictionary [online] Website http://www.tubaterim.gov.tr [accessed 04 2021]

study is given in Table 6.1. It should be noted that the reason for the difference between the number of English terms and the number of Turkish terms in the dictionary is that some terms have more than one translation.

Table 6.1. The number of gold terms for scientific fields

Domain	**Language**	**Gold Term Count**
Chemistry	TR	2566
	EN	3124
Food Science	TR	1552
	EN	2077
Computer Science	TR	2853
	EN	3831
Electronic Engineering	TR	3422
	EN	4673

We marked up tokens of the dataset using BILOU tag scheme [51]. BILOU tag scheme stands for : (B)eginning, (I)nside, (L)ast, (O)utside, (U)nit. Our usage format of BILOU tag scheme is given in Table 6.2. To clarify, single-word terms are marked with a ”U” tag. For multiple-word terms, it starts with a ”B” tag and continues for each inside token with an ”I” tag, then ends with an ”L” tag. For example, 2-word terms are annotated with ”B , L”. For 3-word terms are annotated with ”B , I , L”. Non-terms tokens are annotated with the ”O” tag. Also, $< SOS >$, $< EOS >$ and padding tokens are annotated with the ”O” tag because they are non-terms. Also, it should be noted that a term is labeled in all sentences where it appears so that a term can be labeled more than once.

Table 6.2. Term annotation schema

Tag	**Annotation Format**
B	The first token of a term
I	An inner token of a term
L	The final token of a term
O	A non-term token
U	A single-token term

Since a word can only be assigned one label at a time, we do not handle nested terms such that subsets of the multi-word terms are also terms. For example, let the dictionary has the terms *”bilgi”*, *”bilgi işleme”*, and *”bilgi işleme tekniği”*. If the term *”bilgi işleme tekniği”* appears in the text, we annotate it with ”B, I, L”. Here, we do not handle subsets of the term, we ignore them because one label can be assigned to a word at a time.

In addition, we do not annotate the inflectional forms of the terms. We label the exact cases as in the terminology dictionary. For instance, we do not annotate the words *'bilgisayarı'*, *"bilgisayarın'*, and *"bilgisayarlar"* since they have a suffix. On the other hand, we annotate the word *"bilgisayar"* as a single-token term with a "U" label.

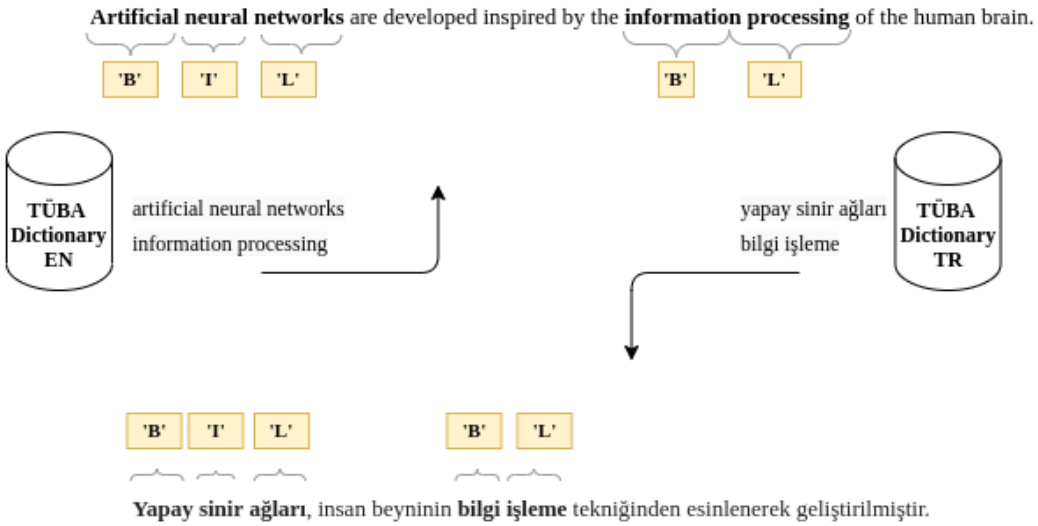

Fig. 6.1. Example sentences for terminology labeling

The example sentences of terminology labeling operation are given in Figure 6.1. In the English sentence, the term *"artificial neural networks"* is labeled with "B, I, L" and the term *"information processing"* is labeled with "B, L". In the Turkish sentence, the term *"yapay sinir ağları"* are labeled with "B, I, L" and the term *"bilgi işleme"* is labeled with "B, L". Remaining words that are not found in the dictionary are tagged with the 'O' tag, but for simplicity, the figure does not include these tags.

Terminology labeling operation is performed for each domain with different data sizes. The total numbers of annotated terms for each data size and each domain are given in Table 6.3. It should be noted that the values in the table are the numbers of annotated unique terms.

6.4. Model

We trained two kinds of deep neural network models for each domain dataset: monolingual and multilingual. Monolingual models are trained with only Turkish data. Multilingual

Table 6.3. The numbers of annotated terms

Domain	Language	Sentences	Annotated Terms
Chemistry	TR	20,000	995
		25,000	1048
		30,000	1083
		35,000	1109
		40,000	1141
	EN	5,000	962
		10,000	1200
		15,000	1337
		20,000	1438
Food Science	TR	20,000	465
		25,000	484
		30,000	500
		35,000	516
		40,000	527
	EN	5,000	426
		10,000	510
		15,000	572
		20,000	608
Computer Science	TR	20,000	588
		25,000	624
		30,000	650
		35,000	674
		40,000	689
	EN	5,000	690
		10,000	903
		15,000	1038
		20,000	1133
Electronic Engineering	TR	20,000	678
		25,000	732
		30,000	771
		35,000	807
		40,000	850
	EN	5,000	832
		10,000	1123
		15,000	1283
		20,000	1403

models are trained with both Turkish and English data. For both of them, our neural network model consists of four layers:

- Embedding Layer
- Word-level Bidirectional LSTM
- Time Distributed Dense Layer
- CRF Layer

The structure of the model is given in Figure 6.2.

The training data consists of samples such that each sample consists of a padded sentence and target labels. These sentences are fed into the network and target labels are used to compare to the prediction and calculate an error. Since the network is a recurrent network, the input is one word at a time and the output is one label at a time. First, we need to vectorize words to feed into the network. So, we start with a one-hot representation for each word. We convert all words to one-hot vector format such that the dimension is the number of unique words, that is, vocabulary size.

In neural network models, words are generally represented by using word embeddings. Word embeddings are vector representations of words in a vector space. In our model architecture, the first layer is the embedding layer. It accepts the one-hot representation of one word at a time and turns indexes of the word into dense vectors of fixed size 300 and then feeds it into the BiLSTM layer. Since we start with a one-hot representation for each word, we can look up the words in the embeddings space. We used FastText pre-trained word embeddings. FastText distributes pre-trained multilingual word vectors in dimension 300. They are trained on Wikipedia and Common Crawl [36]. Further, they published aligned word vectors for 44 languages based on the pre-trained vectors computed on Wikipedia [37, 38]. Since we train two kinds of models in this study: monolingual and multilingual, there is a difference in utilization of word embeddings according to the model type. Monolingual models are trained with only Turkish data by using Turkish word embeddings. Multilingual models are trained with both Turkish and English data in the same domain by using aligned word embeddings. For short, we used Turkish word embeddings for monolingual models. Besides, aligned word embeddings of English and Turkish are used together for multilingual models. Additionally, we generate random embedding vectors for unknown words which are not included in the pre-trained vectors. Also, it should be noted that the weights of the embedding layer are not updated during

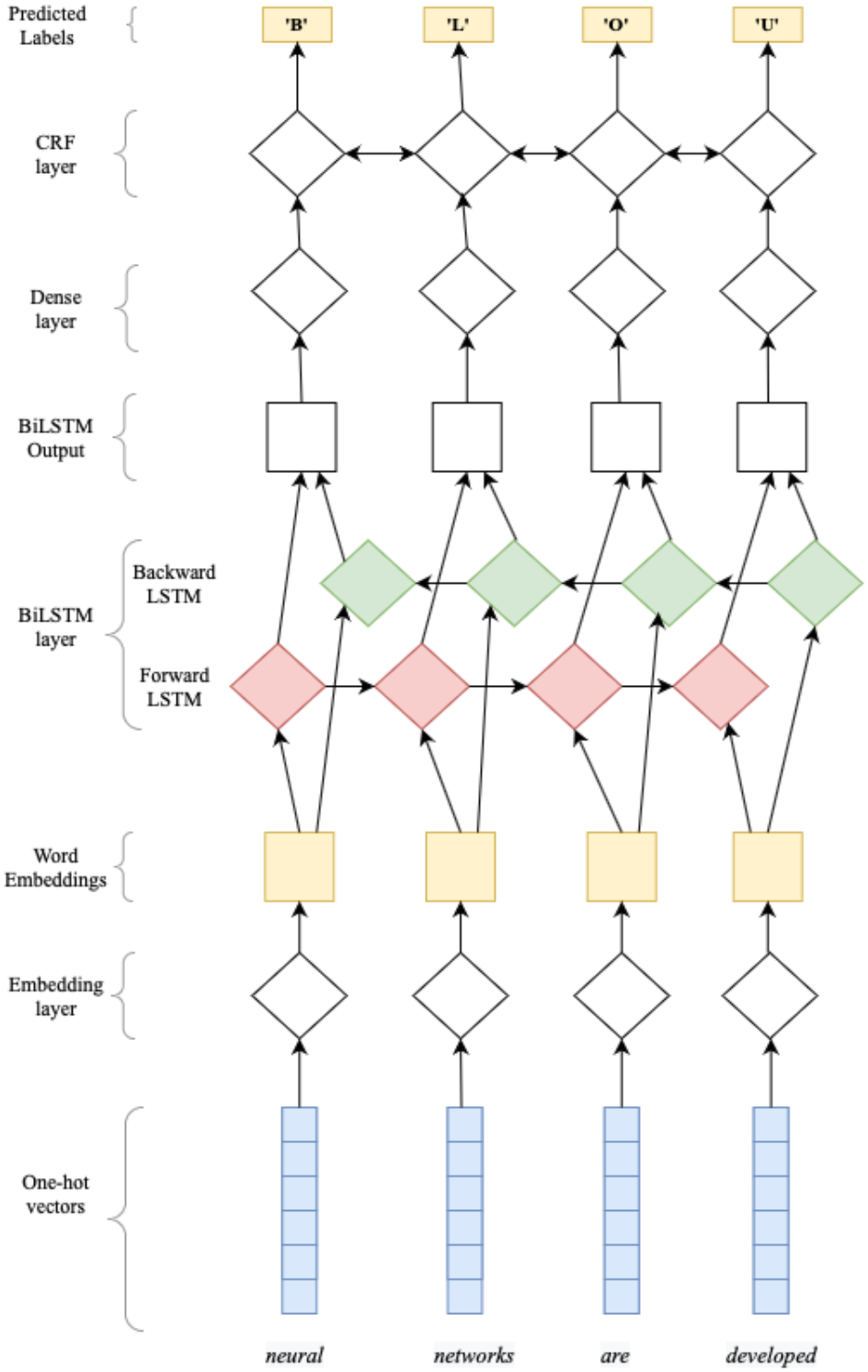

Fig. 6.2. The structure of the model

training because we set the embedding matrix to be frozen.

As an example of the inputs and outputs of the embedding layer, let there are n words in a sentence, then n-words are converted to one-hot representation to obtain sparse vectors and these vectors are the inputs. Then, the output of the embedding layer is 300-dimensional dense word vectors of n words.

The second layer of the network is word-level BiLSTM with 200 cells. This layer accepts word embeddings and generates outputs for each word in order to feed into the dense layer. BiLSTM is a special kind of RNN. Recurrent neural networks (RNN) [52] are a type of artificial neural network which are designed to recognize patterns in sequences. Long Short-Term Memory (LSTM) [40] is a special kind of RNN and they were developed to deal with RNN's the vanishing gradient problem. Bidirectional LSTM (BiLSTM) is the bidirectional version of LSTM. BiLSTM is actually two separate LSTM network layers. One of the layers, the forward LSTM layer, feeds the sentence beginning to the end. The other layer, the backward LSTM, feeds the sentence from the end to the beginning. Thus, BiLSTM processes the sentences in both the forward and the reverse. In this layer of our network, two layers of the BiLSTM generate two different context vectors for each time, and then these vectors are joined to create the final vector. Subsequently, the final vector is fed to the dense layer.

The third layer of the network is Time Distributed Dense layer. Time Distributed Dense layer is used for mapping to the five output classes of BILOU encoding. Here, time distributed means that we apply the same dense function to every time step. Since we want to generate a tag for each word in a sentence, we need to apply the same dense function for each word. This layer accepts the word-level bidirectional LSTM layer's output as input and generates the probabilities for classification using ReLU (Rectified Linear Unit) activation function [53]. Therefore, the probabilities of five output classes are generated for one word at a time.

The final layer of the model is Conditional random fields (CRF) layer. CRF is a commonly used method to take advantage of the surrounding context for sequence labeling task [27]. It is important to highlight that the relationship between labels is remarkable. Some rules can be simply derived. For example, a 'B' label can not be followed by another 'B' tag. An 'I' tag can not proceed by a 'B' tag, also an 'L' tag can not be followed by a 'I' tag.

For this reason, we add a CRF layer to the model in order to consider neighboring labels. Therefore, we take dependencies between tags into account by adding a CRF layer to our model. The input of this layer is the probabilities of five classes of BILOU encoding. The output of this layer is the recalculated probabilities, taking into account the dependencies between tags. In short, the final prediction is made for each word in this layer.

To mention hyperparameters of the network:

- We use 50 as batch size so that the weights of the model are updated after each batch is fed to the model.
- We use 50 as the number of epochs.
- We use Adam as optimizer function for training with the learning rate 0.001. Adam optimization [54] is based on stochastic gradient descent method and it is efficient in terms of computation time and memory. So, since our data is large and to reduce the training time, we prefer to use Adam optimizer.

The reason for selecting above mentioned hyperparameters is that we obtained the optimum results with these parameters and also we considered computation time and memory requirement in our experiments.

We train our network architecture with the back-propagation algorithm [55]. So, the purpose is to minimize the loss of CRF, that is, the difference between the predicted output of the network with the target output. Here, the target output is the tags we previously formed for each word in a sentence, in terminology labeling phase. On the other hand, the predicted output is the tags that the system predicts for that sentence.

We train our models with different data sizes. In that way, we measured the effects of the addition of the data both in English and Turkish. We keep constant 20,000 Turkish sentences for both monolingual and multilingual models. We gradually add 5000 Turkish sentences to the monolingual model and 5000 English sentences to the multilingual model up to 20,000 sentences.

6.5. Terminology Tag Decoding

Terminology tag decoding is the process of converting tags to terms. Our neural network model predicts tags for given sentences. The pipeline of the system is given in Figure 6.3. Since our model predicts a tag for each word in an input sentence, we need to convert predicted tags to words, in order to obtain predicted terms.

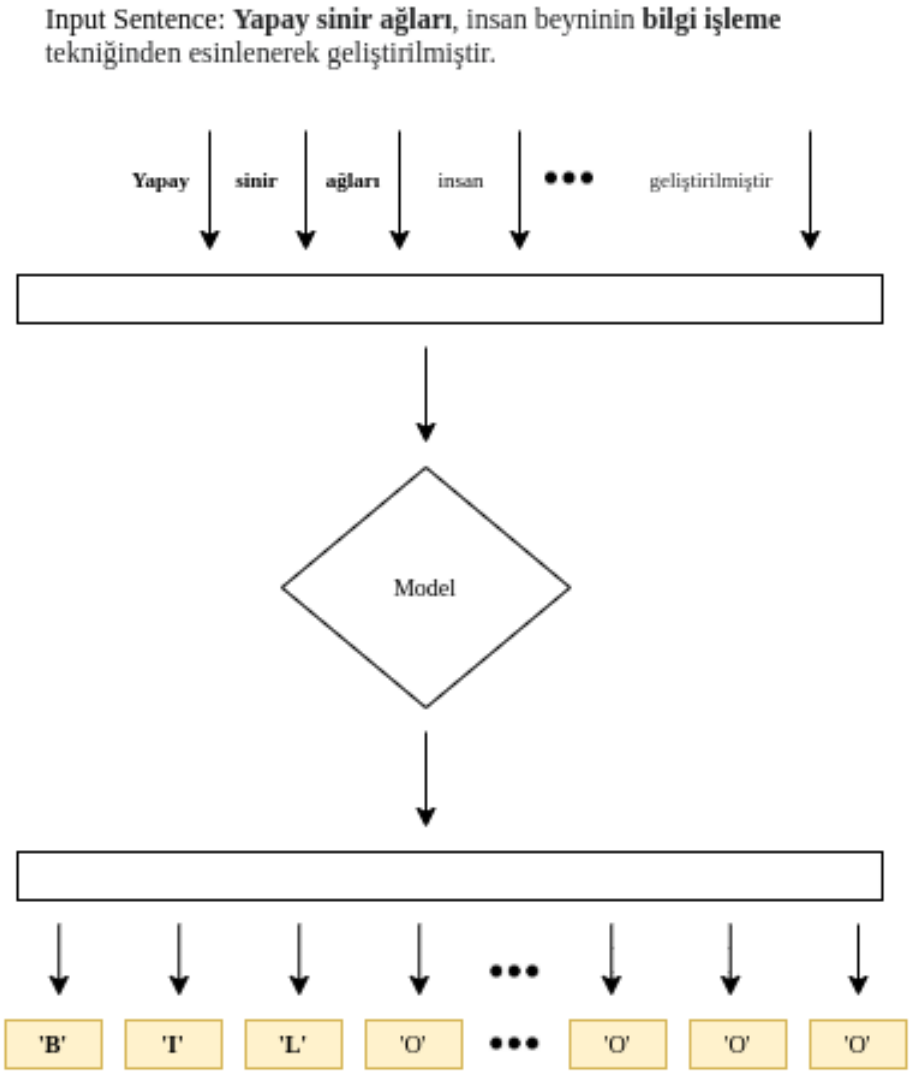

Fig. 6.3. Pipeline of the system

Basically, tag decoding operation is needed to obtain terms. We extract words and word groups by using word indices. The term decoder takes two inputs: a sentence and predicted tags for the sentence. Then, it uses word index information to extract terms. Finally, it

returns the extracted terms of the given sentence. For instance, a sentence and predicted tags for the sentence are given in Figure 6.4. The output of the terminology decoding operation is obviously a set of terms for each given sentence. Certainly, for sentences that do not contain any terms, the output of the terminology decoding operation is the empty set.

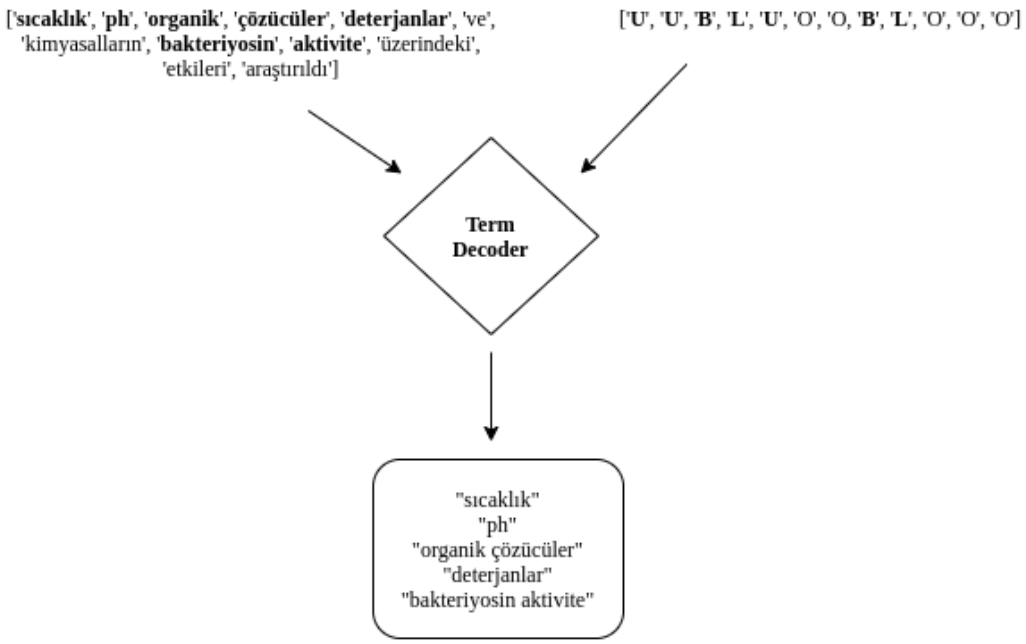

Fig. 6.4. Terminology tag decoding

6.6. Evaluation Metrics

Predicted terms is the set of extracted terms by the model on test data. The actual terms are the set of terms which are actually present on test data. We use commonly-used information retrieval metrics:

- Precision
- Recall
- F1-score

6.6.1. Precision

It attempts to answer the question that what proportion of predicted terms was actually correct.

Precision is defined as follows:

$$Precision = \frac{|\{Predicted\ Terms\} \cap \{Actual\ Terms\}|}{|\{Predicted\ Terms\}|} \tag{6.1}$$

Very low precision means that a lot of predictions are actually not terms. It implies that the prediction set has a lot of some non-term members.

6.6.2. Recall

It is not only important for a model to extract the correct terms but also how comprehensive it is. The question here is that what percentage of actual terms was extracted by the model.

Recall attempts to answer the question that what proportion of actual terms was extracted.

Recall is defined as follows:

$$Recall = \frac{|\{Predicted\ Terms\} \cap \{Actual\ Terms\}|}{|\{Actual\ Terms\}|} \tag{6.2}$$

6.6.3. F1-Score

When evaluating a model, it is beneficial to consider Precision and Recall together. F1-Score is a measure that combines precision and recall. Precision and recall are evenly weighted.

Mathematically, F1-Score is defined as follows:

$$F_1\text{-}score = 2 \cdot \frac{Precision * Recall}{Precision + Recall} \tag{6.3}$$

6.7. Results

In this subsection, we will discuss the experimental results. We tested both monolingual and multilingual models trained with different data sizes on fixed-size Turkish test data. It should be noted that the test phase of each domain is totally independent from each other. We calculate the Recall, Precision, and F1-score as we defined in the previous section. The results for each data size and each domain are given in Table 6.4.

Table 6.4. Results

Domain	Model	Training Sentences	Actual Terms	Predicted Terms	Precision	Recall	F1-score
Chemistry	Monolingual	20,000 TR	569	613	0.693	0.747	**0.719**
		25,000 TR		557	0.785	0.768	0.776
		30,000 TR		623	0.742	0.812	0.775
		35,000 TR		601	0.797	0.842	0.819
		40,000 TR		542	0.871	0.830	**0.850**
	Multilingual	20,000 TR + 5,000 EN		604	0.735	0.780	0.757
		20,000 TR + 10,000 EN		571	0.806	0.806	0.806
		20,000 TR + 15,000 EN		594	0.778	0.812	0.794
		20,000 TR + 20,000 EN		529	0.849	0.789	**0.818**
Food Science	Monolingual	20,000 TR	247	285	0.660	0.761	**0.707**
		25,000 TR		271	0.716	0.785	0.749
		30,000 TR		284	0.708	0.814	0.757
		35,000 TR		307	0.674	0.838	0.747
		40,000 TR		247	0.838	0.838	**0.838**
	Multilingual	20,000 TR + 5000 EN		267	0.738	0.798	0.767
		20,000 TR + 10,000 EN		207	0.884	0.741	**0.806**
		20,000 TR + 15,000 EN		279	0.706	0.798	0.749
		20,000 TR + 20,000 EN		251	0.765	0.777	0.771
Computer Science	Monolingual	20,000 TR	308	334	0.659	0.714	**0.685**
		25,000 TR		324	0.701	0.737	0.718
		30,000 TR		312	0.731	0.740	0.735
		35,000 TR		313	0.760	0.773	**0.767**
		40,000 TR		352	0.710	0.812	0.758
	Multilingual	20,000 TR + 5000 EN		325	0.708	0.747	0.727
		20,000 TR + 10,000 EN		262	0.817	0.695	0.751
		20,000 TR + 15,000 EN		277	0.830	0.747	**0.786**
		20,000 TR + 20,000 EN		334	0.671	0.727	0.698
Electronic Engineering	Monolingual	20,000 TR	352	340	0.594	0.574	**0.584**
		25,000 TR		276	0.725	0.568	0.637
		30,000 TR		285	0.73	0.594	0.656
		35,000 TR		369	0.591	0.619	0.605
		40,000 TR		314	0.745	0.665	**0.703**
	Multilingual	20,000 TR + 5000 EN		283	0.703	0.565	0.627
		20,000 TR + 10,000 EN		297	0.677	0.571	0.619
		20,000 TR + 15,000 EN		290	0.731	0.602	**0.660**
		20,000 TR + 20,000 EN		289	0.727	0.597	0.655

From these results, it is clear that if we have only 20,000 Turkish sentences, then using multilingual models with additional English data improves the term extraction performance. The improvement is achieved in both Precision and Recall. If we examine

the results, for Chemistry, the monolingual model trained with 20,000 Turkish sentences achieves 0.719 in F1-score. On the other hand, the multilingual model trained with 20,000 Turkish and 20,000 English sentences achieves 0.818 in F1-score. Although the results show that adding English data and using a multilingual model provide an improvement, the improvement rate varies according to domain and the size of the additional data. For example, the highest improvement in F1-score for Computer Science is 10.1 % by the multilingual model trained with additional 15,000 English sentences. Besides, the highest improvement for Chemistry is 9.9 % by the multilingual model trained with additional 20,000 English sentences. When we compare the highest improvement rates in F1-score for each domain, the domain with the highest improvement in performance is Computer Science by 10.1 %, and the domain with the least improvement is Electronic Engineering by 7.6 %.

When we examine the extracted terms by the monolingual model and the multilingual model, we noticed that there are some terms recognized by the multilingual model and not detected by the monolingual model. For instance, the term *"deniz tuzu"* does not appear in Turkish training data in Chemistry, but the multilingual model recognizes this term on test data. Here, we noticed that *"sea salt"* appears on English training data. Another example, for the domain of Electronic Engineering, the term *"besleme gerilimi"* appears only one time in Turkish training data. Since the frequency of the term is low, the monolingual model can not detect this term, although it exists in the test data. Conversely, the multilingual model detects the term, because the term *"supply vector"* appears 7 times in English training data. From these examples, it is apparent that there is an effect such that a translation of a Turkish term appears in English training data. This affects the training of the model, hereby there is an improvement by exchanging information between the data in English and the data in Turkish. Certainly, it is achieved by using the shared parameters and the shared word vector space on the single joint multilingual model.

On the other hand, we noticed that monolingual models outperform multilingual models trained with the same data size. So, the question here is that what is the difference in terms of performance between adding Turkish data and adding English data. So, we compare the performance of monolingual and multilingual models trained with the same data size. The comparison of monolingual and multilingual models by data size is given for each domain in Figure 6.5.

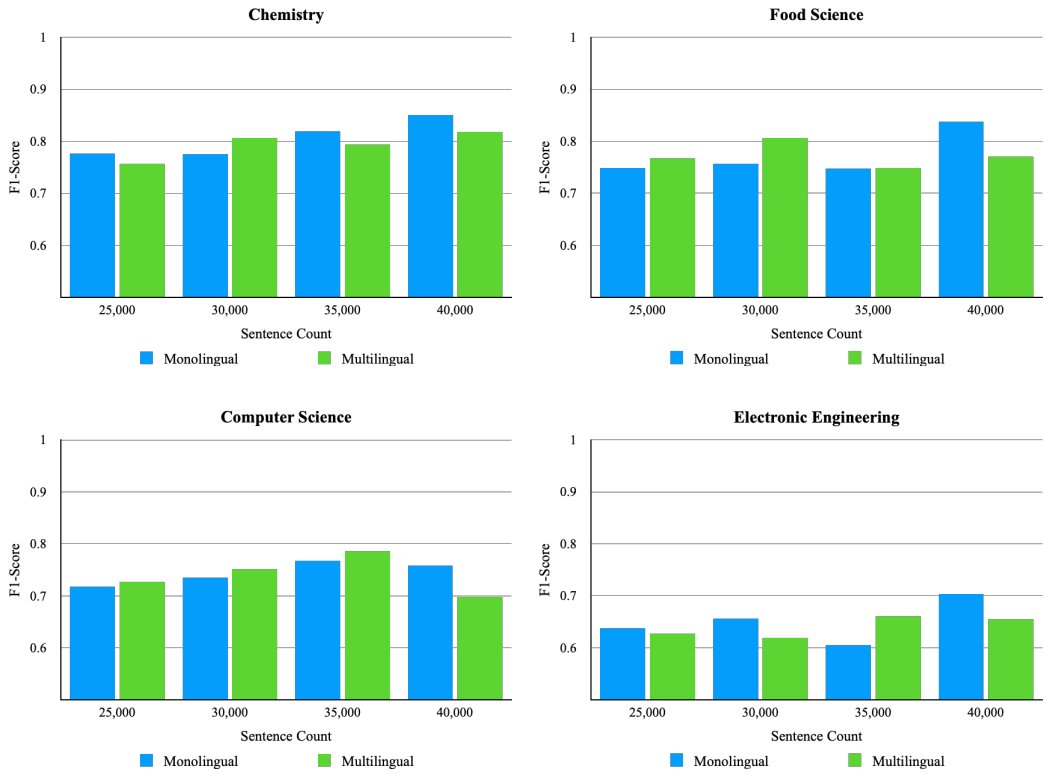

Fig. 6.5. Comparison of monolingual and multilingual models

Comparisons revealed that the monolingual model generally outperforms the multilingual model with the same data size for all domains. From here, we see that multilingual models have some shortcomings. We noticed two reasons for the gap between the monolingual and the multilingual models. The first reason is the frequency factor. Obviously, adding data in the same language and domain amplifies the frequencies of the terms. This affects the training of the models. As a result of this, there are some terms recognized by the monolingual models but not by the multilingual models on test data. For example, if we consider the two models for Chemistry: monolingual model with 40,000 Turkish sentences and the multilingual model with 20,000 Turkish and 20,000 English sentences, the term *"iridyum"* never occurs in 20,000 Turkish training sentences. However, the addition of 20,000 sentences amplifies the frequency as 2 times so that the monolingual model recognizes the term on test data whereas the multilingual model does not.

The second reason is the problem of identifying nested terms. There are incorrectly detected words that are part of a gold term. To give an example of this situation, the terms *"oturum açma"*, *"oturum anahtarı"*, and *"oturum katmanı"* exist in Computer Science

terminology dictionary. On the test data, not all these terms appear, but similar terms like *"oturum başlatma"* appear. Additionally, the word *"oturum"* appears alone on test sentences. In this case, the multilingual model detects the word *"oturum"* as a single-token term. Here, whether the subsets of multi-word terms can be considered terms is a matter of debate. There are also errors by monolingual models due to the same situation. However, we noticed that the error rates of subset term extraction for monolingual models are less than the multilingual models for some domains. The error rates of subset term extraction for both monolingual and multilingual models are given for each domain and data size in Table 6.5.

We also conduct another experiment for Computer Science to answer the question that as more English sentences are included for training, does the performance improve continuously or is there no improvement after a certain point for the multilingual model. Another question is that does it cause noise after a certain point. We keep the number of Turkish sentences constant as 20,000 and increased the number of English sentences up to 50,000 for training. The results of the experiment for each data size are given in Table 6.6.

According to results of the experiment, if we take the performance of the monolingual model trained with 20,000 Turkish sentences as a baseline (0.685), the multilingual models trained with 5,000 to 15,000 English sentences constantly improves the performance. The F1-scores for these models are 0.727, 0.751, 0.786, respectively. However, the multilingual model trained with 20,000 English sentences reduces the F1-score to 0.698. In the range of the sentences between 25,000 and 50,000, whether there is improvement and the rate of improvement varies. On the other hand, the highest F1-Score is achieved by the multilingual model trained with 45,000 English sentences. But in general, there is no big performance difference between adding 15,000 English sentences and adding 50,000 English sentences. Actually, more than 15,000 English sentences no longer provide a major performance improvement. It can even be seen that some data sizes cause performance degradation due to noise. The performance comparison multilingual models trained by different number of English sentences in the field of Computer Science is given in Fig 6.6.

Table 6.5. The error rates of subset term extraction

Domain	Data Size	Model	Error
Chemistry	25,000	Monolingual	0.142
		Multilingual	**0.250**
	30,000	Monolingual	0.112
		Multilingual	**0.171**
	35,000	Monolingual	0.050
		Multilingual	**0.182**
	40,000	Monolingual	0.143
		Multilingual	**0.200**
Food Science	25,000	Monolingual	0.091
		Multilingual	**0.157**
	30,000	Monolingual	0.108
		Multilingual	**0.291**
	35,000	Monolingual	**0.170**
		Multilingual	0.121
	40,000	Monolingual	**0.150**
		Multilingual	0.102
Computer Science	25,000	Monolingual	0.124
		Multilingual	**0.168**
	30,000	Monolingual	0.083
		Multilingual	**0.125**
	35,000	Monolingual	0.147
		Multilingual	**0.191**
	40,000	Monolingual	0.157
		Multilingual	**0.280**
Electronic Engineering	25,000	Monolingual	**0.223**
		Multilingual	0.167
	30,000	Monolingual	0.276
		Multilingual	**0.302**
	35,000	Monolingual	**0.331**
		Multilingual	0.269
	40,000	Monolingual	**0.300**
		Multilingual	0.165

Table 6.6. Results in the field of Computer Science

Domain	Model	Training Sentences	Actual Terms	Predicted Terms	Precision	Recall	F1-score
Computer Science	Monolingual	20,000 TR	308	334	0.659	0.714	0.685
		25,000 TR		324	0.701	0.737	0.718
		30,000 TR		312	0.731	0.740	0.735
		35,000 TR		313	0.760	0.773	0.767
		40,000 TR		352	0.710	0.812	0.758
	Multilingual	20,000 TR + 5000 EN		325	0.708	0.747	0.727
		20,000 TR + 10,000 EN		262	0.817	0.695	0.751
		20,000 TR + 15,000 EN		277	0.830	0.747	0.786
		20,000 TR + 20,000 EN		334	0.671	0.727	0.698
		20,000 TR + 25,000 EN		276	0.833	0.747	0.788
		20,000 TR + 30,000 EN		330	0.739	0.792	0.765
		20,000 TR + 35,000 EN		318	0.730	0.753	0.741
		20,000 TR + 40,000 EN		325	0.674	0.711	0.692
		20,000 TR + 45,000 EN		291	0.821	0.776	0.798
		20,000 TR + 50,000 EN		298	0.802	0.776	0.789

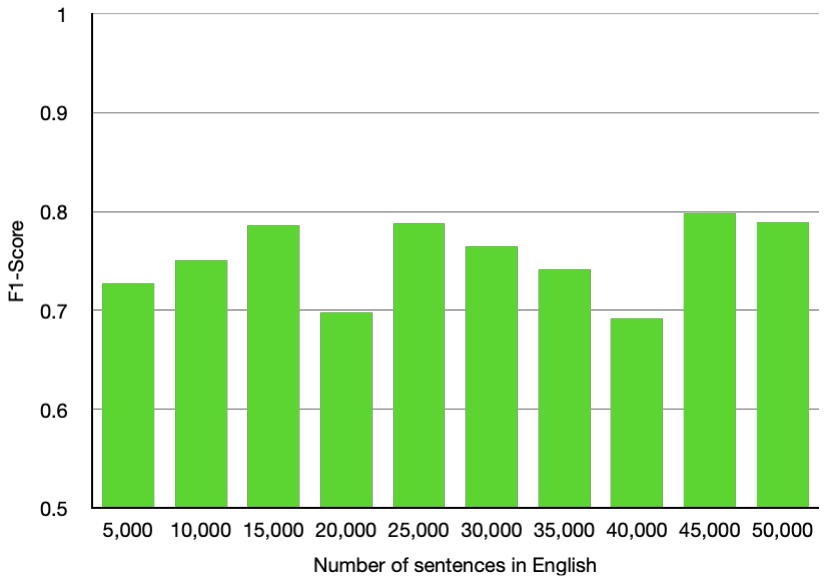

Fig. 6.6. Performance comparison of multilingual models by different number of sentences in English in the field of Computer Science

7. CONCLUSION

Automatic term extraction is a useful starting point for ontology learning, machine translation, etc. In this thesis, within the scope of automatic term extraction, we focused on two different problems. The first problem is to measure inconsistency of scientific terminology for different scientific disciplines. The second problem is to overcome the difficulties of traditional term extraction methods and leverage multilingual with joint multilingual learning based on deep learning methods with sequence labeling.

The first part of the thesis is the study of measuring terminology inconsistency. Terminology consistency is an important factor for the dissemination of scientific information among researchers. Scientific fields vary in terms of terminology consistency. Translations for scientific terms are more consistent for some of the scientific fields than others. In this thesis, we proposed a valid metric for terminology inconsistency and measured terminology inconsistency of scientific groups by our metric. Our results showed that the order of scientific groups by inconsistency in terminology is: PHY (Physical Sciences and Engineering) $>$ SOC (Social and Behavioral Sciences) $>$ LIF (Life Sciences). We also surveyed for verification of our metric and results.

The second part of the thesis is the study of automatic term extraction with joint multilingual learning. Deep learning approaches for automatic term extraction give promising results with sufficient data. It is the fact that having enough data allows the deep neural models to improve. Conversely, some domains may have limited data in some languages. At this point, we try to improve automatic term extraction by using data in another language in the same domain. In this thesis, we proposed a joint multilingual model for automatic term extraction with neural sequence labeling and deep learning. It is a single model in two languages: Turkish and English. This model enables data and word embeddings sharing. In this way, it is possible to tackle the problems of lack of data in some domains.

Our results demonstrated that adding English data and using a multilingual model provide an improvement for automatic term extraction task, considering that we only have a limited number of Turkish sentences. Our evaluation shows that the highest improvement in F1-score is 10.1 % in Computer Science, the least improvement is 7.6 % in Electronic

Engineering. Also, extensive results carried out show that the multilingual model operates at a performance close to the monolingual model trained with sufficient data. This proves that term extraction performance can be improved by using data from another language with using a joint multilingual model. In this study, we also compared the monolingual and multilingual models. We explained the reasons for the gap between the monolingual and multilingual models in terms of performance. Although the multilingual model has its shortcomings, we concluded that it can be utilized with sources in another language in case of lack of data.

REFERENCES

[1] K. Kageura, B. Umino, Methods of automatic term recognition: A review, Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication 3 (2) (**1996**) 259–289. doi:10.1075/term.3.2.03kag.

[2] H. Costa, A. Zaretskaya, G. Corpas Pastor, M. Seghiri Domínguez, Nine terminology extraction tools: Are they useful for translators? (**2016**).

[3] M. A. Saraireh, Inconsistency in technical terminology: A problem for standardization in arabic, Babel 47 (1) (**2001**) 10–21. doi:10.1075/babel.47.1.03sar.

[4] A. Schindler, Terminology in speech pathology: Old problem, new solutions, Advances in Speech Language Pathology 7 (2) (**2005**) 84–86. doi:10.1080/14417040500125327.

[5] G. Tan, G. Fang, On terminology consistency in english translations of zhuangyi texts, Journal of Contemporary Educational Research 4 (07 **2020**). doi:10.26689/jcer.v4i7.1362.

[6] A. Morshedi Tonekaboni, M. Abianeh², An investigation about the english -persian translation of terminology in biology university textbook, **2020**.

[7] F. Mu, Z. Yu, L. Wang, Y. Wang, Q. Yin, Y. Sun, L. Liu, T. Ma, J. Tang, X. Zhou, Keyphrase extraction with span-based feature representations, CoRR abs/2002.05407 (**2020**). arXiv:2002.05407.

[8] R. Alzaidy, C. Caragea, C. L. Giles, Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, **2019**, p. 2551–2557. doi:10.1145/3308558.3313642.

[9] P. Domingos, A few useful things to know about machine learning, Commun. ACM 55 (10) (**2012**) 78–87. doi:10.1145/2347736.2347755.

[10] N. Astrakhantsev, ATR4S: Toolkit with state-of-the-art automatic terms recognition methods in Scala, arXiv preprint arXiv:1611.07804 (**2016**).

[11] M. P. Marcus, B. Santorini, M. A. Marcinkiewicz, Building a large annotated corpus of English: The Penn Treebank, Computational Linguistics 19 (2) (**1993**) 313–330.

[12] N. Astrakhantsev, D. Fedorenko, D. Turdakov, Methods for automatic term recognition in domain-specific text collections: A survey, Programming and Computer Software 41 (**2015**) 336–349. doi:10.1134/S036176881506002X.

[13] K. Frantzi, S. Ananiadou, H. Mima, Automatic recognition of multi-word terms: The c-value/ nc-value method, Int. J. on Digital Libraries 3 (**2000**) 115–130.

[14] K. Ahmad, L. Gillam, L. Tostevin, University of surrey participation in TREC8: Weirdness Indexing for Logical Document Extrapolation and Retrieval (WILDER), in: Proceedings of The Eighth Text REtrieval Conference, **1999**.

[15] D. A. Evans, R. G. Lefferts, Clarit-trec experiments, Information Processsing and Management 31 (3) (**1995**) 385–395.

[16] L. L. Earl, Experiments in automatic extracting and indexing, Information Storage and Retrieval 6 (4) (**1970**) 313–330. doi:10.1016/0020-0271(70)90025-2.

[17] Z. Zhang, J. Gao, F. Ciravegna, JATE 2.0: Java automatic term extraction with Apache Solr, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), European Language Resources Association (ELRA), Portorož, Slovenia, **2016**, pp. 2262–2269.

[18] G. Bordea, P. Buitelaara, T. Polajnara, Domain-independent term extraction through domain modelling, in: the 10th International Conference on Terminology and Artificial Intelligence (TIA 2013), **2013**.

[19] N. Astrakhantsev, Methods and software for terminology extraction from domain-specific text collection, Ph. D. thesis, Institute for System Programming of Russian Academy of Sciences (2015).

[20] K. Meijer, F. Frasincar, F. Hogenboom, A semantic approach for extracting domain taxonomies from text, Decision Support Systems 62 (2014) 78–93.

[21] A. Rigouts Terryn, V. Hoste, P. Drouin, E. Lefever, TermEval 2020: Shared task on

automatic term extraction using the annotated corpora for term extraction research (ACTER) dataset, in: Proceedings of the 6th International Workshop on Computational Terminology, European Language Resources Association, Marseille, France, **2020**, pp. 85–94.

[22] M. Kucza, J. Niehues, T. Zenkel, A. Waibel, S. Stüker, Term extraction via neural sequence labeling a comparative evaluation of strategies using recurrent neural networks, **2018**, pp. 2072–2076. doi:10.21437/Interspeech.2018-2017.

[23] X. Han, L. Xu, F. Qiao, Cnn-bilstm-crf model for term extraction in chinese corpus, in: X. Meng, R. Li, K. Wang, B. Niu, X. Wang, G. Zhao (Eds.), Web Information Systems and Applications, Springer International Publishing, Cham, **2018**, pp. 267–274.

[24] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (**2018**). arXiv:1810.04805.

[25] R. Wang, W. Liu, C. McDonald, Featureless domain-specific term extraction with minimal labelled data, in: Proceedings of the Australasian Language Technology Association Workshop 2016, Melbourne, Australia, **2016**, pp. 103–112.

[26] L. Rabiner, B. Juang, An introduction to hidden markov models, IEEE ASSP Magazine 3 (1) (**1986**) 4–16. doi:10.1109/MASSP.1986.1165342.

[27] J. D. Lafferty, A. McCallum, F. C. N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **2001**, p. 282–289.

[28] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, **2014**, pp. 1532–1543. doi:10.3115/v1/D14-1162.

[29] J.-D. Kim, T. Ohta, Y. Tateisi, J. Tsujii, GENIA corpus—a semantically annotated

corpus for bio-textmining, Bioinformatics 19 (**2003**) i180–i182. doi:10.1093/bioinformatics/btg1023.

[30] B. QasemiZadeh, A.-K. Schumann, The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), European Language Resources Association (ELRA), Portorož, Slovenia, **2016**, pp. 1862–1868.

[31] W. A. Gale, K. W. Church, A program for aligning sentences in bilingual corpora, Computational Linguistics 19 (1) (**1993**) 75–102.

[32] K. Knight, A statistical mt tutorial workbook, **2003**.

[33] F. J. Och, H. Ney, A systematic comparison of various statistical alignment models, Computational Linguistics 29 (1) (**2003**) 19–51.

[34] C. Dyer, V. Chahuneau, N. A. Smith, A simple, fast, and effective reparameterization of ibm model 2, in: Proceedings of NAACL, **2013**.

[35] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, R. L. Mercer, The mathematics of statistical machine translation: Parameter estimation, Computational Linguistics 19 (2) (**1993**) 263–311.

[36] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, T. Mikolov, Learning word vectors for 157 languages, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), **2018**.

[37] A. Joulin, P. Bojanowski, T. Mikolov, H. Jégou, E. Grave, Loss in translation: Learning bilingual word mapping with a retrieval criterion, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, **2018**.

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

[39] A. Joulin, P. Bojanowski, T. Mikolov, E. Grave, Improving supervised bilingual

mapping of word embeddings, CoRR abs/1804.07745 (**2018**). arXiv:1804.07745.

[40] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (**1997**) 1735–80. doi:10.1162/neco.1997.9.8.1735.

[41] T. Mikolov, Q. V. Le, I. Sutskever, Exploiting similarities among languages for machine translation (**2013**). arXiv:1309.4168.

[42] J. Alaux, E. Grave, M. Cuturi, A. Joulin, Unsupervised hyperalignment for multilingual word embeddings, arXiv preprint arXiv:1811.01124 (**2018**).

[43] E. Grave, A. Joulin, Q. Berthet, Unsupervised alignment of embeddings with wasserstein procrustes, arXiv preprint arXiv:1805.11222 (**2018**).

[44] P. Jawanpuria, A. Balgovind, A. Kunchukuttan, B. Mishra, Learning multilingual word embeddings in latent metric space: A geometric approach, Transactions of the Association for Computational Linguistics 7 (**2019**) 107–120. doi:10.1162/tacl_a_00257.

[45] M. Artetxe, G. Labaka, E. Agirre, Learning bilingual word embeddings with (almost) no bilingual data, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, **2017**, pp. 451–462. doi:10.18653/v1/P17-1042.

[46] N. R. Council, Assessing Research-Doctorate Programs: A Methodology Study, The National Academies Press, Washington, DC, **2003**.

[47] D. Varga, L. Németh, P. Halácsy, A. Kornai, V. Trón, V. Nagy, Parallel corpora for medium density languages, In Proceedings of the RANLP 2005 (**2005**) 590–596.

[48] A. A. Akın, M. D. Akın, Zemberek, an open source nlp framework for turkic languages, Structure 10 (**2007**) 1–5.

[49] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal (**1948**).

[50] J. R. Landis, G. G. Koch, The measurement of observer agreement for categorical data, Biometrics 33 (1) (**1977**) 159–174.

[51] L. Ratinov, D. Roth, Design challenges and misconceptions in named entity recognition, in: Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), Association for Computational Linguistics, Boulder, Colorado, **2009**, pp. 147–155.

[52] J. L. Elman, Finding structure in time, Cognitive Science 14 (2) (**1990**) 179–211. doi:10.1016/0364-0213(90)90002-E.

[53] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, H. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, Nature 405 (**2000**) 947–51. doi:10.1038/35016072.

[54] D. Kingma, J. Ba, Adam: A method for stochastic optimization, International Conference on Learning Representations (12 **2014**).

[55] Hecht-Nielsen, Theory of the backpropagation neural network, in: International 1989 Joint Conference on Neural Networks, **1989**, pp. 593–605 vol.1. doi:10.1109/IJCNN.1989.118638.