MULTILINGUAL IDENTIFICATION OF VERBAL MULTIWORD EXPRESSIONS USING BIDIRECTIONAL LONG SHORT-TERM MEMORY BASED ARCHITECTURES

by

Gözde Berk

B.S., Computer Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2019

ACKNOWLEDGEMENTS

First of all, I would like to express my very great appreciation to my advisor Prof. Tunga Güngör for his continuous support throughout my master's study. I am grateful to work with him due to his patience, motivation, enthusiasm, and mentoring during all stages of this study.

I would like to acknowledge that this research was supported by Boğaziçi University Research Fund Grant Number 14420.

I would like to offer my special thanks to my colleague Berna Erden. We have been through all the hardships, joys, and successes of this study together.

Last but not least, I would like to thank my family and friends for always being there for me, supporting and encouraging me.

ABSTRACT

MULTILINGUAL IDENTIFICATION OF VERBAL MULTIWORD EXPRESSIONS USING BIDIRECTIONAL LONG SHORT-TERM MEMORY BASED ARCHITECTURES

Verbal multiword expression (VMWE) identification is a challenging task for many natural language processing studies. In this study, sequence tagging approach accompanied with stochastic models and variants of IOB tagging scheme is used for VMWE identification. In the scope of this thesis, a VMWE annotated Turkish corpus is constructed as the first part of the PARSEME shared task 1.1 which is constructing VMWE annotated corpora in many languages. Additionally, a multilingual system called *Deep-BGT* is developed as the second part of the shared task which is developing language-independent VMWE identification systems using the corpora constructed in the first part. The Turkish corpus is one of the biggest corpora in the shared task. The training and test corpora that were published in the PARSEME shared task 1.0 are updated as the PARSEME shared task 1.1 training and development corpora according to the new guidelines. A new test corpus is constructed from scratch. Deep-BGT uses the bidirectional Long Short-Term Memory model with a Conditional Random Field layer on top (BiLSTM-CRF). To the best of our knowledge, this study is the first one that employs the BiLSTM-CRF model for VMWE identification. Deep-BGT was ranked the second in the open track in terms of the general ranking metric. Moreover, a novel tagging scheme called *bigappy-unicrossy* is introduced to rise to the challenge of overlapping VMWEs. Finally, the VMWE identification system is advanced by evaluating a subset of hyperparameters which consists of tagging scheme, number of units, number of BiLSTM layers, and classifier. A comprehensive analysis of BiLSTM based architectures for multilingual identification of VMWEs is presented accordingly.

ÖZET

ÇİFT YÖNLÜ UZUN-KISA VADELİ BELLEK TABANLI MİMARİLER KULLANILARAK ÇOK SÖZCÜKLÜ FİİL İFADELERİNİN ÇOK DİLLİ SAPTANMASI

Çok sözcüklü fiil ifadesi saptama birçok doğal dil işleme çalışmaları için zorlayıcı bir görevdir. Bu çalışmada, stokastik modeller ve IOB etiketleme şemasının varyantları eşliğinde dizi etiketleme yaklaşımı çok sözcüklü fiil ifadesi saptaması için kullanılmaktadır. Bu tez kapsamında, PARSEME ortak çalışmanın ilk bölümü olan birçok dilde çok sözcüklü fiil ifadesi etiketli derlemlerin oluşturulması dahilinde çok sözcüklü fiil ifadesi etiketli Türkçe derlem oluşturulmuştur. Ek olarak, *Deep-BGT* adında çok dilli bir sistem, PARSEME ortak çalışmanın ikinci bölümü olan dilden bağımsız çok sözcüklü fiil ifadesi saptama sistemlerinin birinci bölümde oluşturulan derlemlerin kullanılarak geliştirilmesi kapsamında geliştirilmiştir. Türkçe derlemi ortak çalışmadaki en büyük derlemlerden biridir. PARSEME ortak çalışma 1.0'da yayınlanmış eğitim ve test derlemleri yeni etiketleme kurallarına göre düzenlenerek PARSEME ortak çalışma 1.1 eğitim ve geliştirme derlemleri olarak güncellenmiştir. Sıfırdan yeni bir test derlemi oluşturulmuştur. Deep-BGT üstte koşullu rastgele alanlar katmanı ile birlikte çift yönlü uzun-kısa vadeli bellek (BiLSTM-CRF) modelini kullanmaktadır. Bildiğimiz kadarıyla, bu çalışma çok sözcüklü fiil ifadesi saptaması için BiLSTM-CRF modelini kullanan ilk çalışmadır. Deep-BGT genel sıralama ölçevine göre açık yarışta ikinci olmuştur. Buna ek olarak, zorlayıcı çakışan çok sözcüklü fiil ifadelerinin üstesinden gelmek için *bigappy-unicrossy* adında yeni bir etiketleme şeması tanıtılmaktadır. Son olarak, çok sözcüklü fiil ifadesi saptama sistemi, etiketleme şeması, ünite sayısı, BiLSTM katmanı sayısı ve sınıflandırıcıdan oluşan bir üst değişkenler altkümesinin değerlendirilmesiyle geliştirilmiştir. Çok sözcüklü fiil ifadelerinin çok dilli saptanması için BiLSTM tabanlı mimarilerin kapsamlı bir analizi bu doğrultuda sunulmuştur.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b	Bias term
W	Weight matrix

LIST OF ACRONYMS/ABBREVIATIONS

AR	Arabic
BG	Bulgarian
BiLSTM	Bidirectional Long Short-Term Memory
CRF	Conditional Random Fields
DE	German
DEPREL	Dependency Relation
EL	Greek
EN	English
ES	Spanish
EU	Basque
F	F1 Score
FA	Farsi
FR	French
HE	Hebrew
HI	Hindi
HU	Hungarian
HR	Croatian
IAV	Inherently Adpositional Verbs
ID	Idiom
IReflV	Inherently Reflexive Verb
IRV	Inherently Reflexive Verb
IT	Italian
LSTM	Long Short-Term Memory
LT	Lithuanian
LVC	Light Verb Construction
MVC	Multi-Verb Construction
MWE	Multiword Expression
NER	Named Entity Recognition

NLP	Natural Language Processing
OTH	Other
P	Precision
PL	Polish
POS	Part-of-speech
PT	Brazilian Portuguese
R	Recall
RNN	Recurrent Neural Network
RO	Romanian
SL	Slovene
TR	Turkish
VID	Verbal Idiom
VMWE	Verbal Multiword Expression
VPC	Verb-Particle Construction

1. INTRODUCTION

1.1. What is Verbal Multiword Expression?

Multiword expressions (MWEs) are "a pain in the neck for natural language processing (NLP)" [1]. According to the conventional understanding, MWEs are lexical items which are formed of multiple words and their properties are unpredictable by their component words [2, 3]. For example, the meaning of *kick the bucket* is *to die* but it cannot be predicted by looking at its components. Moreover, lexical, syntactic, semantic, pragmatic and/or statistical idiomaticity occurs in MWEs [3].

According to the definition of PARSEME [4], the component words of a MWE consist of a head word and at least one other syntactically related word. If the head word of a MWE is verb, it is called verbal MWE (VMWE). Considering the *kick the bucket* example, *kick* is the head word and the remaining words are syntactically related words. Also, *kick* is verb. So, *kick the bucket* is a VMWE. According to the same definition, single-token MWEs also exist such as *snowman*.

1.2. VMWE Identification

The process of identification of MWEs is considered as the detection of the occurrences of MWEs separately in running text [3]. Identification of VMWEs is challenging for many NLP studies such as parsing and machine translation because of the nature of MWEs [5]. According to Constant *et al.* [2], the automatic annotation of MWEs can be handled by rule-based methods, classifiers, sequence tagging models, and parsing.

1.3. Approach and Contributions

There are mainly three key points focused on throughout this thesis. First of all, it is hard to carry out studies without having enough annotated corpora. The main importance is not just having a large amount of corpora. The consistency and

quality of the corpora within and between languages should be also parallel to measure the performance of the identification system properly. So, constructing an extensive multilingual corpus with VMWE annotations is necessary [5]. The very same idea is also valid for Turkish due to inadequate VMWE-annotated Turkish corpora. So, one aim of this study is to construct a Turkish VMWE Corpus.

Identification of MWEs is challenging for many NLP studies. In this study, sequence tagging approach is used for VMWE identification. In recent years, deep learning architectures have been broadly applied for a wide range of NLP tasks, especially for sequence tagging. However, they are not widespread in the area of VMWE identification. Therefore, another aim of this study is to develop a multilingual system for automatic identification of VMWEs using deep learning architectures. Different bidirectional Long Short-Term Memory (BiLSTM) based architectures are compared throughout the study. To the best of our knowledge, this is the first study that employs the bidirectional Long Short-Term Memory (BiLSTM) model with a Conditional Random Field (CRF) layer on top (BiLSTM-CRF) for VMWE identification.

Moreover, sequence tagging models are accompanied with tagging schemes. Due to the challenging nature of VMWEs, current tagging schemes are not sufficient to represent them. Therefore, a novel tagging scheme called *bigappy-unicrossy* is proposed in this study.

In the scope of this thesis, 3 papers and 1 corpus have been published as follows:

- Turkish Verbal Multiword Expressions Corpus [6]
- Annotated corpora and tools of the PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions (edition 1.1) [7]
- Deep-BGT at PARSEME Shared Task 2018: Bidirectional LSTM-CRF Model for Verbal Multiword Expression Identification [8]
- Representing Overlaps in Sequence Labeling Tasks with a Novel Tagging Scheme: bigappy-unicrossy [9]

1.4. Outline

This thesis is organized as follows:

- Chapter 2 presents related work.
- Chapter 3 explains the PARSEME shared task edition 1.1 on automatic identification of VMWEs.
- Chapter 4 gives detailed information about the construction of Turkish VMWE corpus.
- Chapter 5 describes the Deep-BGT system that participated to the PARSEME shared task edition 1.1.
- Chapter 6 proposes a novel tagging scheme called bigappy-unicrossy to represent overlaps in sequence labeling tasks and explores the effect of a tagging scheme for VMWE identification
- Chapter 7 presents a analysis of bidirectional long short-term memory (BiLSTM) based architectures for multilingual identification of VMWEs and evaluates hyperparameters for this task.
- Chapter 8 brings this thesis to a conclusion and describes the future work.

2. RELATED WORK

2.1. VMWE Annotation

There are various studies focusing on the annotation of VMWEs. Rosén *et al.* [10] present a survey that compares the existing treebanks with MWE annotations in terms of MWE categories and the annotation strategy. It is seen that there is no common way of annotating MWEs between languages and theoretical frameworks [10]. Adalı *et al.* [11] also mention the lack of standards in Turkish MWE resources and they attempt to ameliorate this situation.

The PARSEME network [12] intends to present a unified methodology for VMWE annotation [4, 5]. The PARSEME shared task 1.0 [5] aims to construct a corpus which covers 18 languages based on universal terminologies, guidelines and methodologies. In the PARSEME shared task 1.1 [4], the objective is to have more flawless and comprehensive annotation guidelines and methodologies and to extent the corpus by including more languages.

2.2. VMWE Identification

MWE identification is one of the challenging and well-known tasks of NLP [4, 5]. Constant *et al.* outline the challenges about MWE processing and the processing methods to cope with the nature of MWEs [2]. The PARSEME shared task 1.0 [5] and 1.1 [4] are interested in automatic identification of VMWEs. The desire is to have language-independent VMWE identification systems.

There are several studies that approaches the MWE identification task using different methods such as rule-based methods, classifiers, sequence tagging models, and parsing [2]. In rule based methods, rules varied from simple to advanced are applied for projection of MWE lexicons [2]. In general, classifiers are applied to word sense disambiguation [2]. A candidate is classified to distinguish a true MWE from a regular

co-occurence [2]. If MWE identification is approached as a tagging problem, stochastic models along with an IOB tagging scheme can be used [2]. Since the sequence taggers applying deep learning architectures are waiting to be discovered [2] and we treat MWE identification as MWE tagging, the sequence tagging approach accompanied with stochastic models and variants of IOB tagging scheme is used throughout the thesis and the attention is on such methods.

Ramshaw and Marcus [13] proposed the IOB tagging scheme which is the most classic tagging scheme. The IOB2 tagging scheme is the same with the IOB tagging scheme except the way of treating the single-token chunks [14]. Schneider *et al.* [15] propose different tagging schemes which are variants of the IOB tagging scheme. The proposed tagging schemes include no gaps 1-level, no gaps 2-level, gappy 1-level, and gappy 2-level. The gappy 1-level and the gappy 2-level tagging schemes [15] attempt to encode discontinuous and nested MWEs. The no gaps 2-level and the gappy 2-level tagging schemes [15] discriminate the expressions as strong and weak.

Deep neural networks are commonly employed for many NLP tasks including sequence tagging recently [8]. Graves *et al.* [16] make use of deep recurrent neural networks (RNNs), especially bidirectional Long Short-term Memory (LSTM) RNNs for phoneme recognition. The end-to-end training methods such as Connectionist Temporal Classification and RNN transducer are combined with deep, bidirectional LSTM RNNs. In phoneme recognition, this combination together with weight noise attains state-of-the art results.

Lample *et al.* [17] use two different neural architectures and deliver the state-of-the-art results for named entity recognition (NER). The first architecture uses bidirectional LSTMs and Conditional Random Fields (CRF). The second one applies a transition-based approach. Hand-crafted features and domain-specific knowledge is required for the state-of-the-art NER systems that work on small and supervised training corpora. In this study, character based and unsupervised word representations learned from the supervised and unannotated corpora are used rather than any language-specific knowledge or resources.

Legrand and Collobert [18] applies a neural network model to learn phrase representations for phrase tagging, especially MWE tagging. In phrase prediction problems, the widespread method is to use special tagging schemes. This study proposes to learn fixed-size continuous representations for arbitrarily sized chunks by word embeddings. The proposed method competes with a baseline IOBES-based system for MWE tagging.

Huang *et al.* [19] presents various architectures based on LSTM networks for sequence tagging. LSTM networks, bidirectional LSTM networks, LSTM with a Conditional Random Field (CRF) layer on top, and bidirectional LSTM with a CRF layer on top are the proposed models. The bidirectional LSTM CRF model is evaluated on POS, chunking and NER tasks. The bidirectional LSTM takes both past and future input features into consideration. The CRF layer takes the sentence level tag information into account.

In PARSEME shared task, the deep learning architectures are widespread among the participating systems. For instance, Mumpitz [20] uses a BiLSTM network. GBD-NER [21] combines a graph-based encoding layer with the BiLSTM layer. Moreover, Veyn [22] uses a RNN together with three different tagging schemes. Deep-BGT [8] which is developed as part of this thesis makes use of BiLSTM-CRF with the gappy-1 level tagging scheme.

2.2.1. The BiLSTM-CRF Model

A RNN has a memory which keeps track of history based on long distance features [19]. A RNN consists of an input layer x, hidden layer h and output layer y [19, 23]. A RNN connects the previous hidden state with the current hidden state different form the feedforward network [19]. The computation of the values in the hidden and output layers are as follows [19]:

$$h(t) = f(Ux(t) + Wh(t-1)) \tag{2.1}$$

$$y(t) = g(Vh(t)) \tag{2.2}$$

The connection weights U, W, and V are computed in training time. $f(z)$ and $g(z)$ are sigmoid and softmax activation functions.

In LSTM networks, purpose-buit memory cells shown in Figure 2.1 [19, 24] are used for the hidden layer updates to utilize long range dependencies in the data [19,25]. The implementation of the LSTM memory cell is as follows [19]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.4}$$

$$c_t = f_t c_{t-1} + i_t tanh(\sigma(W_{xc}x_t + W_{hc}h_{t-1} + b_c)) \tag{2.5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.6}$$

$$h_t = o_t tanh(c_t) \tag{2.7}$$

σ is the logistic sigmoid function. The input gate, forget gate, output gate and cell vectors are represented by i, f, o, and c respectively.

A BiLSTM network makes use of future features in addition to past features by backward and forward states apart from a LSTM network [16, 19].

Conditional Random Fields (CRF) are proposed by Lafferty *et al.* [26]. A CRF model connects both the inputs and outputs different from a LSTM and a BiLSTM networks [19, 26]. It is more advantageous than hidden Markov models and stochastic

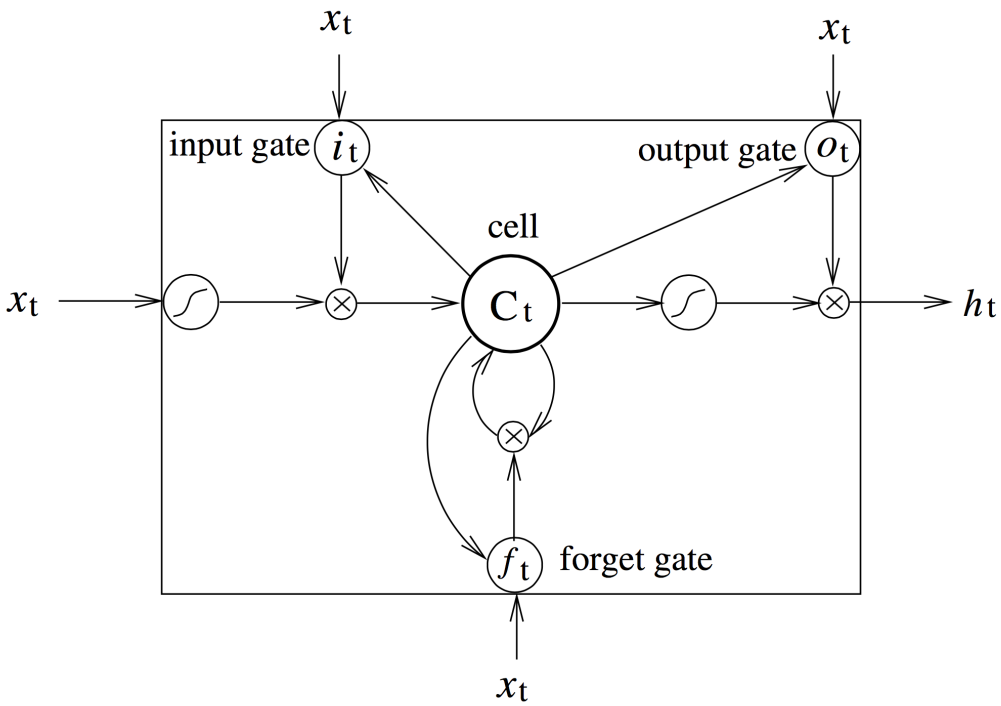

Figure 2.1. A Long Short-Term Memory cell.

grammars in respect of sequence labeling [26]. When a CRF model is combined with a BiLSTM network for sequence tagging tasks, not only past and future input features but also sentence level tag information is used [27].

3. THE PARSEME SHARED TASK

3.1. Motivation of the Shared Task

The IC1207 COST Action, PARSEME (PARSing and Multi-word Expressions) [12] is a scientific research network which focuses on multiword expressions (MWEs) and gathers experts from different disciplines and from 31 countries which have signed the Memorandum of Understanding. 30 languages and six dialects from 10 language families are covered.

PARSEME organized two shared tasks (edition 1.0 [5] and edition 1.1 [4]) on automatic identification of VMWEs in 2017 and 2018. Both shared tasks are divided into two parts. The first part is constructing VMWE annotated corpora in many languages. The second part is identification of VMWEs in running text. The aim is to compare and evaluate language-independent VMWE identification systems using a gold standard corpus annotated by the PARSEME participants in the first part.

I participated in the PARSEME shared task 1.1. In the first part of the shared task, I was part of the Turkish annotation team. Turkish annotation team consisted of three people with two annotators and one language leader. I was one of the annotators. All of our team members were native speakers of Turkish. The PARSEME shared task 1.0 includes 18 languages and Turkish is one of these languages. The PARSEME shared task 1.1 covers 20 languages including Turkish. Since Turkish is available in both of the shared tasks, the responsibility in the PARSEME shared task 1.1 is to update the Turkish training and test corpora from the previous shared task and present them as training and development corpora. Also, the other responsibility is to construct a new test corpus with at least 500 annotated VMWEs from scratch due to the fact that the PARSEME shared task 1.0 test corpus became publicly available previously and there should be a blind test corpus for the second part of the shared task.

As Turkish annotation team, we participated in the second part of the shared task with a system named *Deep-BGT*. The second part has two different tracks which are open and close. Systems using the provided corpora only submit their results to the closed track. In the case of using additional resources such as word embeddings like in our system, results should be submitted to the open track. There were 17 participating systems. The Deep-BGT system submitted results for 10 languages to the open track and ranked the second in terms of the general evaluation metric. Detailed information about both parts of the shared task will be given thereafter.

3.2. Annotation Guidelines

PARSEME released generic annotation guidelines for all languages. To make all languages consistent, all language teams should follow these guidelines. The guidelines are based on decision trees which consist of sequential tests. A unified decision tree to show which steps should be followed for VMWE annotation is shown in Figure 3.1 [4].

PARSEME divides VMWEs into 8 categories:

- Light verb constructions in which the verb is semantically totally bleached (LVC.full)
- Light verb constructions in which the verb adds a causative meaning to the noun (LVC.cause)
- Verbal idioms (VID)
- Inherently reflexive verbs (IRV)
- Fully non-compositional verb-particle constructions (VPC.full)
- Semi non-compositional verb-particle constructions (VPC.semi)
- Multi-verb constructions (MVC)
- Inherently adpositional verbs (IAV)

Light verb constructions (LVCs) and verbal idioms (VIDs) are two universal categories which means that they are valid for all languages participating in the task. Inherently reflexive verbs (IRVs), verb particle constructions (VPCs) and multi-verb

Figure 3.1. Decision tree for joint VMWE identification and classification.

constructions (MVCs) are three quasi-universal categories. So, they are only valid for some language groups or languages. Inherently adpositional verbs (IAVs) is an optional experimental category. Table 3.1 shows the number of VMWEs belonging to the categories for each language in the corpus.

Table 3.1. Number of VMWEs according to the categories.

Lang.	**VID**	**IRV**	**LVC.full**	**LVC.cause**	**VPC.full**	**VPC.semi**	**IAV**	**MVC**	**LS.ICV**
AR	1320	17	1769	0	1080	0	0	33	0
BG	1260	3223	1909	222	0	0	90	0	0
DE	1341	308	294	32	1695	153	0	0	0
EL	645	0	1622	89	38	0	0	11	0
EN	139	0	244	43	297	45	60	4	0
ES	327	714	392	81	1	0	511	713	0
EU	774	0	2866	183	0	0	0	0	0
FA	17	1	3435	0	0	0	0	0	0
FR	2165	1509	1882	97	0	0	0	24	0
HE	959	0	904	223	153	0	0	0	0
HI	61	0	641	26	0	0	0	306	0
HR	180	725	577	102	1	0	886	0	0
HU	104	0	1143	401	5156	956	0	0	0
IT	1496	1144	748	191	106	2	499	34	37
LT	308	0	479	25	0	0	0	0	0
PL	503	2279	1833	228	0	0	309	0	0
PT	1130	863	3449	94	0	0	0	0	0
RO	1611	3784	313	183	0	0	0	0	0
SL	727	1631	241	65	0	0	714	0	0
TR	3690	0	3449	0	0	0	0	2	0
Total	*18757*	*16198*	*28190*	*2285*	*8527*	*1156*	*3049*	*1127*	*37*

An LVC consists of a verb and a noun which is predicative and thereby indicates an event or a state (e.g. to *make* a *decision*). A preposition is followed by the noun (e.g. to *come into bloom*) or the noun depends on the verb (e.g. to *give* a *lecture*). If the verb is light which means that it only adds meaning expressed as morphological features, the subcategory is LVC.full (e.g. to *make* a *presentation*). If the verb is causative

which means that the subject is the cause of the state or event, the subcategory is LVC.cause (e.g. to *give* a *headache*).

A VID is formed of a head verb and one or more of the head verb's dependents (e.g. to *kick the bucket*). The type of the dependent is a characteristic feature to categorize a candidate expression as a VID in the case of only one lexicalized dependent. If the dependent is reflexive clitic or particle, the candidate may be an IRV or a VPC but not a VID. If the verb has no lexicalized dependent, the candidate may be a MVC or a VID. In the case of extended nominal phrases, the candidate may be an LVC or a VID. If the dependent belongs to any other category, the candidate is a VID. Moreover, VIDs also include sentential expressions without open slots like proverbs and conventionalized sentences (e.g. *Rome was not built in a day*). When there are multiple lexicalized dependents of the head verb, the candidate is definetely a VID but sometimes there may be also embedded VMWEs (e.g. to *let the cat out of the bag*). VIDs may also include expressions with no single clearly identifiable head verb (e.g. to *drink and drive*).

IRVs are reflexive verbs which occur with the clitic or whose non-reflexive versions have different meanings or subcategorization frames (e.g. to *find oneself* in a difficult situation).

VPCs consist of a lexicalized head verb whose dependent is particle. If the semantics of the VPC is fully non-compositional in which the meaning of the expression is totally different from the particle, the subcategory is VPC.full (e.g. to *do in*). If it is partly non-compositional in which the meaning of the expression is different from the particle such that the particle adds non-spatial semantics which is slightly predictable, the subcategory is VPC.semi (e.g. to *eat up*).

MVCs are formed by two successive adjacent verbs who have the same subject, represent the common event, and behave as a single predicate (e.g. it will *make do*).

IAVs are formed by idiomatic combinations of verbs together with an idiomatic selected preposition or postposition (e.g. to *stand for* something).

3.3. The PARSEME Corpora

The annotated corpora of the PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions [7] cover 20 languages. These languages are as follows according to their language families:

- Germanic languages: English (EN), German (DE)
- Romance languages: French (FR), Italian (IT), Romanian (RO), Spanish (ES), Brazilian Portuguese (PT)
- Balto-Slavic languages: Bulgarian (BG), Croatian (HR), Lithuanian (LT), Polish (PL), Slovene (SL)
- Other languages: Arabic (AR), Basque (EU), Farsi (FA), Greek (EL), Hebrew (HE), Hindi (HI), Hungarian (HU), Turkish (TR)

A language is unique. There may be similarities between languages which share some characteristics, families, grammar rules, and so on. However, no language is identical with another one at the end of the day. As a result, the corpus of each language will be different from each other in terms of categories of the VMWEs covered, frequency of the VMWEs, and the size of the corpus. The PARSEME 1.1 Corpus Statistics are shown in Table 3.2. The categories covered and the number of VMWEs representing the categories were given in Table 3.1. It is seen that languages differ in sizes in terms of number of sentences, number of tokens, total number of VMWEs and number of VMWEs belonging to the categories. Additionally, another reason behind the differences in the sizes of the corpora is that Arabic, Basque, Croatian, English and Hindi are presented in the PARSEME Shared Task 1.1. So, there is no available corpora from the previous shared task and all of them except Arabic has no development corpora.

Table 3.2. PARSEME 1.1 corpus statistics.

Languages	**# of Sentences**	**# of Tokens**	**# of VMWEs**
BG	21599	480413	6704
DE	8996	173293	3823
EL	8250	224762	2405
EN	7436	124203	832
ES	5515	182364	2739
EU	11158	157807	3823
FA	3617	61568	3453
FR	21067	528132	5677
HE	18700	369013	2239
HI	1684	35430	1034
HR	3837	89536	2451
HU	6159	156336	7760
IT	15728	430789	4257
LT	11104	208512	812
PL	16121	274318	5152
PT	27904	638002	5536
RO	56703	1015623	5891
SL	13511	280522	3378
TR	18612	376464	7141
Total	*280838*	*6072331*	*79326*

The provided corpora are in cupt format [4] which is an extended form of the CoNLL-U format [28]. The cupt format gathers all 10 columns of a CoNLL-U file in the same order with an additional column called PARSEME:MWE which is the eleventh column. An example of cupt format is seen in Figure 3.2. The PARSEME:MWE column is written in bold.

```
# global.columns = ID FORM LEMMA UPOS XPOS FEATS HEAD DEPREL DEPS MISC PARSEME:MWE
# source_sent_id = http://hdl.handle.net/11234/1-2515 UD_English/en-ud-train.conllu email-enronsent34_01-0024
# text = A little birdie told me that you were checking into tickets for Seoul.
1       A       a       DET     DT      Definite=Ind|PronType=Art       3       det     3:det   _       1:VID
2       little  little  ADJ     JJ      Degree=Pos      3       amod    3:amod  _       1
3       birdie  birdie  NOUN    NN      Number=Sing     4       nsubj   4:nsubj _       1
4       told    tell    VERB    VBD     Mood=Ind|Tense=Past|VerbForm=Fin        0       root    0:root  _       1
5       me      I       PRON    PRP     Case=Acc|Number=Sing|Person=1|PronType=Prs      4       iobj    4:iobj  _       *
6       that    that    SCONJ   IN      _       9       mark    9:mark  _       *
7       you     you     PRON    PRP     Case=Nom|Person=2|PronType=Prs  9       nsubj   9:nsubj _       *
8       were    be      AUX     VBD     Mood=Ind|Tense=Past|VerbForm=Fin        9       aux     9:aux   _       *
9       checking        check   VERB    VBG     Tense=Pres|VerbForm=Part        4       ccomp   4:ccomp _       2:VPC.semi
10      into    into    ADP     IN      _       11      case    11:case _       2
11      tickets ticket  NOUN    NNS     Number=Plur     9       obl     9:obl   _       *
12      for     for     ADP     IN      _       13      case    13:case _       *
13      Seoul   Seoul   PROPN   NNP     Number=Sing     11      nmod    11:nmod SpaceAfter=No   *
14      .       .       PUNCT   .       _       4       punct   4:punct _       *
```

Figure 3.2. The cupt format.

Additionally, annotated corpora of the PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions covering 19 languages except Arabic are available at [7]. Since the Arabic corpus does not have an open licence, the corpus can be obtained through LDC. Because of this requirement, Arabic was optional for the second part of the shared task but no one submitted results for Arabic. Therefore, the second part of the PARSEME shared task was evaluated based on 19 languages.

3.4. The PARSEME Evaluation Metrics

The second part of the shared task is to develop a language-independent system for VMWE identification using the provided corpora. The PARSEME shared task encourages the participants to design a multilingual system. A system's result is the automatic annotations of the blind test corpus by the system itself. This annotated corpus is compared with the gold standard. Evaluation is based on precision (P), recall (R), and F1 score (F). The evaluation metrics can be examined in two categories as general metrics and metrics dedicated to specialized phenomena.

General metrics include VMWE-based and token-based results. The results are also provided for each language and for all participating languages by taking macro-average of the scores for each language. Macro-averages are calculated by arithmetically averaging F1 scores obtained for each of the participating 19 languages. If a system does not submit results for a language, the F1 score is accepted as 0. If there is no VMWE belonging to the specific phenomena in a language, this language is not included in the evaluation.

Metrics dedicated to specialized phenomena are continuity, length, novelty and variability. These metrics are correlated with the challenges of VMWEs which will be discussed later. Continuous and discontinous VMWEs are evaluated separately in the continuity metric. VMWEs are divided into single-token and multi-token VMWEs and evaluated accordingly in the length metric. The novelty metric evaluates based on seen and unseen VWWEs. If a VMWE is annotated at least one in the training corpus, it is accepted as seen. Otherwise, it is accepted as unseen. The last metric is variability. The variants of a VMWE are formed by differentiating the strings between the first and last lexicalized components of the VMWE. The results based on these special metrics are provided only in the macro-averages.

4. CONSTRUCTION OF TURKISH VMWE CORPUS

4.1. Annotation Procedure

The PARSEME shared task 1.1 is the second edition of the shared task. At the end of the PARSEME shared task 1.0, a corpus was released. This corpus was annotated according to the previous annotation guidelines provided by PARSEME [5]. Since Turkish was one of the languages in the previous corpus, there are three tasks regarding the second edition of the shared task. The first task is to update the PARSEME shared task 1.0 training corpus according to the new annotation guidelines and release it as the PARSEME shared task 1.1 training corpus. The second task is to update the PARSEME shared task 1.0 test corpus according to the new annotation guidelines and release it as the PARSEME shared task 1.1 development corpus. The third task is to construct a new test corpus with 500 annotated VMWEs from scratch because the previous test corpus became publicly available beforehand.

Constructing PARSEME 1.1 test corpus and editing PARSEME 1.0 training and development corpora took four months. PARSEME annotation guidelines are used for annotation [4]. Additionally, some other specifications for Turkish are used.

There are some changes in the decision process and the scope of VMWEs regarding the new annotation guidelines. Also, categories were changed as shown in Figure 4.1. In the PARSEME shared task 1.0, there were five groups of VMWEs which are light verb constructions (LVCs), idioms (IDs), inherently reflexive verbs (IReflVs), verb-particle constructions (VPCs), and other (OTH). Turkish had only three groups of VMWES which are ID, LVC, and OTH. In the PARSEME shared task 1.1, some groups changed their scope and their annotation procedure. Then, they changed their names accordingly. Some new groups were added. Also, a group was removed. In the end, there are five groups of VMWEs which are light verb constructions (LVCs) with two subcategories (LVC.full and LVC.cause), verbal idioms (VIDs), inherently reflexive verbs (IRVs), verb-particle constructions (VPCs) with two subcategories (VPC.full

and VPC.semi), and multi-verb constructions (MVCs). Turkish has only three groups of VMWES which are VID, LVC.full, and MVC.

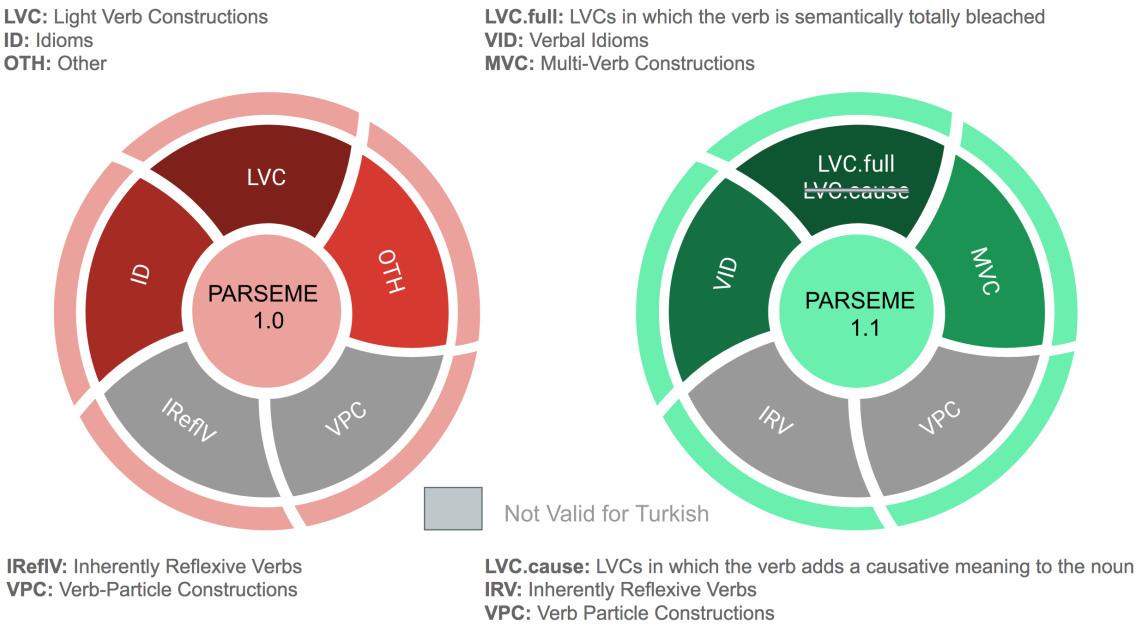

Figure 4.1. Differences of categories between the shared tasks.

The differences between the two annotation guidelines regarding Turkish can be listed as follows:

- The category of ID is updated as VID.
- The category of LVC is divided into two subcategories. There is no VMWE categorized as LVC.cause in Turkish. So, LVC is updated as LVC.full.
- A new category called MVC is added. MVCs are rarely seen in Turkish.
- The category of OTH is removed. So, the VMWEs belonging to this group may be a VID or an LVC.full or a MVC according to the new guidelines. There is also a possibility such that a VMWE belonging to this group is actually not a VMWE in terms of the new guidelines.

As mentioned above, Turkish has three types of VMWEs: LVC.full, VID and MVC. Each category has its own decision tree which includes tests specific to the

category. The sequence of categorical tests to apply is decided according to the decision tree for joint VMWE identification and classification which was shown in Figure 3.1 [4]. This generic tree directs the annotator to the categorical decision trees. In the end, the candidate expression is annotated as VMWE with a category or it is found to be a non-VMWE.

Figure 4.2 [4] shows the decision tree of VID. This decision tree consists of five tests. If one of the tests is passed, the candidate is annotated as VID. To understand these tests, we can give an example and examine the tests one by one. For example, let us consider the sentence *Karakolda ifade verdim* which means *I gave statement at the police station.* Here, the candidate expression is *ifade vermek* which means *to give statement.* The first test checks whether there is a cranberry word in the candidate or not. In our example, *ifade* and *vermek* are both meaningful words. So, there is no cranberry word and the candidate failed in the first test. The second test checks whether there is a meaning shift in the case of a regular replacement of a component or not. We can replace *ifade* which means *statement* with its synonym *açıklama* which means *explanation.* After this replacement, it is seen that there is an unexpected meaning shift. Therefore, *ifade vermek* passes the second test and is annotated as VID.

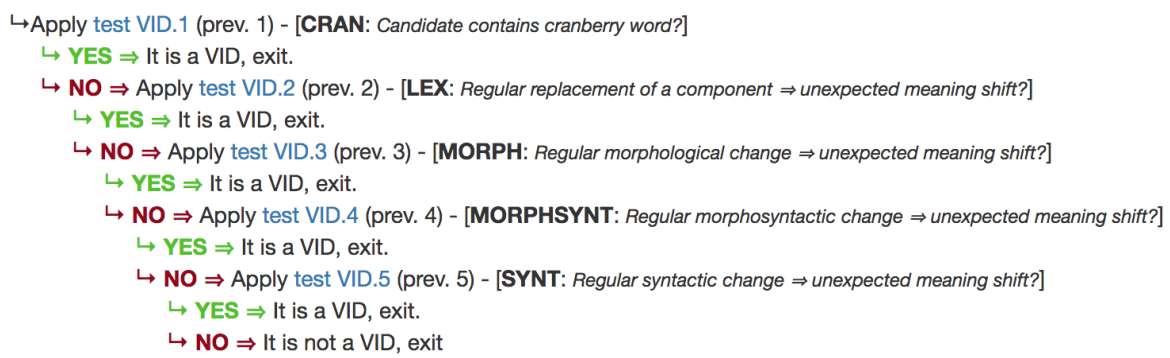

Figure 4.2. Decision tree for categorization of VID.

To understand the remaining tests, we can continue with the same example. Test three checks whether there is a meaning shift in the case of a morphological change or not. We can replace *ifade* with its plural form *ifadeler.* After this replacement,

it is seen that there is an unexpected meaning shift. Therefore, *ifade vermek* passes the third test. Test four checks whether there is a meaning shift in the case of a morphosyntactic change or not. We can replace *ifade* with its possesive form *ifadeni*. After this replacement, it is seen that there is an unexpected meaning shift. Therefore, *ifade vermek* passes the fourth test. The fifth test checks whether there is a meaning shift in the case of a syntactic change or not. We can take a look at the sentence *Karakolda verdiğim ifade tekrar kontrol edildi* which means *The statement I gave at the police station was checked again*. There is no unexpected meaning shift here. Therefore, *ifade vermek* fails the final test.

The decision tree of LVC is shown in Figure 4.3 [4]. If all of the tests are passed, the candidate is annotated as LVC. Let us follow the example *Ali ders çalışmaya karar verdi* which means *Ali decided to study* to understand the tests. The candidate expression in the example is *karar vermek* which is *to decide*. Test zero checks if the noun is abstract. The noun is *karar* in the candidate and it is abstract. So, this test is passed. Test one checks if the noun is predicative. The noun has at least one semantic argument which is *karar veren*. Therefore, this test is also passed. The second test requires the subject of the verb be a semantic argument of the noun. *Ali* is the subject of the verb and also the semantic argument of the noun. Thereby, test two is passed.

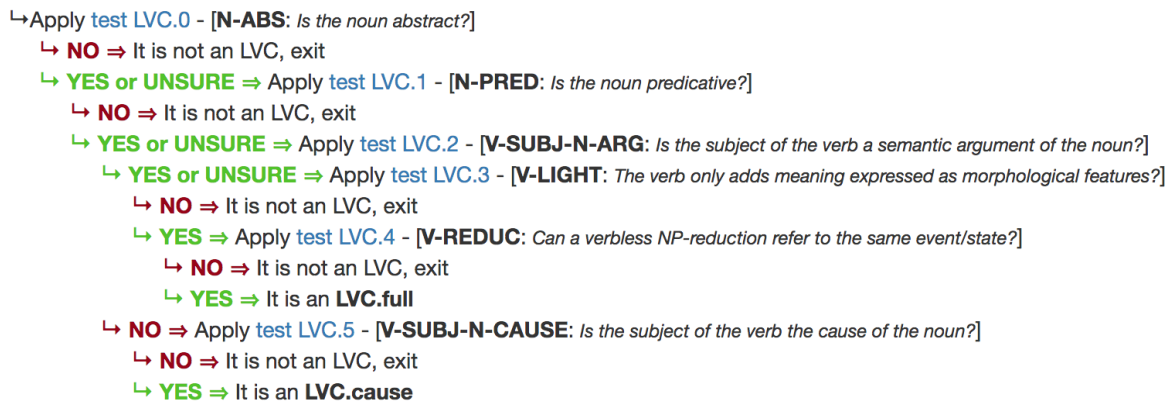

Figure 4.3. Decision tree for categorization of LVC.

Test three checks whether the verb only adds meaning expressed as morphological features or not. The verb *vermek* is semantically light. It only gives the meaning of

performing the activity of the noun. So, the third test is passed. The fourth test expects that the noun phrase formed by verb reduction refers to the same event or state. So, *Ali'nin kararı* which is *Ali's decision* refers to the same event of deciding. Hence, this test is passed too and the candidate expression is annotated as LVC.full. There is no example of LVC.cause in Turkish. Consequently, the final test is unnecessary for Turkish.

There is no multilingual decision tree for MVCs but there are some common properties regarding them. There are three tests for Turkish. We can examine the sentence *İki seçenek arasında gidip geliyorum* which means *I go between two options*. The candidate expression is *gidip gelmek* which means *to go between*. Firstly, a MVC is formed by a sequence of two adjacent verbs and both verbs should have the same subject. In the example, *gitmek* which is *to go* and *gelmek* which is *to come* have the same subject. Secondly, the verbs should depict closely connected actions and they may be seen as part of the same event. The verbs *gitmek* and *gelmek* are part of the same event. Finally, the verbs should function together as a single predicate. As we can see, the predicate of the sentence is *gidip gelmek*. As a conclusion, *gidip gelmek* is annotated as MVC.

The previous Turkish corpus consists of 2911 ID, 2624 LVC, and 634 OTH. Due to the changes in the categories, the VMWEs annotated as LVC and ID were updated as LVC.full and VID. Also, the previous training and test corpora are reviewed and some minor mistakes especially considering nested VMWEs were corrected. Since there is no more a category of OTH, the VMWEs annotated as OTH were examined.

While annotating Turkish corpus in edition 1.0, the study about categorization of Turkish MWEs was taken into consideration [11]. According to this study, Turkish VMWEs are divided into three categories which are verbal compounds, light verb constructions and idiomatic expressions. The VMWEs annotated as OTH includes the verbs *vermek*, *almak* etc., which match verbal compounds in the study. Using the new guidelines and the specifications for Turkish which will be explained in detail afterwards, the VMWEs categorized as OTH were re-annotated.

After the arrangements, the PARSEME shared task 1.0 training and test corpora became the PARSEME shared task 1.1 training and development corpora. The last task was to construct a new test corpus. The test corpus was created by gathering newspaper articles on politics, world, life, art and columns. Milliyet was selected as the newspaper source because it was used in the previous corpus. The articles were tokenized and parsed sentence by sentence and transformed into ConLL-U format using ITU NLP Pipeline [29]. Some paragraphs including tokens such as political party abbreviations were removed beforehand. The annotation is handled separately by the annotators using the guidelines. Then conflicts were resolved.

4.2. Annotation Specifications for Turkish

In Turkish, there are only three types of VMWEs which are LVC.full, VID and MVC. LVCs and VIDs are more widespread in Turkish. However, MVCs are limited. So, the main focus is on LVCs and VIDs. The annotation guidelines provided by PARSEME are used while annotating the corpus. In addition to the guidelines, some strategies are developed for Turkish and some inferences are obtained. These are explained in detail as follows.

In Turkish, there are six light verbs which are *olmak (to be)*, *etmek (to do)*, *yapmak (to make)*, *kılmak (to render)*, *eylemek (to make)* and *buyurmak (to order)* [11]. These light verbs do not only form LVC. They may form other types of VMWEs too. On the other hand, the verbs forming LVCs are not restricted with just light verbs. Also, some supportive verbs may be used. For example, *karar vermek* is an LVC and the verb *vermek* is not a light verb.

To categorize a candidate expression as LVC, all tests in the decision tree should be passed as shown in Figure 4.3. However, the annotation of LVCs is mainly based on the step of categorizing the verb as semantically light or not. If the verb is semantically light, there is a big potential of being LVC for the candidate expression. Otherwise, there is a little chance of being LVC. In general, the candidate expressions passing this test are LVCs.

Some nouns have tendency to behave like they have transformed into a verb when the noun is combined with a specific verb. It is referred as performing the activity of the noun. For example, the noun *talimat* has the meaning of *instruction*. Also, it has tendency to give the meaning of *talimat vermek* which means *to instruct*. Therefore, the noun *talimat* is connected with the verb *vermek* which becomes semantically light accordingly.

While annotating a VMWE consisting of a noun *(n)* and the verb *almak*, it should be checked whether there is a LVC in the form of *n+vermek* or not. If there is such an LVC, *n+almak* cannot be an LVC but it may be a VID because *vermek* may give the meaning of performing an activity of *n*. In that case, *n+almak* cannot give the meaning of performing an activity of *n* which results in the failure from the semantically light test. For example, *oy vermek* which is *to vote* semantically is an LVC and it passes the semantically light test since *vermek* adds no meaning to the noun *oy* which means *vote* besides that of performing an activity. However, *oy almak* is a VID because *almak* which means *to receive* gives meaning to *oy* and the total meaning becomes *to receive vote*.

To decide light/void semantics of a verb, the Turkish dictionary provided by Türk Dil Kurumu (TDK) [30] is made use of. For example, *söz vermek* means *to promise* and *söz* means *promise*. *Söz* is also *vaat* which means *vow* and *söz vermek* means *vadetmek* which is *to vow* in the dictionary. Therefore, *vermek* adds no meaning to *söz*. It only adds the meaning of performing the activity of *promise*.

According to the inferences obtained from the annotation process, we observed that the expressions formed with the verbs *sağlamak*, *duymak*, *ulaşmak*, *karşılamak*, *kullanmak*, *uygulamak* are hard to annotate by applying the tests. However, it is seen that they are not VMVEs in most of the cases in terms of annotators' observations.

4.3. Final Corpus

To sum up, after modifications on the PARSEME 1.0 Turkish corpus, PARSEME 1.0 training set is updated as PARSEME 1.1 training set, PARSEME 1.0 test set is updated as PARSEME 1.1 development set. Additionally, a new test set is constructed and presented as PARSEME 1.1 test set. The results are shown in Table 4.1.

Table 4.1. Turkish corpus.

	LVC	**VID**	**OTH**	**MVC**	***Sum***
PARSEME 1.0 Training Corpus	2624	2911	634	0	*6169*
PARSEME 1.1 Training Corpus	2950	3169	0	1	*6120*
PARSEME 1.0 Test Corpus	199	249	53	0	*501*
PARSEME 1.1 Development Corpus	227	273	0	0	*500*
PARSEME 1.1 Test Corpus	273	235	0	1	*509*

The PARSEME 1.1 test set consists of 14388 tokens and 577 sentences. The overall PARSEME 1.1 Turkish corpus includes 376464 tokens and 18612 sentences. Table 4.2 shows the statistics of the PARSEME 1.1 Turkish corpus. The PARSEME 1.1 corpora containing 19 languages including the final version of PARSEME 1.1 Turkish corpus are available at [7].

Table 4.2. PARSEME 1.1 Turkish corpus statistics.

	# of sentences	**# of tokens**
Training Corpus	16715	334880
Development Corpus	1320	27196
Test Corpus	577	14388
Total	*18612*	*376464*

5. THE DEEP-BGT SYSTEM

5.1. Motivation

As Turkish annotation team, we developed a language-independent system called *Deep-BGT* [8] and participated in the second part of the shared task. After some research, we found out that sequence tagging approach fits our problem well. Then, we decided to combine sequence tagging with a deep learning model. It is known that the bidirectional Long Short-Term Memory model with a Conditional Random Field layer on top (BiLSTM-CRF) is a well-known architecture for sequence tagging tasks [19]. Also, our system is the first one that employs this architecture for VMWE identification. Therefore, we agreed on BiLSTM-CRF model and developed the Deep-BGT system.

5.2. Tagging Scheme

According to [2], the MWE identification task can be addressed using sequence tagging methods with the BIO tagging scheme. However, it is not the best tagging option regarding the challenging nature of the MWEs. Therefore, Schneider *et al.* [15] proposed the gappy 1-level tagging scheme to overcome the challenge of discontinuity for MWE identification. The Deep-BGT system makes use of the gappy 1-level tagging scheme. The tagging scheme will be explained in detail in the following chapter.

5.3. BiLSTM-CRF Model

The BiLSTM-CRF model has five layers as shown in Figure 5.1. The input layer merges POS (part-of-speech) tag, DEPREL (dependency relation) tag and word embedding of each token. DEPREL tag is used as a feature to perceive sentence level dependencies. Since the provided corpora are not big enough to train word embeddings, pre-trained fastText word embeddings are used [31]. These embeddings were trained on Common Crawl and Wikipedia [31]. The vocabulary size of the embeddings is 2M words. The vector dimension of the embeddings is 300.

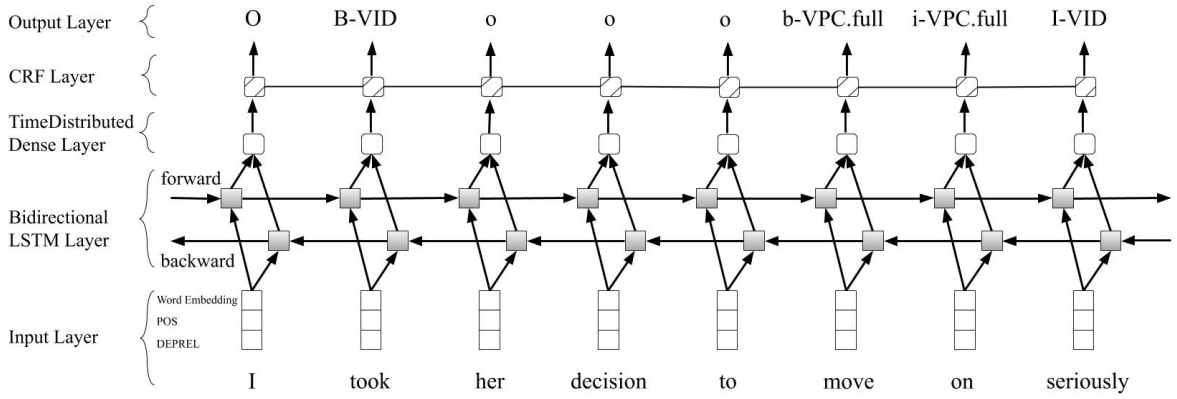

Figure 5.1. BiLSTM-CRF model.

The concatenation of the three embeddings is passed to the bidirectional LSTM layer which takes past and future features into consideration through left-to-right and right-to-left passes. The output of the BiLSTM layer is passed to the time distributed dense layer which shapes the output according to the number of tags. Then, the CRF layer comes to decode the sequence labels. Finally, the output layer produced from the CRF layer includes tags for each token.

Keras [32] with Tensorflow [33] backend is used for the implementation of the architecture. While developing the system, the time was limited because there was only one month between the release of the corpora and the submission of results. Therefore, there was no chance for fine tuning the parameters by taking all of the languages into consideration. Thereby, a study about evaluating parameters for five sequence tagging tasks was used [34]. According to the study, Nadam is found to be the best optimizer comparing with SGD, Adagrad, Adadelta, RMSProp and Adam. Nadam converges on average after nine epochs. So, Nadam is chosen as the optimization algorithm. Moreover, variational dropout is found to be better than naive dropout and no-dropout. So, a fixed dropout rate of 0.1 is chosen for the system.

Furthermore, the study claims that mini-batch sizes of 1 up to 16 are suitable for small training sets and mini-batch sizes of 8 up to 32 are suitable for large training sets. Mini-batch size of 64 does not perform well. Table 5.1 shows the chosen batch

sizes and number of epochs for each language according to the size of training corpora by taking the results of study into consideration. The optimum number of recurrent units are found to be 100 in the study. However, the study also states that the number of recurrent units slightly change the results. Therefore, the number of units is chosen as 20 to decrease the computation power. Since CRF is used as a classifer, the metric and loss function are selected as crf_viterbi_accuracy and crf_nll respectively.

Table 5.1. Model parameters.

Languages	**Batch Size**	**# of Epochs**
BG, FR, PT, RO	32	12
DE, ES, HU	16	15
IT, PL, SL	16	12

5.4. Results

The Deep-BGT system covered 10 languages in the PARSEME shared task 1.1. Five of these languages are the Romance languages, which are Spanish (ES), French (FR), Italian (IT), Brazilian Portuguese (PT), and Romanian (RO). The remaining five languages are chosen according to two criteria. The system learns better with more data because it has a deep learning architecture. So, the languages with more data are favored. Moreover, the languages with higher occurring frequency of VMWEs are considered. The sizes of the corpora are given in Table 3.2 and the frequencies are calculated from the statistics provided in the table. As a result, Bulgarian (BG), German (DE), Hungarian (HU), Polish (PL), and Slovenian (SL) are covered as the remaining five languages. Turkish (TR) is not chosen because the Deep-BGT team members and the Turkish annotation team members are the same. So, we did not want to introduce bias to system evaluation. We wanted to submit results for all 19 languages but we did not have adequate computation power to run our system on all languages in limited time.

The Deep-BGT system makes use of pre-trained word embedding additional to the provided corpora by PARSEME. Therefore, the system competes in the open track. The Deep-BGT system is ranked the second in the open track in terms of the general ranking metric.

Table 5.2 shows the cross-lingual macro average results of the Deep-BGT system over 19 languages in terms of MWE-based F-measure. Each row in the table represents a metric from the general metrics and metrics focusing on specific phenomena. The second column shows the official results on 19 languages. The Deep-BGT system submitted results for 10 languages due to time constraint and lack of computation power. As explained previously, the cross-lingual macro average results are calculated based on 19 languages and the languages with no submitted results are assumed to have 0 precision, 0 recall and 0 F1 score. So, the results regarding 19 languages conceals the real performance of the Deep-BGT system. To handle this issue, the results are recalculated by arithmetic averaging over the 10 languages covered. Consequently, the third column shows the unofficial results on 10 languages.

Table 5.2. The macro-averaged results of Deep-BGT.

Metrics	**Official Results on 19 Languages**	**Unofficial Results on 10 Languages**	**Official Best Results on 19 Languages**
General ranking	28.79	54.70	58.09
Continuous VMWEs	31.23	59.34	62.74
Discontinuous VMWEs	23.19	44.06	43.19
Multi-token VMWEs	29.24	55.56	59.93
Single-token VMWEs	25.87	43.12	28.32
Seen-in-train VMWEs	36.66	69.65	76.31
Unseen-in-train VMWEs	12.99	24.68	28.46
Variant-of-train VMWEs	29.94	56.89	63.25
Identical-to-train VMWEs	41.01	77.92	87.63

As already explained before, the shared task presents some specific metrics. The performance of the system regarding these metrics can be understood better with a

reference. The third column of Table 5.2 shows the best results obtained for each metric from the system ranked first. These results can be a reference to judge the performance of the Deep-BGT system.

The Deep-BGT system handles not only continuous VMWEs but also discontinuous ones. The system performs better in continuous VMWEs compared to discontinuous ones as expected because it is easier to identify continuous VMWEs. However, the performance of the Deep-BGT system regarding discontinuous VMWEs is slightly higher than the performance of the best system by means of the gappy 1-level tagging scheme.

According to the VMWE definition of PARSEME, single-token VMWEs also occur. The Deep-BGT system also takes them into consideration. The performance of the system in respect of multi-token VMWEs is higher than single-token ones. Actually, there is an invisible success for single-token VMWEs because the Deep-BGT system outperforms the best system.

It is known that the identification of seen-in-train VMWEs and identical-to-train VMWEs is easier than the identification of unseen-in-train and variant-of-train VMWEs in the case of deep learning architectures. The results also verifies this claim.

Table 5.3 shows the results of the Deep-BGT system in terms of languages. It presents MWE-based and Token-based precision (P), recall (R), F-measure (F1), and rankings in the open-track. Deep-BGT is the best system in Bulgarian (BG) according to both MWE-based and Token-based F-measure, and in German (DE) according to MWE-based F-measure. Discontinuity is more common in Germanic languages [2]. So, ranking the first in German in terms of MWE-based scores adds to the value of Deep-BGT. The system was ranked first in terms of the token-based F-measure in French (FR) and Polish (PL). In general, the Deep-BGT system was ranked second.

Table 5.4 and Table 5.5 show MWE-based and token-based F1 scores per VMWE category of Deep-BGT respectively. If a language does not have a VMWE belonging

Table 5.3. The language-specific results of Deep-BGT.

	MWE-based				**Token-based**			
Languages	**P**	**R**	**F1**	**Rank**	**P**	**R**	**F1**	**Rank**
BG	85.96	52.99	65.56	1	91.00	52.82	66.85	1
DE	60.94	36.35	45.53	1	77.92	37.64	50.76	3
ES	24.50	34.20	28.55	2	33.13	38.61	35.66	2
FR	57.81	49.80	53.51	2	78.88	56.45	65.80	1
HU	78.00	71.26	74.48	2	80.71	73.11	76.72	2
IT	45.52	25.60	32.77	2	70.00	27.63	39.62	2
PL	70.87	56.70	63.00	2	80.23	57.85	67.23	1
PT	72.44	46.11	56.35	2	79.40	44.83	57.30	2
RO	79.80	69.10	74.07	2	92.11	73.66	81.86	2
SL	58.90	38.40	46.49	2	72.19	40.34	51.76	2

to a category in the test corpus, the cell that combines the category and the language is marked with "-". When we compare these tables with Table 3.1 which shows the number of VMWEs per category for each language, there is a correlation between them. F1 scores in Table 5.4 and Table 5.5 are correlated with the number of VMWEs in Table 3.1. The underlying reason for the correlation is that the Deep-BGT system makes use of deep learning architectures. To conclude, the system learns better with more data and the role of the occurrence frequency of VMWEs cannot be ignored.

5.5. Extending Deep-BGT to 19 Languages

After the shared task, we wanted to test our system on all of the 19 languages using the hyperparameters in Table 5.6. Table 5.7 shows the results of the Deep-BGT extended to 19 languages in the columns labeled with *extended*. In the shared task, the evaluation is based on the annotation of the blind test. The results of the participated 10 languages are preserved. For the remaining nine languages, the evaluation procedure applied is the same. Each result is the evaluation of a single run as in the shared task.

Table 5.4. MWE-based F1 scores per VMWE category of Deep-BGT.

Lang.	**LVC.full**	**LVC.cause**	**VID**	**IRV**	**VPC.full**	**VPC.semi**	**MVC**	**IAV**	**LS.ICV**
BG	50.65	26.67	24.14	87.32	-	-	-	0.00	-
DE	4.17	0.00	24.35	33.77	63.47	0.00	-	-	-
ES	18.03	0.00	6.94	39.22	0.00	-	23.40	31.06	-
FR	61.38	0.00	32.26	78.70	-	-	0.00	-	-
HU	60.00	61.02	62.50	-	74.06	90.24	-	-	-
IT	31.71	20.51	9.59	51.14	57.89	-	33.33	28.07	0.00
PL	53.72	15.38	3.42	82.40	-	-	-	61.90	-
PT	66.56	0.00	21.94	50.70	-	-	-	-	-
RO	68.97	4.65	56.86	85.26	-	-	-	-	-
SL	16.33	0.00	10.11	65.61	-	-	-	44.60	-

Table 5.5. Token-based F1 scores per VMWE category of Deep-BGT.

Lang.	**LVC.full**	**LVC.cause**	**VID**	**IRV**	**VPC.full**	**VPC.semi**	**MVC**	**IAV**	**LS.ICV**
BG	51.45	26.25	31.73	87.53	-	-	-	0.00	-
DE	9.43	0.00	36.62	48.19	67.44	6.25	-	-	-
ES	21.10	0.00	11.05	39.78	0.00	-	33.50	30.86	-
FR	62.67	0.00	59.92	79.35	-	-	0.00	-	-
HU	65.82	66.07	78.57	-	76.27	89.16	-	-	-
IT	37.39	26.67	21.13	52.72	58.23	-	30.77	33.85	0.00
PL	55.90	15.69	32.87	83.25	-	-	-	57.78	-
PT	67.60	0.00	28.77	50.35	-	-	-	-	-
RO	67.23	75.25	73.45	85.69	-	-	-	-	-
SL	21.05	22.22	25.64	66.97	-	-	-	43.77	-

Table 5.6. Model parameters.

Languages	**Batch Size**	**# of Epochs**
BG, FR, HE, LT, PT, RO, TR	32	12
DE, EL, ES, EU, HI, HU	16	15
FA, IT, PL, SL	16	12
EN, HR	8	15

Table 5.7. The performance of Deep-BGT on 19 languages.

	MWE-based			**Token-based**		
Language	**shared task**	**extended**	**improved**	**shared task**	**extended**	**improved**
BG	65.56	65.56	67.03	66.85	66.85	67.72
DE	45.53	45.53	50.75	50.76	50.76	55.10
EL	-	53.44	60.54	-	61.73	66.97
EN	-	31.09	31.60	-	30.57	31.19
ES	28.55	28.55	33.59	35.66	35.66	38.64
EU	-	72.05	72.62	-	75.42	75.03
FA	-	71.44	79.31	-	78.01	81.33
FR	53.51	53.51	61.96	65.80	65.80	64.78
HE	-	28.69	27.45	-	32.56	29.30
HI	-	70.51	73.35	-	72.34	74.78
HR	-	53.94	51.85	-	57.00	54.58
HU	74.48	74.48	74.83	76.72	76.72	76.48
IT	32.77	32.77	38.17	39.62	39.62	42.73
LT	-	21.39	22.85	-	21.34	22.89
PL	63.00	63.00	65.87	67.23	67.23	67.70
PT	56.35	56.35	61.32	57.30	57.30	62.91
RO	74.07	74.07	85.89	81.86	81.86	86.33
SL	46.49	46.49	54.06	51.76	51.76	56.46
TR	-	55.72	55.95	-	56.59	57.52

5.6. Improving Deep-BGT

We realized a minor mistake when a VMWE has more than two tokens and it is discontinuous at the same time. In that case, the middle tokens were tagged as if not belonging to the chunk instead of being part of the chunk. The tagging procedure of discontinuous VMWEs with more than two tokens is corrected accordingly. The results after correction is in the columns labeled with *improved* in Table 5.7. The results are computed by taking the macro-average of F-measures belonging to five separate runs not just a single run as in the shared task. It is seen that the performances increase for most of the languages.

6. A NOVEL TAGGING SCHEME: BIGAPPY-UNICROSSY

6.1. Motivation

While developing the Deep-BGT system, it is seen that the hardest part is to tackle with the challenging nature of MWEs. Especially, an overlapping MWE is an important problem in the case of IOB encoding [2]. The gappy 1-level tagging scheme [15] tries to solve this problem but the problem cannot be accomplished completely. In this thesis, a new tagging scheme called *bigappy-unicrossy* [9] is introduced to rise to the challenge of overlapping MWEs [9]. This novel tagging scheme can be used for other sequence labeling tasks too.

6.2. Challenges

The nature of MWEs brings about some challenges. Discontinuity, overlaps, ambiguity, and variability are the challenges for MWE identification [2]. A MWE can appear in two different forms which are continuous and discontinuous. If a MWE takes a token between its tokens and this token does not belong to the MWE, the MWE is discontinuous [9]. For example, *pay attention* is a continuous VMWE if there is no other tokens in between. When it has a token in between (e.g. *pay much attention*), it becomes discontinuous because the VMWE is still *pay attention* here. Discontinuity varies from language to language [2]. Table 6.1 shows how discontinuity varies from language to language in the test sets.

Overlaps cover different problems such as nesting, shared tokens, and so on [2]. If there is at least one MWE inside an another MWE, it is called nesting [9]. There are two VMWEs in the sentence *She gave me the most over-rated piece of advice*. The first VMWE is *gave advice* which is LVC.full and this VMWE is an ordinary MWE in terms of overlaps. The second VMWE is *over-rated* which is VPC.full and it is

Table 6.1. The percentage of discontinuous VMWEs for each language in the PARSEME corpus.

Language	**Discontinuity %**
BG	29
DE	46
EL	45
EN	41
ES	28
EU	19
FA	21
FR	44
HE	24
HI	7
HR	42
HU	8
IT	33
LT	40
PL	30
PT	43
RO	33
SL	51
TR	59

referred as a nested MWE. If two or more MWEs share one or more tokens, it is called shared tokens [9]. It can be solved by being able to put more than one tag for each token [2]. If preserving only one MWE by eliminating remaining ones that share tokens is acceptable, some other strategies such as choosing the longest MWE [2], choosing the first seen MWE etc. may be created. The sentence *I made both changes and additions* is an example of shared tokens. Here, there are two VMWEs belonging to the category of LVC.full: *made changes* and *made additions*. As it is seen, *made* is the shared token.

Crossing is a special case of overlaps [9]. It occurs when some or all tokens of different MWEs are positioned crosswise [9]. The sentence *I made not only changes but also additions* can be given as an example to both shared tokens, crossing, and nesting. *Made changes* and *made additions* are both MWEs and they share the token *made* as in the previous example. There is also one more MWE belonging to the category of complex function words in this sentence which is *not only but also*. The example sentence contains crossing regarding *made changes* and *not only but also*. Additionally, nesting occurs considering *made additions* and *not only but also*. According to our inference, overlaps brings about the challenge of discontinuity because crossing MWEs are also discontinuous.

Ambiguity stems from the fact that the group of tokens can be a MWE if tokens lose their original meanings or it can be a non-MWE if each token contributes to the sentence with its original meaning [2, 9]. We can consider these two sentences to understand ambiguity thoroughly: *Proceedings will be published in hardcopy as well as softcopy* and *He played as well as he usually does*. The ambiguity arises form *as well as* because it is a complex function word and thereby a MWE in the first sentence but it is only used for comparison in the second one.

Variability originates from the flexible nature of MWEs [2] since MWEs do not always appear in fixed forms [9]. For example, to *give advice* is a VMWE and it can appear in different forms such as *gives advice*, *gave advice* and so on.

6.3. Tagging Schemes

MWE identification is not straightforward due to challenges. In this study, it is attempted to solve the challenges of discontinuity and overlaps that include nesting and crossing by the bigappy-unicrossy tagging scheme [9]. Also, the bigappy-unicrossy tagging scheme is compared with the IOB2 and gappy 1-level tagging schemes. Table 6.2 shows eight example sentences which are tagged with IOB2, gappy 1-level and bigappy-unicrossy tagging schemes separately to facilitate understanding the difference between these tagging schemes.

6.3.1. The IOB1 Tagging Scheme

Ramshaw and Marcus [13] proposed the IOB tagging scheme. In the study, text chunking is interpreted as a tagging problem. The chunk structure is encoded by words together with tags. The tag set of the IOB tagging scheme is {I, O, B}. If the chunk spans more than one token, the beginning of the chunk is represented by B and the remaining of the chunk is represented by I. If the chunk is single-token, I is used. So, B can be used when it is followed by I. O is used for a token outside of any chunk. The IOB tagging scheme has also been referred as the IOB1 tagging scheme since the IOB2 tagging scheme has been introduced [35].

6.3.2. The IOB2 Tagging Scheme

The idea behind the IOB2 scheme is to give B tag for single token-chunks instead of I tag [14, 35]. Therefore, B represents every initial token of the chunk ignoring the chunk size. The tag set of the IOB2 is as same as IOB1 which is {I, O, B}. B represents a token in the beginning of the chunk and thereby single-token chunks. I stands for the remaining part of the chunk other than the initial token. O symbolizes a token outside of the chunk. Hence, I is only used when it follows B. The IOB2 tagging scheme seems to be more convenient than the IOB1 tagging scheme for VMWE identification in terms of single-token VMWEs. Thus the IOB2 tagging scheme is used in the experiments to create a baseline for the other tagging schemes.

Table 6.2. Examples to the tagging schemes.

Example 1	Please	double-check							
IOB2	O	B							
gappy 1-level	O	B							
bigappy-unicrossy	O	B							
Example 2	Let's	have	a	look	at	your	homework		
IOB2	O	B	I	I	O	O	O		
gappy 1-level	O	B	I	I	O	O	O		
bigappy-unicrossy	O	B	I	I	O	O	O		
Example 3	She	didn't	pay	much	attention	to	that		
IOB2	O	O	B	O	I	O	O		
gappy 1-level	O	O	B	o	I	O	O		
bigappy-unicrossy	O	O	B	o	I	O	O		
Example 4	She	gave	me	the	most	over-rated	piece	of	advice
IOB2	O	B	O	O	O	O	O	O	I
gappy 1-level	O	B	o	o	o	b	o	o	I
bigappy-unicrossy	O	B	o	o	o	b	o	o	I
Example 5	I	took	her	decision	to	move	on	seriously	
IOB2	O	B	O	O	O	O	O	I	
gappy 1-level	O	B	o	o	o	b	i	I	
bigappy-unicrossy	O	B	o	o	o	b	i	I	
Example 6	I	took	her	decision	to	make	some	changes	seriously
IOB2	O	B	O	O	O	O	O	O	I
gappy 1-level	O	B	o	o	o	o	o	o	I
bigappy-unicrossy	O	B	o	o	o	b	o	i	I
Example 7	I	made	not	only	changes	but	also	additions	
IOB2	O	B	O	O	O	O	O	I	
gappy 1-level	O	B	o	o	o	o	o	I	
bigappy-unicrossy	O	B	b	i	o	i	i	I	
Example 8	I	made	not	only	changes	but	also	additions	
IOB2	O	B	O	O	I	O	O	O	
gappy 1-level	O	B	o	o	I	O	O	O	
bigappy-unicrossy	O	B	b	i	I	i	i	O	

6.3.3. The Gappy 1-level Tagging Scheme

The IOB1 and the IOB2 tagging schemes focus on continuous chunks. However, both continuous and discontinuous chunks should be taken into consideration for VMWE identification task since VMWEs include both types of chunks. Moreover, MWEs bring about some challenges as explained above. Some of these challenges can be addressed by special tagging schemes. The gappy 1-level tagging scheme [15] is introduced to touch upon discontinuity and nesting.

In the gappy 1-level tagging scheme, the tag set is {I, O, B, i, o, b}. I, O, B tags have similar roles with the ones in the IOB tagging schemes. Schneider *et al.* [15] accept MWEs as chunks containing more than one word. So, the difference between the IOB1 and IOB2 tagging schemes disappears. Consequently, B represents the beginning of the chunk, I represents the remaining part of the chunk, O stands for a token outside of the chunk. All B tags occur together with I tags because single-token MWEs are not accepted by definition [15].

I, O, B tags and i, o, b tags have similar roles in general. If there is nesting, the lowercase tags are used for nested chunks. Otherwise, the uppercase tags are used. b stands for the beginning of the nested chunk, i represents the remaining part of the nested chunk, and o is used for a token outside of the nested chunk which is called a gap for the chunk that wraps the nested chunk. The co-occurrence strategy of b and i tags is applied as in B and I tags.

The gappy 1-level tagging scheme allows only multi-token MWEs as already mentioned before. However, our definition of MWEs involves both single-token and multi-token MWEs. To handle this issue, the gappy 1-level tagging scheme is modified by allowing the single-token MWEs in the IOB2 tagging scheme fashion. b tag is used for nested single-token MWEs. B tag is used for ordinary single-token MWEs. The modified version of the gappy 1-level scheme is used in the experiments and is referred as *gappy 1-level*.

The gappy 1-level tagging scheme solves the discontinuity problem partially since it uses the o tag as a solution but it does not allow discontinuity in the nested chunks. The rejection of discontinuity in the nested chunks is connected with the nesting problem since only continuous nested chunks are allowed. Therefore, the nesting problem is also solved partially.

6.4. The Bigappy-unicrossy Tagging Scheme

The variants of IOB tagging schemes such as gappy 1-level tagging scheme offer partial solutions for the challenges of MWE identification. The IOB2 tagging scheme takes account of continuous chunks but disregards discontinuous chunks. The gappy 1-level tagging scheme takes both continuous and discontinuous chunks into consideration but only allows nested continuous chunks and ignores nested discontinuous chunks. As a result, it proposes partial solutions to the challenges of discontinuity and nesting. Moreover, the other cases of overlaps such as crossing and shared tokens are eliminated. Therefore, it is noticed that a new tagging scheme to cover more of the challenges is a necessity. A novel tagging scheme called *bigappy-unicrossy* [9] is developed to represent overlaps in sequence labeling tasks and to solve the discontinuity problem accordingly.

The tag set of the bigappy-unicrossy tagging scheme is I, O, B, i, o, b. The tag set of the bigappy-unicrossy tagging scheme and the gappy 1-level tagging scheme is the same. The roles of the tags in gappy 1-level overlaps with bigappy-unicrossy. However, the tags have some extra roles in the bigappy-unicrossy tagging scheme. The bigappy-unicrossy tagging scheme allows two levels of discontinuity, one level of nesting, and one level of crossing. The name bigappy-unicrossy is given accordingly.

B represents the beginning of the chunk and single-token chunks. I represents the continuation of the chunk in the case of a multi-token chunk whose beginning is tagged with B. O is used for a token outside of the chunk.

Discontinuous chunks, nested chunks, and chunks with crosswise positioned tokens are handled with the lowercase tags. Tokens that are in between of the chunk

but does not belong to the chunk can be called gaps [15]. o is used for a gap of the chunk to represent the discontinuity in the chunk as in the gappy 1-level. From the perspective of a nested chunk, o stands for a token outside of the nested chunk. In the bigappy-unicrossy tagging scheme, the o tag has an additional role different from the gappy 1-level tagging scheme. The gaps in the nested chunks are also represented by the o tag. In other words, o is used for all gappy chunks including the nested ones and each gap is tagged with o. The gaps are not differentiated as the ones belonging to chunks or nested chunks due to the fact that the tokens symbolizing the gaps have the same role, which is being a token that can be inserted to the chunk without being a part of the chunk. From another point of view, a gap of an ordinary chunk and a gap of a nested chunk do not have specific roles in terms of the type of the chunk.

We can examine the example 6 in Table 6.2 which is *I take her decision to make some changes seriously* to understand the suggestion of treating all the gaps the same. In the example, *take seriously* and *make changes* are VMWEs. The gaps for *take seriously* are *her*, *decision*, *to*, *make*, *some*, and *changes*. Since *make changes* is a nested VMWE here, the actual gaps are *her*, *decision*, *to*, and *some*. The gap for *make changes* is *some*. *some* has the same role for both of the VMWEs. So, all the gaps are tagged with *o*. Eventually, the bigappy-unicrossy tagging scheme allows two levels of discontinuity: one level of discontinuity in the outer chunks and one level of discontinuity in the nested chunks. Gappy 1-level tagging scheme cannot tag the nested VMWEs in the examples 6 and 7 in Table 6.2 since it does not allow discontinuous nested chunks. However, bigappy-unicrossy tags them due to two level of discontinuity.

The lowercase tags are specialized version of the uppercase tags to handle nesting in the gappy 1-level tagging scheme. In the bigappy-unicrossy tagging scheme, the lowercase tags are specialized for not only the nested chunks but also the crossy chunks. b represents the beginning of the nested or crossy chunk. It is also used for single-token ones. *i* stands for the remaining part of the nested or crossy chunk. It is used in the case of multi-token chunks where *b* is followed by one or more *i*. *o* symbolizes the gaps in the nested or crossy chunks as explained beforehand. The example 8 in Table 6.2 shows how crossy chunks can be tagged thanks to bigappy-unicrossy.

The bigappy-unicrossy tagging scheme treats crossing cases and nesting cases similarly due to fact that both cases have the similar identification procedure to some extent. Tagging procedure of the bigappy-unicrossy tagging scheme is shown in Figure 6.1. The similarity between a nested chunk and a crossy chunk is that the beginning index of the nested or crossy chunk Y is larger than the beginning index of chunk X but smaller than the end index of chunk X. The difference between a nested chunk and a crossy chunk stems from the index of the last token and the middle tokens of chunk Y. If the index of the last token of chunk Y is smaller than the index of the last token of chunk X, it is called nesting. If the index of the last token of chunk Y is bigger than the index of the last token of chunk X, it is called crossing. If the middle tokens of chunk X and chunk Y are positioned crosswise in the first case, there is also crossing.

There are two types of tag sets which are uppercase and lowercase in the bigappy-unicrossy tagging scheme. Therefore, bigappy-unicrossy supports only one level of nesting or one level of crossing. The tagging procedure starts with using uppercase tags. When an uppercase or a lowercase tag set is in use to tag a chunk, it is held until the end of the chunk. So, the tag set cannot be used for another chunk till it is released. After the last token of the chunk, the tag set is released.

To sum up, the bigappy-unicrossy tagging scheme solves the discontinuity problem in two levels. The challenges of crossing and nesting are overcomed in one level. Incorporating more levels is not necessary since such a case is very scarce. So, the overlapping nature of the MWEs is handled partially.

6.5. Model and Experiments

The bigappy-unicrossy tagging scheme is compared with the two other well-known tagging schemes which are IOB2 and gappy 1-level in the verbal multiword expression (VMWE) identification task using the same bidirectional LSTM-CRF model in the improved version of the Deep-BGT system explained in Chapter 5.

for each sentence s in the text **do**
  for each token t in s **do**
    if (t belongs to the beginning of the chunk X **and** X is the first chunk in s) **or** (t belongs to the beginning of the chunk X **and** t comes after the last token of the previous chunk which is tagged with uppercase tags) **then**
      tag t with ***B***
      if X is multi-token **then**
        Find the remaining tokens and tag the tokens with ***I***
        Store the beginning index idx_b and the end index idx_e of X in the list l
      end if
    else if t belongs to the beginning of the nested/crossy chunk Y **and** (Y is the first nested/crossy chunk in s **or** t comes after the last token of the previous chunk which is tagged with lowercase tags) **then**
      tag t with ***b***
      if Y is multi-token **then**
        Find the remaining tokens and tag the tokens with ***i***
        Store the beginning index idx_b and the end index idx_e of Y in the list l
      end if
    end if
  end for
  for each (idx_b,idx_e) in l **do**
    for each token g between idx_b and idx_e **do**
      if g is not tagged **then**
        tag g with ***o***
      end if
    end for
  end for
end for
filter the tokens with no tags in the text
tag them with ***O***

Figure 6.1. Bigappy-unicrossy Tagging Algorithm.

As we use non-deterministic approach, we run our experiments five times in order to maintain reproducible and reliable results and take the average. The aim is to present an empirical study that explores the effect of a tagging scheme for VMWE identification on 19 languages.

6.6. Results

Table 6.3 and Table 6.4 shows the language-specific results on 19 languages for the IOB2, the gappy 1-level and the bigappy-unicrossy tagging schemes based on F-measure respectively. The highest F1 scores obtained for each language in the open track of the PARSEME shared task 2018 is also given for comparison and it is referred as *shared task*. Also, the cross-lingual macro-averages are given in the last rows of the tables. Figure 6.2 also depicts the MWE-based results of the tagging schemes and the best results in the shared task.

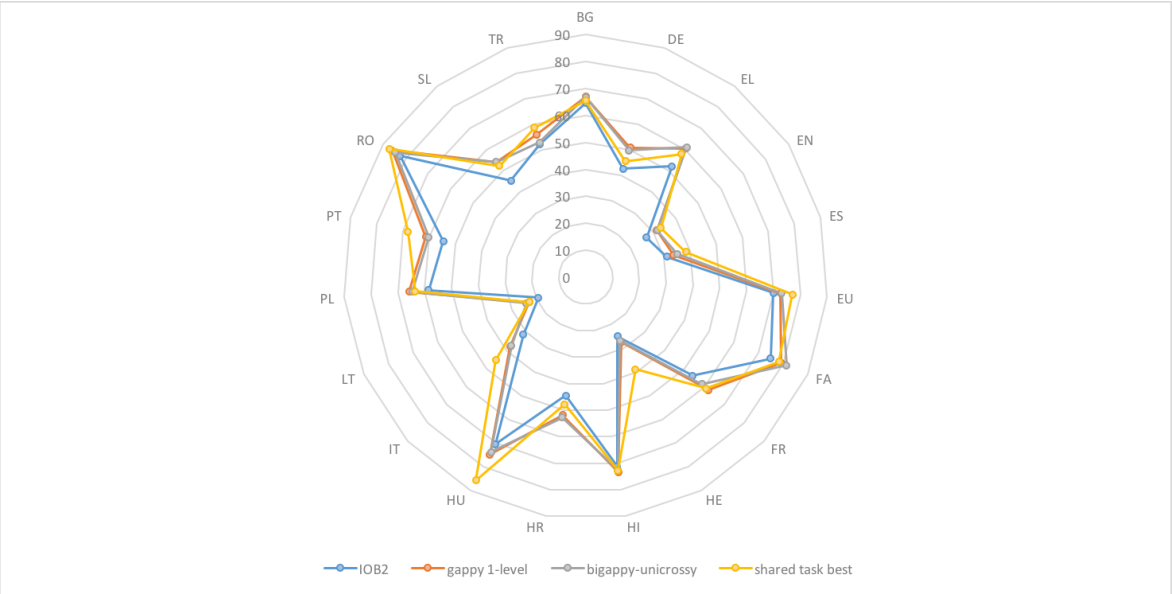

Figure 6.2. Comparison of the tagging schemes.

Both the bigappy-unicrossy and the gappy 1-level tagging schemes outperform the IOB2 tagging scheme for all 19 languages in terms of MWE-based results. They also slightly outperform IOB2 according to token-based macro-averages. This supremacy takes its source from the fact that they consider discontinuous VMWEs unlike IOB2.

Table 6.3. The language-specific MWE-based results for the IOB2, the gappy 1-level, the bigappy-unicrossy tagging schemes and the best PARSEME shared task results in the open track.

Language	**IOB2**	**gappy 1-level**	**bigappy-unicrossy**	**shared task**
BG	64.60	67.03	66.89	65.56
DE	42.62	50.75	49.73	45.53
EL	52.10	60.54	61.11	58.00
EN	26.97	31.60	31.73	33.27
ES	31.07	33.59	35.00	38.39
EU	69.91	72.62	73.07	77.04
FA	75.07	79.31	81.37	78.35
FR	53.92	61.96	58.55	60.88
HE	24.93	27.45	26.74	38.91
HI	71.28	73.35	72.54	72.71
HR	44.62	51.85	52.83	47.84
HU	70.53	74.83	73.84	85.83
IT	31.52	38.17	37.58	45.40
LT	19.15	22.85	24.04	22.86
PL	58.54	65.87	64.65	63.60
PT	54.62	61.32	60.21	68.17
RO	82.34	85.89	84.60	87.18
SL	45.30	54.06	54.22	52.27
TR	52.26	55.95	52.93	58.66
AVG	*51.12*	*56.26*	*55.88*	*57.92*

Table 6.4. The language-specific token-based results for the IOB2, the gappy 1-level, the bigappy-unicrossy tagging schemes and the best PARSEME shared task results in the open track.

Language	**IOB2**	**gappy 1-level**	**bigappy-unicrossy**	**shared task**
BG	67.21	67.72	67.24	66.85
DE	54.28	55.10	53.31	54.65
EL	63.73	66.97	65.61	66.79
EN	30.15	31.19	30.86	34.36
ES	37.54	38.64	39.76	44.69
EU	75.38	75.03	76.06	80.21
FA	82.01	81.33	84.48	82.95
FR	65.05	64.78	61.57	65.80
HE	28.56	29.30	28.74	44.02
HI	74.06	74.78	74.35	75.62
HR	53.68	54.58	56.18	58.19
HU	73.90	76.48	76.13	86.73
IT	40.28	42.73	43.61	55.13
LT	25.31	22.89	24.49	28.13
PL	64.78	67.70	66.41	67.23
PT	62.29	62.91	62.39	73.51
RO	85.31	86.33	85.19	88.69
SL	56.30	56.46	57.50	61.55
TR	58.12	57.52	54.30	61.63
AVG	*57.79*	*58.55*	*58.33*	*62.99*

Figure 6.3 shows the relationship between the discontinuity rates of VMWEs for all languages and the relative success of the bigappy-unicrossy tagging scheme over the IOB2 tagging scheme. Discontinuity percentages were given previously in Table 6.1. The relative success is found by subtracting the MWE-based F1 score of bigappy-unicrossy from that of IOB2. It is seen that the discontinuity ratio and the improvement in the success with the bigappy-unicrossy scheme are correlated to some extent. This result verifies the claim that bigappy-unicrossy outperforms IOB2 since it captures discontinuous VMWEs.

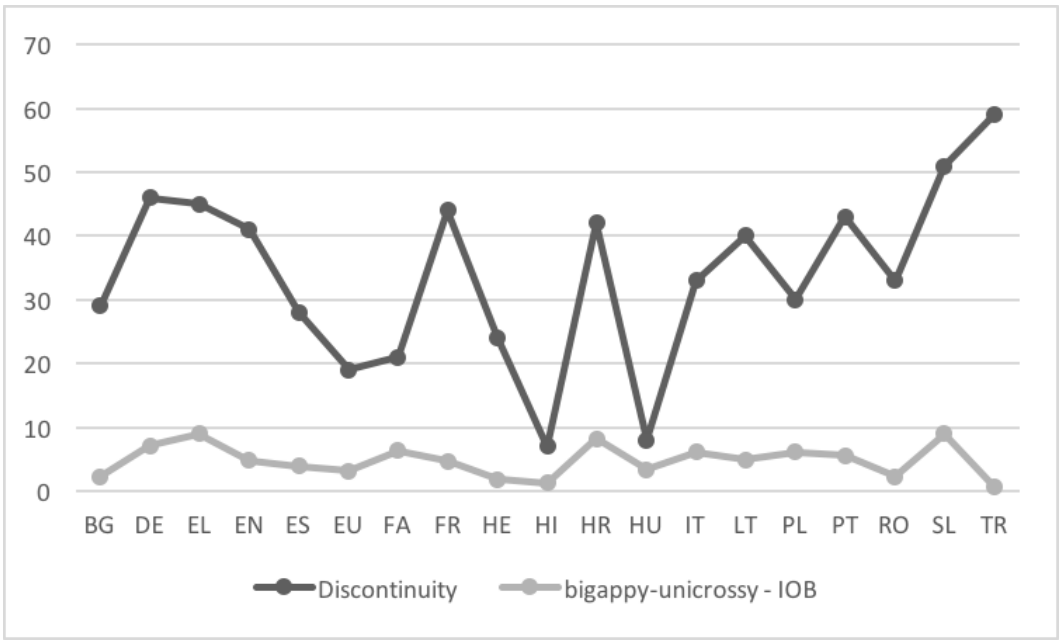

Figure 6.3. The relationship between the relative success of bigappy-unicrossy over IOB2 and the discontinuity rates of VMWEs.

On the contrary, the results of gappy 1-level and bigappy-unicrossy is close to each other. There is only a slight difference of 0.38. The gappy 1-level tagging scheme gives better results in 11 languages while the bigappy-unicrossy tagging scheme gives better results in 8 languages. The underlying reason behind this closeness is the low frequency of overlaps in the PARSEME corpus. The corpus only covers VMWEs. If other types of MWEs are also covered, the overlap frequency will be higher and the bigappy-unicrossy tagging scheme will its actual performance.

Furthermore, the bigappy-unicrossy tagging scheme can be used in different sequence labeling tasks other than MWE identification. It can prove itself better in the other domains of NLP or in their combinations.

The BiLSTM-CRF model together with the gappy 1-level tagging scheme or the bigappy-unicrossy tagging scheme competes with the MWE-based best shared task results as shown in Figure 6.2 and Table 6.3. The BiLSTM-CRF model combined with the gappy 1-level tagging scheme is the best system in five languages consisting of BG, DE, FR, HI, PL. The BiLSTM-CRF model combined with the bigappy-unicrossy tagging scheme is the best system in five languages consisting of EL, FA, HR, LT, SL. Both of the tagging schemes obtain better results than the best shared task results in BG, DE, EL, FA, HR, PL, SL. In the case of token-based results, gappy 1-level is the best in BG, DE, EL, PL and bigappy-unicrossy is the best in FA. Token-based results seems to be higher than the MWE-based results as expected. However, token-based results are not high enough to get ahead. Since the identification using deep learning systems becomes more complex if the tagging scheme becomes more complex, the token-based results of all three tagging schemes in Table 6.4 are similar. Above all, the most valuable results are MWE-based results just because identification of the whole is more important than the identification in pieces.

7. A COMPREHENSIVE ANALYSIS OF BILSTM BASED ARCHITECTURES FOR MULTILINGUAL IDENTIFICATION OF VMWES

7.1. Motivation

Previously, a brand new tagging scheme called bigappy-unicrossy has been developed to augment the performance of the system. Tagging scheme is just one parameter of the deep learning architecture. There are various hyperparameters to be fine tuned. The selection of optimal hyperparameters is vital for neural network architectures [27]. It goes without saying that trying out every possible combination of parameters is nearly impossible. So, a subset of parameters are selected to conduct experiments for ensuring the optimization of the system. Thus, a comprehensive analysis of BiLSTM based architectures for multilingual identification of VMWEs is presented.

7.2. Hyperparameter Selection and Evaluation Strategy

In Reimers and Gurevych's study [27], the hyperparameter set consists of pre-trained word embeddings, character representation, optimizer, gradient clipping and normalization, tagging schemes, classifier, dropout, number of LSTM layers, number of recurrent units, mini-batch size and backend. In this thesis, tagging schemes, number of units, number of BiLSTM layers and classifier are selected for evaluation.

The identification system is non-deterministic. Thereby, there is randomness in some parts of the system. To present reliable results, the experiments are run couple times (three or five runs) and the averages of F1 scores are used. The median is not used since the number of runs is not high enough which stems from the time and the processing power constraints.

7.3. Experiments and Results

The idea is not only comparing the effects of the hyperparameters but also finding the best combination of the hyperparameters. The BiLSTM-CRF network shown in Figure 5.1 is simplified to a BiLSTM network shown in Figure 7.1 with a Softmax classifier. The optimum model will be constructed gradually by choosing the hyperparameters one after the other. Firstly, the tagging scheme is selected. Afterwards, the number of units that fits best with the tagging scheme is chosen. Then, the number of BiLSTM layers is chosen accordingly. Finally, the classifier is evaluated.

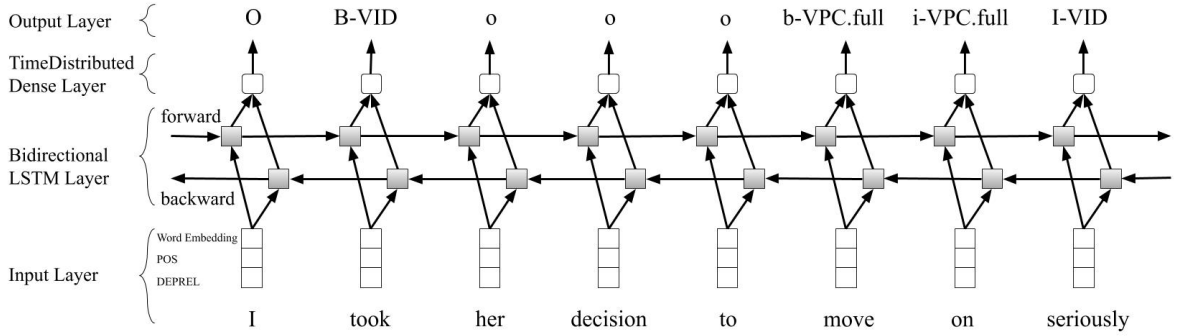

Figure 7.1. BiLSTM model.

7.3.1. Tagging Scheme

The gappy 1-level and the bigappy-unicrossy tagging schemes are compared using BiLSTM and BiLSTM-CRF architectures. The results of the BiLSTM-CRF model is taken from the studies that were explained previously. The base BiLSTM architecture is also constructed with 20 and 100 BiLSTM units separately. As a result, the tagging schemes are compared under three different combinations. Table 7.1 and Table 7.2 show the MWE-based and token-based comparison between these two tagging schemes respectively. According to the results, gappy 1-level and bigappy-unicrossy compete with each other. Since the bigappy-unicrossy is created as part of this thesis, it is favored. Therefore, the experiments continue with the biggapy-unicrossy tagging scheme.

Table 7.1. The language-specific MWE-based results for the gappy 1-level and the bigappy-unicrossy tagging schemes with different models.

Tagging Scheme	**gappy 1-level**			**bigappy-unicrossy**		
Model	**BiLSTM**		**BiLSTM-CRF**	**BiLSTM**		**BiLSTM-CRF**
# of Units / **Languages**	**20**	**100**	**20**	**20**	**100**	**20**
BG	57.50	60.02	67.03	55.82	59.56	66.89
DE	32.19	42.55	50.75	32.66	45.31	49.73
EL	45.62	50.98	60.54	44.72	46.04	61.11
EN	18.83	17.08	31.60	17.01	24.94	31.73
ES	20.15	27.19	33.59	22.58	25.63	35.00
EU	61.96	64.44	72.62	61.86	63.94	73.07
FA	73.53	72.84	79.31	74.67	74.46	81.37
FR	49.28	49.90	61.96	47.15	51.60	58.55
HE	13.06	22.96	27.45	11.27	19.91	26.74
HI	62.70	65.74	73.35	67.33	65.95	72.54
HR	36.76	40.07	51.85	38.88	40.04	52.83
HU	60.85	67.20	74.83	59.97	68.25	73.84
IT	20.22	25.91	38.17	21.37	26.12	37.58
LT	0.76	21.86	22.85	5.00	18.67	24.04
PL	50.16	52.29	65.87	52.53	54.38	64.65
PT	47.71	54.02	61.32	45.17	47.03	60.21
RO	77.75	82.33	85.89	76.44	80.77	84.60
SL	32.92	45.17	54.06	26.23	44.83	54.22
TR	50.37	51.99	55.95	45.20	48.29	52.93
AVG	*42.75*	*48.13*	*56.26*	*42.41*	*47.67*	*55.88*

Table 7.2. The language-specific token-based results for the gappy 1-level and the bigappy-unicrossy tagging schemes with different models.

Tagging Scheme	**gappy 1-level**			**bigappy-unicrossy**		
Model	**BiLSTM**		**BiLSTM-CRF**	**BiLSTM**		**BiLSTM-CRF**
# of Units / **Languages**	**20**	**100**	**20**	**20**	**100**	**20**
BG	64.82	66.41	67.72	63.50	66.29	67.24
DE	47.84	55.10	55.10	49.62	53.45	53.31
EL	63.75	65.45	66.97	64.65	64.74	65.61
EN	24.30	25.22	31.19	23.64	32.20	30.86
ES	34.54	39.58	38.64	36.82	37.95	39.76
EU	73.46	73.59	75.03	74.13	73.80	76.06
FA	82.15	80.86	81.33	81.15	83.32	84.48
FR	65.93	65.77	64.78	62.54	66.26	61.57
HE	22.78	32.44	29.30	18.86	27.27	28.74
HI	71.19	72.48	74.78	74.40	71.99	74.35
HR	50.81	54.26	54.58	51.92	53.60	56.18
HU	67.61	73.25	76.48	67.28	74.61	76.13
IT	37.26	43.08	42.73	40.31	42.81	43.61
LT	2.96	34.47	22.89	16.40	29.62	24.49
PL	61.19	64.45	67.70	64.94	65.75	66.41
PT	61.29	66.26	62.91	58.55	60.84	62.39
RO	84.84	86.62	86.33	84.19	85.59	85.19
SL	47.39	57.10	56.46	46.19	57.55	57.50
TR	60.10	60.65	57.52	52.71	57.18	54.30
AVG	*53.91*	*58.79*	*58.55*	*54.30*	*58.15*	*58.33*

7.3.2. Number of Units

There are so many possibilities in the case of number of units. The optimal number of units is not obvious because it depends on the architecture, the data, the task and so on. If the number of units is very small, the system loses information. If it is very large, overfitting occurs [27].

The set of the number of units is chosen as {20, 100, 200, 300}. These are the number of units per LSTM network. A BiLSTM network consists of one forward and one reverse running LSTM network which have same number of units [27]. Therefore, the number of units per LSTM network is multiplied by two to find the total number of units in a BiLSTM network. In the case of multiple BiLSTM layers, the number of units per BiLSTM network is multiplied by the number of BiLSTM layers since all BiLSTM layers in our system use the same hyperparameters.

Table 7.3 shows the language specific MWE-based and token-based F1 scores for 20, 100, 200, and 300 number of units respectively. Reimers and Gurevych's study [27] states that the number of recurrent units does not have a significant effect on the results of part-of-speech tagging, chunking, named entity recognition, entities, and events tasks. However, the number of units has a significant impact on the VMWE identification task in terms of the results in Table 7.3.

According to the experiments, using more than 20 units obviously improves the results more than 4%. Since each language has different properties, the effect of the number of units differ from language to language. Selection of 100 units suits the languages from Balto-Slavic and Germanic language families well. Languages from Romance language family have higher performance with 200 or 300 units. Owing to the fact that the aim is to develop a multilingual system, the priority is the macro-averaged results. In respect of MWE-based and token-based cross-lingual macro-averages, the number of units is selected as 100 because it gives the highest results with 47.67 and 58.15 F1 scores.

Table 7.3. The language-specific results for different number of units.

	MWE-based				**Token-based**			
# of Units / **Language**	**20**	**100**	**200**	**300**	**20**	**100**	**200**	**300**
BG	55.82	59.56	58.62	55.68	63.50	66.29	66.08	63.70
DE	32.66	45.31	36.83	36.90	49.62	53.45	50.92	52.20
EL	44.72	46.04	46.36	46.24	64.65	64.74	62.56	61.79
EN	17.01	24.94	18.09	21.59	23.64	32.20	24.64	28.57
ES	22.58	25.63	27.11	26.28	36.82	37.95	39.02	37.84
EU	61.86	63.94	63.00	64.00	74.13	73.80	73.49	73.32
FA	74.67	74.46	70.12	73.47	81.15	83.32	79.05	81.54
FR	47.15	51.60	49.03	52.47	62.54	66.26	64.79	65.79
HE	11.27	19.91	20.30	18.82	18.86	27.27	28.98	28.68
HI	67.33	65.95	69.59	68.11	74.40	71.99	74.62	74.49
HR	38.88	40.04	37.44	40.46	51.92	53.60	49.97	52.03
HU	59.97	68.25	66.67	69.74	67.28	74.61	72.93	74.77
IT	21.37	26.12	28.51	25.74	40.31	42.81	46.64	44.32
LT	5.00	18.67	19.97	20.52	16.40	29.62	26.45	31.99
PL	52.53	54.38	54.25	53.64	64.94	65.75	64.44	65.33
PT	45.17	47.03	50.16	52.05	58.55	60.84	64.12	63.51
RO	76.44	80.77	80.86	81.86	84.19	85.59	86.23	86.21
SL	26.23	44.83	43.89	42.16	46.19	57.55	56.62	54.97
TR	45.20	48.29	45.15	44.03	52.71	57.18	52.88	53.73
AVG	*42.41*	*47.67*	*46.63*	*47.04*	*54.30*	*58.15*	*57.07*	*57.62*

7.3.3. Number of BiLSTM Layers

The BiLSTM network that have been used so far has one BiLSTM layer. The aim is to explore the effect of using stacked BiLSTM. Therefore, the stacked BiLSTM networks with 1 (Figure 7.1), 2 (Figure 7.2), and 3 (Figure 7.3) stacked BiLSTM layers are compared.

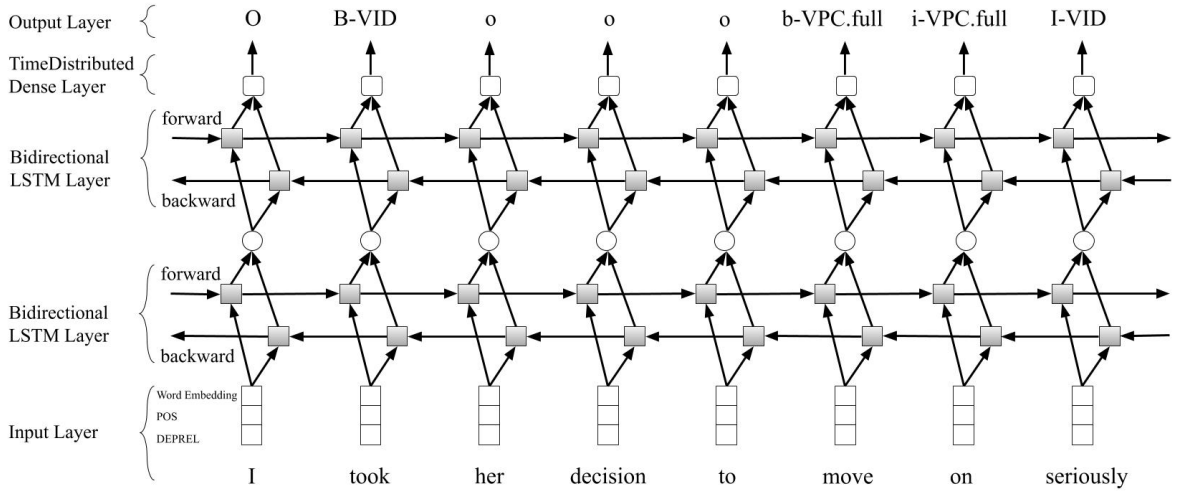

Figure 7.2. 2 layer stacked BiLSTM model.

Table 7.4 shows the language-specific MWE-based and token-based results of 1, 2, and 3 layer stacked BiLSTM models. As the number of stacked BiLSTM layers increases, both of the MWE-based and the token-based F1 scores increase generally. There is a remarkable improvement from 1 layer stacked BiLSTM to 2 layer stacked BiLSTM. The BiLSTM network with 3 stacked BiLSTM-layers gives the best results in terms of cross-lingual macro-averages. Hence, 3 is selected as the number of BiLSTM layers.

7.3.4. Classifier

At the final stage, the decision is to choose the last layer of the network. Actually, there are so many options in terms of activation functions but we reduce the choices to two: a dense layer with Softmax activation function or a dense layer with a linear ac-

Table 7.4. The language-specific results for different number of BiLSTM layers.

	MWE-based			**Token-based**		
# of Layers / **Language**	**1**	**2**	**3**	**1**	**2**	**3**
BG	59.56	62.69	61.98	66.29	68.35	66.82
DE	45.31	43.80	47.19	53.45	54.03	56.22
EL	46.04	55.88	57.86	64.74	65.16	67.09
EN	24.94	26.62	25.49	32.20	31.28	28.61
ES	25.63	29.59	29.29	37.95	39.65	37.90
EU	63.94	71.42	74.51	73.80	75.95	77.81
FA	74.46	78.28	77.67	83.32	83.27	83.35
FR	51.60	58.72	60.96	66.26	67.53	68.46
HE	19.91	24.11	28.98	27.27	34.17	36.19
HI	65.95	67.82	69.64	71.99	73.34	73.93
HR	40.04	45.99	48.47	53.60	55.47	54.43
HU	68.25	73.18	74.50	74.61	77.60	77.91
IT	26.12	34.69	36.94	42.81	47.25	48.57
LT	18.67	24.18	27.25	29.62	30.11	32.97
PL	54.38	59.36	62.76	65.75	65.60	66.70
PT	47.03	58.74	59.42	60.84	65.93	65.46
RO	80.77	84.87	84.57	85.59	87.75	86.78
SL	44.83	52.76	53.50	57.55	60.18	60.15
TR	48.29	57.00	59.78	57.18	63.81	64.78
AVG	*47.67*	*53.14*	*54.78*	*58.15*	*60.34*	*60.74*

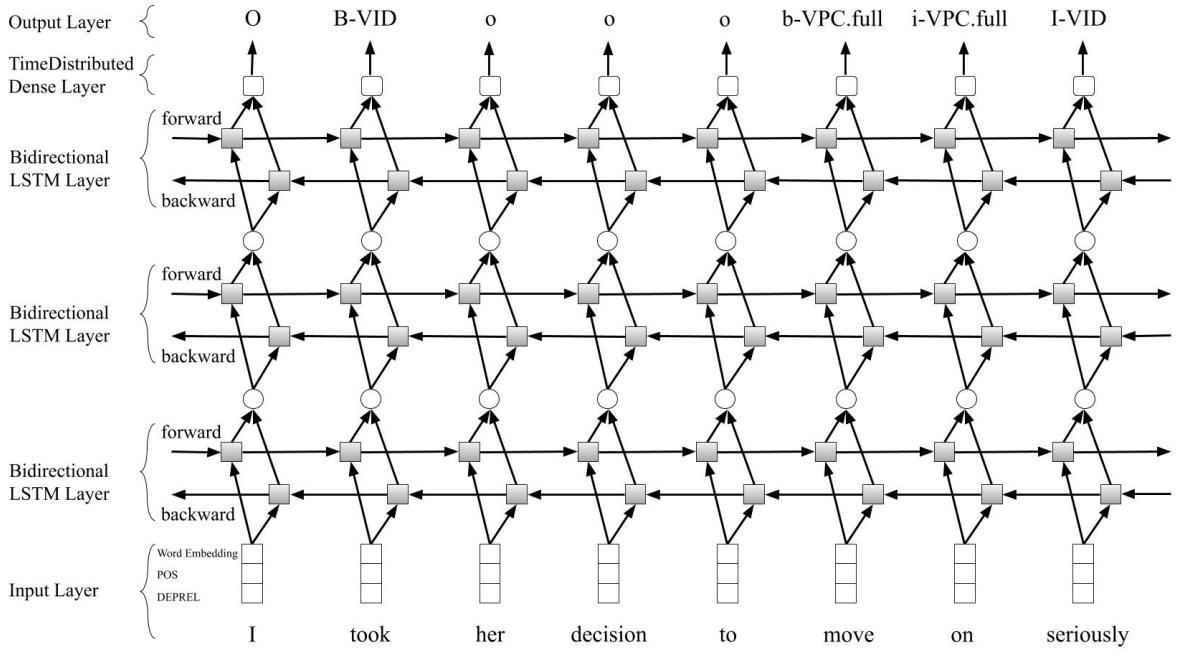

Figure 7.3. 3 layer stacked BiLSTM model.

tivation function followed by a linear-chain CRF by following Reimers and Gurevych's study [27]. In the Softmax classifier, each tag has a probability distribution regarding each token in a sentence [27]. Therefore, each token is evaluated individually apart from other tokens and there is no link between the tags. In the CRF classifier, the tags of the sentence have a probability distribution as a whole and the aim is to maximize this probability [27]. Thus, the relationship between tags is taken into consideration.

When a CRF classifier is combined with a BiLSTM network for sequence tagging tasks, not only past and future input features but also sentence level tag information is used [19]. There is a strong dependency between tags of the tokens in the case of VMWE identification and this dependency makes the CRF classifier suitable.

In the experiments, two different models are compared. The first model is a 3 layer stacked BiLSTM combined with a Softmax classifier. The second one is 3 layer stacked BiLSTM combined with a CRF classifier shown in Figure 7.4. Table 7.5 shows the MWE-based and token based F1 scores of these two models.

Table 7.5. The language-specific results for different classifiers.

	MWE-based		**Token-based**	
Classifier / **Language**	**Softmax**	**CRF**	**Softmax**	**CRF**
BG	61.98	67.06	66.82	68.06
DE	47.19	53.65	56.22	56.78
EL	57.86	61.99	67.09	65.28
EN	25.49	30.96	28.61	33.06
ES	29.29	35.30	37.90	39.16
EU	74.51	75.90	77.81	77.62
FA	77.67	81.00	83.35	83.29
FR	60.96	63.82	68.46	66.08
HE	28.98	31.67	36.19	33.62
HI	69.64	71.79	73.93	73.76
HR	48.47	51.08	54.43	54.82
HU	74.50	76.35	77.91	78.52
IT	36.94	37.80	48.57	43.12
LT	27.25	30.54	32.97	30.65
PL	62.76	66.31	66.70	66.38
PT	59.42	63.80	65.46	65.31
RO	84.57	85.31	86.78	85.95
SL	53.50	57.23	60.15	59.85
TR	59.78	58.29	64.78	58.88
AVG	*54.78*	*57.89*	*60.74*	*60.01*

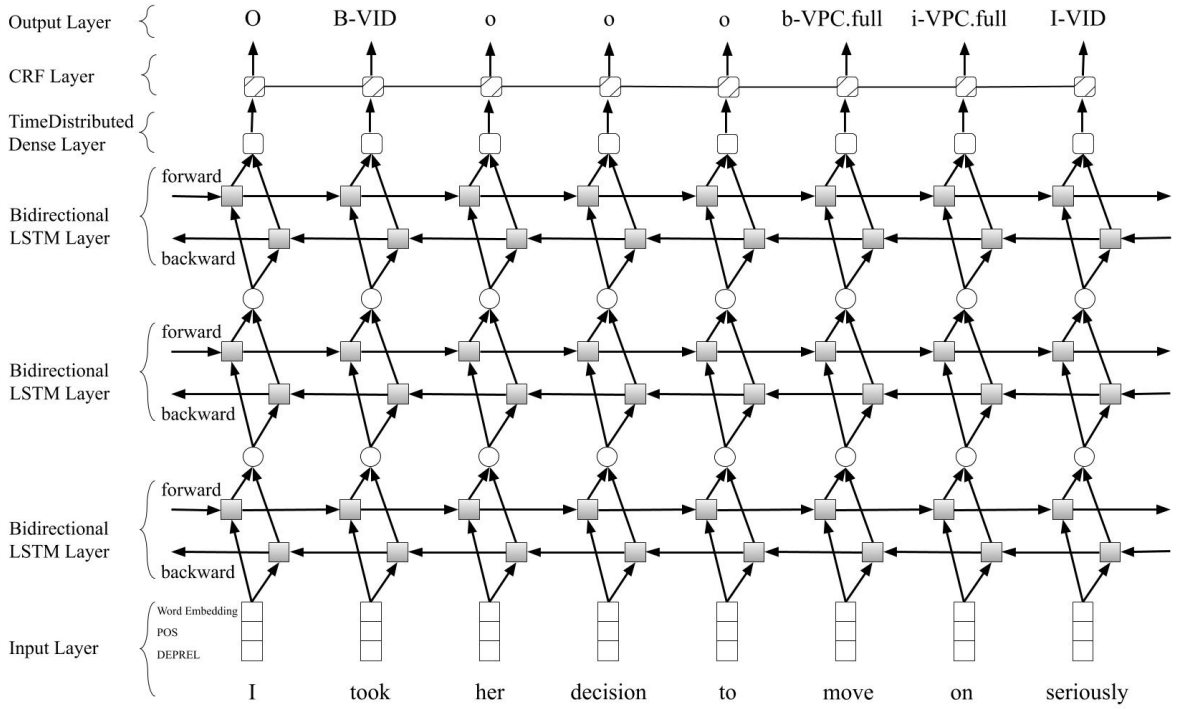

Figure 7.4. 3 layer stacked BiLSTM-CRF model.

The CRF classifier is more successful than the Softmax one in terms of MWE-based results since it tries to maximize the probability of tags in the sentence as a whole. The Softmax classifier seems better than the CRF one according to token-based results because it tries to maximize the probability of each tag individually.

7.4. The Overall Result

Table 7.6 shows the MWE-based and token-based cross-lingual macro-averaged F1 scores of all models. It is clearly seen that the models with the CRF classifier have higher performance in terms of MWE-based scores. The best model according to the MWE-based results includes the bigappy-unicrossy tagging scheme, 100 number of units, 3 layer stacked BiLSTM, and the CRF classifier. The models with 20 number of units, 1 layer stacked BiLSTM, and the CRF classifier follows the best model. It proves the fact that the CRF classifier makes a significant difference. The reason behind this fact is that CRF connects the tags of the tokens in the sentence and maximizes the

probability of the tags accordingly which are necessary since VMWEs consist of more than one token generally.

Table 7.6. The overall results.

Network	**Tagging Scheme**	**# of Units**	**# of Layers**	**Classifier**	**MWE-based F1**	**Token-based F1**
BiLSTM	IOB2	20	1	CRF	51.12	57.79
BiLSTM	gappy 1-level	20	1	CRF	56.26	58.55
BiLSTM	bigappy-unicrossy	20	1	CRF	55.88	58.33
BiLSTM	gappy 1-level	20	1	Softmax	42.75	53.91
BiLSTM	gappy 1-level	100	1	Softmax	48.13	58.79
BiLSTM	bigappy-unicrossy	20	1	Softmax	42.41	54.30
BiLSTM	bigappy-unicrossy	100	1	Softmax	47.67	58.15
BiLSTM	bigappy-unicrossy	200	1	Softmax	46.63	57.07
BiLSTM	bigappy-unicrossy	300	1	Softmax	47.04	57.62
BiLSTM	bigappy-unicrossy	100	2	Softmax	53.14	60.34
BiLSTM	bigappy-unicrossy	100	3	Softmax	54.78	60.74*
BiLSTM	bigappy-unicrossy	100	3	CRF	57.89*	60.01

The model with the bigappy-unicrossy tagging scheme, 100 number of units, 3 layer stacked BiLSTM, and the Softmax classifier obtains the best token-based F1 score. It is followed by the model with 2 layer stacked BiLSTM with the Softmax classifier and 3 layer stacked BiLSTM with the CRF classifier. Firstly, we can infer that the Softmax classifier is more suitable for token-based evaluations since it treats the tag of each token individually. Moreover, the number of layers has an important role on token-based results.

According to both MWE-based and token-based results, as the number of layers increases, the model advances. In addition, switching from 20 number of units to 100 number of units evidently improves the performance. To conclude, the aim was to improve the Deep-BGT system by fine tuning the hyperparameters. It is clearly seen that the objective is accomplished as expected.

8. CONCLUSION AND FUTURE WORK

In the scope of the PARSEME shared task edition 1.1, a Turkish corpus with annotated VMWEs is constructed. The Turkish corpus is one of the biggest corpora in the shared task. In the annotation process, it is observed that the annotation guidelines provided by various sources differ for Turkish. Due to the deficiency in Turkish MWE sources, the annotated corpus will be a valuable source for studies on Turkish natural language processing such as syntactic parsing, machine translation and n-gram language modeling. Not content with VMWEs, the corpora can be extended to MWEs.

A language-independent and deep learning based system called *Deep-BGT* is developed for the second part of the PARSEME shared task edition 1.1. The sequence tagging approach is followed for VMWE identification. Deep-BGT is a hybrid system which uses the BiLSTM-CRF model along with the gappy 1-level tagging scheme. To the best of our knowledge, Deep-BGT is the first bidirectional LSTM-CRF model used in the VMWE identification task. The challenges of discontinuity and overlaps are attempted to be solved.

Deep-BGT is based on a deep learning architecture. Thereby, the more training data is available, the more the system learns. The occurrence frequency of VMWEs in the data is also a significant factor. Therefore, Deep-BGT submitted results for 10 languages accordingly. The system was ranked first in two languages in terms of MWE-based results. It was ranked first in three languages according to token-based results. In general ranking, Deep-BGT is ranked second in the open-track of the shared task. Afterwards, Deep-BGT is improved and extended to 19 languages.

The Deep-BGT system attempted to solve the challenges of discontinuity and overlaps by the gappy 1-level tagging scheme. However, gappy 1-level is not adequate for this objective. Hence, a novel tagging scheme called *bigappy-unicrossy* is introduced to represent overlaps in sequence labeling tasks.

An empirical study that explores the effect of a tagging scheme for VMWE identification on 19 languages by using BiLSTM-CRF network is presented. The bigappy-unicrossy tagging scheme competes with the gappy 1-level tagging scheme. The bigappy-unicrossy tagging scheme is expected to be by far the best scheme on data sets with higher frequency of overlaps. On this basis, the bigappy-unicrossy tagging scheme can be applied on such data sets as future work.

To enhance the performance of the Deep-BGT system, a subset of hyperparameters is chosen to be fine tuned. The subset consists of tagging scheme, number of units, number of BiLSTM layers, and classifier. As a consequence of the evaluation of the hyperparameters, a comprehensive analysis of BiLSTM based architectures for multilingual identification of VMWEs is presented.

The 3 layer stacked BiLSTM network with 100 number of units, the bigappy-unicrossy tagging scheme, and the CRF classifier gives the best performance in terms of MWE-based F1 scores. The 3 layer stacked BiLSTM network with 100 number of units, the bigappy-unicrossy tagging scheme, and the Softmax classifier gives the best performance according to token-based F1 scores.

Different kind of hyperparameters can be evaluated as future work. In addition, the number of BiLSTM layers can be increased since there is a rise in the performance of the system with more BiLSTM layers.

Novel methodologies can be explored to address the challenges of MWE identification. These solutions can be related to preprocessing, postprocessing, or deep learning architecture. Finally, this study can be expanded to the other sequence labeling tasks as future work.

REFERENCES

1. Sag, I. A., T. Baldwin, F. Bond, A. Copestake and D. Flickinger, "Multiword Expressions: A Pain in the Neck for NLP", A. Gelbukh (Editor), *Computational Linguistics and Intelligent Text Processing*, pp. 1–15, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

2. Constant, M., G. Eryiğit, J. Monti, L. Van Der Plas, C. Ramisch, M. Rosner and A. Todirascu, "Multiword expression processing: a survey", *Computational Linguistics*, Vol. 43, No. 4, pp. 837–892, 2017.

3. Baldwin, T. and S. N. Kim, "Multiword expressions.", *Handbook of natural language processing*, Vol. 2, pp. 267–292, 2010.

4. Ramisch, C. *et al.*, "Edition 1.1 of the PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG 2018)*, Association for Computational Linguistics, Santa Fe, New Mexico, USA, August 2018.

5. Savary, A. *et al.*, "The PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions", *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pp. 31–47, Association for Computational Linguistics, Valencia, Spain, Apr. 2017, `https://www.aclweb.org/anthology/W17-1704`.

6. Berk, G., B. Erden and T. Güngör, "Turkish verbal multiword expressions corpus", *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, May 2018.

7. Ramisch, C. *et al.*, *Annotated corpora and tools of the PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions (edition 1.1)*, 2018, `http://hdl.handle.net/11372/LRT-2842`, LINDAT/CLARIN digital library at

the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

8. Berk, G., B. Erden and T. Güngör, "Deep-BGT at PARSEME Shared Task 2018: Bidirectional LSTM-CRF Model for Verbal Multiword Expression Identification", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 248–253, Association for Computational Linguistics, Santa Fe, New Mexico, USA, Aug. 2018, https://www.aclweb.org/anthology/W18-4927.

9. Berk, G., B. Erden and T. Güngör, "Representing Overlaps in Sequence Labeling Tasks with a Novel Tagging Scheme: bigappy-unicrossy", A. Gelbukh (Editor), *Computational Linguistics and Intelligent Text Processing*, Springer International Publishing, 2019 (to appear).

10. Rosén, V., K. De Smedt, G. S. Smørdal Losnegaard, E. Bejček, A. Savary and S. Osenova, "MWEs in Treebanks: From Survey to Guidelines", *Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, May 2016, https://hal.archives-ouvertes.fr/hal-01505051.

11. Adalı, K., T. Dinç, M. Gokirmak and G. Eryigit, "Comprehensive annotation of multiword expressions for Turkish", *Proceedings of TurCLing*, pp. 60–66, 2016.

12. Savary, A. *et al.*, "PARSEME – PARSing and Multiword Expressions within a European multilingual network", *7th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC 2015)*, Poznań, Poland, Nov. 2015, https://hal.archives-ouvertes.fr/hal-01223349.

13. Ramshaw, L. and M. Marcus, "Text Chunking using Transformation-Based Learning", *Third Workshop on Very Large Corpora*, 1995, http://aclweb.org/anthology/W95-0107.

14. Ratnaparkhi, A., *Maximum Entropy Models for Natural Language Ambiguity Resolution*, Ph.D. Thesis, University of Pennsylvania, Philadelphia, PA, USA, 1998, aAI9840230.

15. Schneider, N., E. Danchik, C. Dyer and N. A. Smith, "Discriminative lexical semantic segmentation with gaps: running the MWE gamut", *Transactions of the Association for Computational Linguistics*, Vol. 2, pp. 193–206, 2014.

16. Graves, A., A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks", *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

17. Lample, G., M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer, "Neural architectures for named entity recognition", *arXiv preprint arXiv:1603.01360*, 2016.

18. Legrand, J. and R. Collobert, "Phrase Representations for Multiword Expressions", *Proceedings of the 12th Workshop on Multiword Expressions*, pp. 67–71, Association for Computational Linguistics, 2016, `http://aclweb.org/anthology/W16-1810`.

19. Huang, Z., W. Xu and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging", *arXiv preprint arXiv:1508.01991*, 2015.

20. Ehren, R., T. Lichte and Y. Samih, "Mumpitz at PARSEME Shared Task 2018: A Bidirectional LSTM for the Identification of Verbal Multiword Expressions", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 261–267, Association for Computational Linguistics, Santa Fe, New Mexico, USA, Aug. 2018, `https://www.aclweb.org/anthology/W18-4929`.

21. Boros, T. and R. Burtica, "GBD-NER at PARSEME Shared Task 2018: Multi-Word Expression Detection Using Bidirectional Long-Short-Term Memory Net-

works and Graph-Based Decoding", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 254–260, Association for Computational Linguistics, Santa Fe, New Mexico, USA, Aug. 2018, https://www.aclweb.org/anthology/W18-4928.

22. Zampieri, N., M. Scholivet, C. Ramisch and B. Favre, "Veyn at PARSEME Shared Task 2018: Recurrent Neural Networks for VMWE Identification", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 290–296, Association for Computational Linguistics, Santa Fe, New Mexico, USA, Aug. 2018, https://www.aclweb.org/anthology/W18-4933.

23. Elman, J. L., "Finding structure in time", *Cognitive Science*, Vol. 14, No. 2, pp. 179 – 211, 1990, http://www.sciencedirect.com/science/article/pii/036402139090002E.

24. Graves, A. and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures", *Neural Networks*, Vol. 18, No. 5, pp. 602 – 610, 2005, http://www.sciencedirect.com/science/article/pii/S0893608005001206, iJCNN 2005.

25. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural Comput.*, Vol. 9, No. 8, pp. 1735–1780, Nov. 1997, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

26. Lafferty, J. D., A. McCallum and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 282–289, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, http://dl.acm.org/citation.cfm?id=645530.655813.

27. Reimers, N. and I. Gurevych, "Optimal Hyperparameters for Deep LSTM-

Networks for Sequence Labeling Tasks", *CoRR*, Vol. abs/1707.06799, 2017, http://arxiv.org/abs/1707.06799.

28. *CoNLL-U*, http://universaldependencies.org/format.html, accessed at June 2019.

29. Eryiğit, G., "ITU Turkish NLP Web Service", *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Association for Computational Linguistics, Gothenburg, Sweden, April 2014.

30. *TDK*, http://www.tdk.gov.tr/, accessed at June 2019.

31. Grave, E., P. Bojanowski, P. Gupta, A. Joulin and T. Mikolov, "Learning Word Vectors for 157 Languages", *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

32. Chollet, F. *et al.*, *Keras*, 2015, https://keras.io, accessed at June 2019.

33. Abadi, M. *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", *CoRR*, Vol. abs/1603.04467, 2016, http://arxiv.org/abs/1603.04467.

34. Reimers, N. and I. Gurevych, "Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging", *arXiv preprint arXiv:1707.09861*, 2017.

35. Sang, E. F. T. K. and J. Veenstra, "Representing Text Chunks", *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pp. 173–179, Association for Computational Linguistics, Stroudsburg, PA, USA, 1999, https://doi.org/10.3115/977035.977059.