IDENTIFICATION OF VERBAL MULTIWORD EXPRESSIONS USING DEEP LEARNING ARCHITECTURES AND REPRESENTATION LEARNING METHODS

by

Berna Erden

B.S., Computer Engineering, Boğaziçi University, 2015

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 and foremost, I would like to express my sincere gratitude to my advisor Prof. Tunga Güngör for his motivation, guidance and immense knowledge. I cannot imagine a better advisor for my masters study.

I am also grateful to my thesis committee members: Assoc. Prof. Arzucan Özgür and Assoc. Prof. Gülşen Cebiroğlu Eryiğit for sparing their time and for their insightful comments on my thesis.

I am indebted to Dr. Suzan Üsküdarlı for her kindness and encouragement during my bachelor and masters studies. I am pleased to thank Prof. Arda Yurdakul for her valuable suggestions at the beginning of my masters study.

My sincere thanks also go to my colleague and my little sister Gözde Berk. We have studied and discovered the world together. I believe that we will go on. Thanks to my lab mates in AI Lab.

I owe my special thanks to my besties: Beste Sağlam, Pelin Yiğit and Hatip Kabak. There are no words to describe your friendships. You have always been there for me even when I returned your calls one week later. I would like to thank Melce Hüsünbeyi for the sleepless nights in the library and the intellectual conversations. I extend my thanks to Melodi Çalışkan and Merve Bozo for the fun time we spent together and the stimulating discussions in which we explored deep learning. I am very thankful to Ebru Aydoğan who informed me of my acceptance to the masters programme and have provided a funny and free place out of my research.

I am grateful to my auntie Mukadder Akıllıoğlu for her interest and wonderful dishes during my bachelor and masters studies. I would like to thank my cousin Derya Başdelioğlu for being my best friend and inspiring me since my childhood.

Last but not least, I would like to thank my mother Ayşe Erden, my father Nusret Erden, for giving me the opportunities that have made me who I am; and my brother Mustafa Erden who has always been there for me when I needed. I would like to thank them all for their unconditional love and continued tolerance. Your love and support are with me in whatever I pursue.

This thesis was supported by Boğaziçi University Research Fund Grant Number 14420.

ABSTRACT

IDENTIFICATION OF VERBAL MULTIWORD EXPRESSIONS USING DEEP LEARNING ARCHITECTURES AND REPRESENTATION LEARNING METHODS

Understanding multiword expressions (MWEs) plays an instrumental role in Natural Language Processing applications such as parsing and machine translation. MWE identification is a task that automatically detects and classifies MWEs in running text. As with the basic characteristics of MWEs, significant challenges exist in MWE identification. Considering the recent attempts of the PARSEME network on verbal multiword expressions (VMWEs), we focus on the identification of VMWEs. We update the PARSEME Turkish train and test corpora 1.0 (2017) as the PARSEME Turkish train and development corpora 1.1 (2018). We construct the PARSEME Turkish test corpus 1.1. In addition, we develop a multilingual VMWE identification system based on bidirectional long short term memory with conditional random fields networks accompanied with the gappy 1-level tagging scheme. To extend our study, we examine the impact of data representation format on the VMWE identification task. We introduce the bigappy-unicrossy tagging scheme to recognize overlaps in sequence labelling tasks. Our results show that data representation format is important to identify discontinuous VMWEs. Moreover, we enhance our neural VMWE identification model with automatically learned embeddings by neural networks to respond to the variability challenge. We compare character-level convolutional neural networks and character-level bidirectional long short-term (BiLSTM) networks. We analyze two different schemes to represent morphological information using BiLSTM networks. Our results demonstrate that character embeddings and morphological embeddings improve performance in general. The choice of representation learning method depends on language.

ÖZET

ÇOK SÖZCÜKLÜ FİİL İFADELERİNİN DERİN ÖĞRENME MİMARİLERİ VE GÖSTERİM ÖĞRENME METOTLARI İLE SAPTANMASI

Çok sözcüklü ifadelerini (ÇSİ) anlamak Doğal Dil İşleme'de ayrıştırma, makine çevirisi gibi uygulamalar için önemlidir. ÇSİ'leri saptama metinde otomatik olarak ÇSİ'leri tanımlama ve sınıflandırma işlemidir. ÇSİ'ler temel karakterlerinden dolayı zorlayıcıdır. PARSEME ağının çok sözcüklü fiil ifadeleri (ÇSFİ) üzerine yaptığı güncel çalışmaları takip ederek, ÇSFİ saptanması üzerine odaklandık. PARSEME Türkçe eğitim ve test derlemi 1.0'ı (2017), PARSEME Türkçe eğitim ve geliştirme derlemi 1.1 (2018) olarak güncelledik. PARSEME Türkçe test derlemi 1.1'i oluşturduk. Ek olarak, çift yönlü uzun kısa-vadeli bellek ve koşullu rastgele alanlar ağını ve gappy 1-level etiketleme şeması ile kullanan çok dilli ÇSFİ'leri saptayan bir sistem geliştirdik. Çalışmamızı ilerletmek için, veri gösterim formatının ÇSFİ saptama işlemi üzerindeki etkisini inceledik. Bigappy-unicrossy etiketleme formatını dizi etiketleme işlemlerinde çakışmaları tanımlamak için geliştirdik. Sonuçlarımız, veri gösterim formatının süreksiz ÇSFİ'leri tanımada önemli olduğunu gösterdi. Ayrıca, değişkenlik problemi için sinir ağları ile otomatik olarak öğrenilmiş gömmeleri kullanarak sistemimizi zenginleştirdik. Karakter seviyesinde evrişimli sinir ağlarını ve karakter seviyesinde çift yönlü uzun kısa-vadeli sinir ağlarını karşılaştırdık. İki farklı ek bilgisi gösterim şeklini çift yönlü uzun kısa-vadeli sinir ağları kullanarak inceledik. Sonuçlarımız karakter ve ek bilgisi gömmelerinin performansı genel olarak geliştirdiğini gösteriyor. Gösterim öğrenme metotu seçimi dile bağlıdır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

h_t	Hidden vector at time t
o_t	Output vector at time t
x_t	Input vector at time t
X	An input sentence
y	A tag sequence
σ	Sigmoid function
$softmax$	Softmax function
$tanh$	Hyperbolic tangent function

LIST OF ACRONYMS/ABBREVIATIONS

AR	Arabic
BiLSTM	Bidirectional Long Short Term Memory
BiLSTM-CRF	Bidirectional Long Short Term Memory with Conditional Random Fields
BG	Bulgarian
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
DE	German
EL	Greek
EN	English
ES	Spain
EU	Basque
FA	Farsi
FR	French
HE	Hebrew
HI	Hindu
HR	Croatian
HU	Hungarian
IAV	Inherently Adpositional Verb
ID	Idiom
IReflV	Inherently Reflexive Verb
IRV	Inherently Reflexive Verb
IT	Italian
LT	Lithuanian
LS.ICV	Inherently Clitic Verb
LSTM	Long Short-Term Memory
LVC	Light Verb Construction
MVC	Multi-verb Construction
MWE	Multiword Expression

NER	Named Entity Recognition
NLP	Natural Language Processing
OTH	Other Verbal MWE
PARSEME	PARSing and Multi-word Expressions
PL	Polish
POS	Part-of-Speech
PT	Portuguese
RNN	Recurrent Neural Network
RO	Romanian
SL	Slovenian
SVM	Support Vector Machine
TR	Turkish
VID	Verbal Idiom
VMWE	Verbal Multiword Expression
VPC	Verb-particle Construction

1. INTRODUCTION

Language is often accepted as one of the important components of human intelligence. In the context of Artificial Intelligence (AI), it is therefore required to properly mimic the understanding of human language to build intelligent systems. With the increasing demand for such intelligent systems, Natural Language Processing (NLP) has received special attention in recent years. Today, NLP researchers study a number of difficult problems to contribute to the development of AI. Multiword expressions (MWEs), described as "a pain in the neck for NLP" [1], are one of the significant challenges in NLP research that needs to be analyzed with a special and comprehensive methodology.

MWEs can be defined as lexical items that are made up of multiple lexemes and have the idiomatic character at lexical, syntactic, semantic, pragmatic, and/or statistical levels [2]. In more detail, the term idiomaticity used to describe MWEs refers to a situation in which the linguistic properties of MWEs cannot be derived from that of component items. For example, the expression of *kick the bucket* means to die. Each component word of the expression does not make an individual contribution to the semantics of the whole. As another example, *traffic light*, it is possible to figure out the semantics of the expression from its parts; however, the expression also produces a new compositional semantics which refers to an object.

Interpreting MWEs plays a key role in NLP applications such as parsing and machine translation. The incorporation of the MWE processing methods into parsing helps resolve the problem of ambiguity and reduce complexity. In the context of machine translation, this may improve the translation of phrases and the word alignment. MWE identification is one of the ways to handle MWEs in NLP tasks. To be more precise, MWE identification is a process that automatically annotates MWEs in running text. The issue of MWE identification can be addressed by following numerous approaches such as rule-based methods, supervised and unsupervised learning algorithms, and sequence labelling methods. It is also worth noting that the basic

characteristics of MWEs present considerable challenges in MWE identification. The challenges can be summarized under four headings: discontinuity, overlaps, ambiguity, and variability [3].

In the last few years, significant attempts have been made to tackle the specific problems in MWE research. A scientific organization of PARSEME (PARSing and Multi-word Expressions) [4] has managed regular events related to the treatment of MWEs. The recent events have been dedicated to verbal multiword expressions (VMWEs) whose syntactic head is a verb such as *to make a mistake*. In edition 1.1 of the PARSEME Shared Task on automatic identification of VMWEs, the organization has released the gold standard corpora including annotated VMWEs on 20 languages in collaboration with NLP researchers. The shared task has provided an opportunity for NLP researchers to develop cross-lingual VMWE identification systems.

On the question of VMWE identification, first, we need to explore the sequence labelling literature. Sequence labelling is a type of pattern recognition task that assigns sequences of categorical labels to sequences of input data [5]. Several downstream NLP tasks, including Part-of-Speech (POS) tagging, Named Entity Recognition (NER) and chunking, can be treated using sequence labelling models. The prominent models employed supervised algorithms like Support Vector Machine (SVM), and generative and discriminative algorithms based on Hidden Markov Models and Conditional Random Fields (CRFs) in the decades of 2000s and 2010s [6–9]. Nevertheless, these approaches heavily relied on the handcrafted features and external resources. With the recent advances in deep learning, Convolutional Neural Network (CNN) and Long Short-term Memory (LSTM) architectures have been shown to be effective for numerous NLP tasks [10–12]. In addition, recent studies based on bidirectional LSTM with CRF (BiLSTM-CRF) networks that combine a statistical approach and a neural network approach have achieved remarkable results on sequence labelling tasks without using the hand-crafted features [13–15].

In addition to the promising deep learning architectures, it is also necessary to review the representation learning methods that have been broadly utilized for sequence

labelling models. In the 2010s, distributed word representations or word embeddings have become an essential part of NLP models to capture semantic and syntactic information [16, 17]. Accordingly, word embeddings have been widely used for neural network based sequence labelling models. Besides word embeddings, many neural sequence labelling models have exploited character-level representations of words in order to extract morphological and syntactic information without using the hand-crafted features [14, 15, 18–20]. To obtain character-level representations, most of the proposed models have employed a CNN or a bidirectional LSTM (BiLSTM) network. Some studies comparing these two approaches used in sequence labelling models have demonstrated that the choice of neural network architecture to learn character-level representations depends on the type of sequence labelling task and the corresponding data set [21, 22]. Additionally, a recent study focusing on morphological analysis has discovered the importance of morphological embeddings computed over morphological tags on the NER task [23].

Turning now to VMWE identification, a huge body of the literature has paid attention to statistical and machine learning approaches [24–27]. Neural network based approaches have gained an interest in recent years [28–31]. However, a more conceptual analysis is required to properly build neural networks for VMWE identification. To this purpose, it is important to consider the major challenges that result from the nature of MWEs. In the context of the discontinuity and overlapping challenge, feasible solutions have been put forward to recognize discontinuous and nested MWEs in supervised algorithms [32], but the overlapping challenge remains limited. Moreover, even though it is beneficial to take into morphological analysis account for the variability challenge [3], previous studies have exclusively focused on the handcrafted features to capture morphological information [24–26,33]. To date, there are only a few works that examine character-level neural models for VMWE identification. [31].

In this Thesis, we aim to develop a multilingual VMWE identification system by responding to the challenges that stem from the nature of MWEs. As a starting point, we update the PARSEME Turkish train and test corpora edition 1.0 as the PARSEME Turkish train and development corpora edition 1.1 based on the PARSEME annotation

guideline 1.1. Also, we create the PARSEME Turkish test corpus edition 1.1. To the main purpose, we propose a VMWE identification system based BiLSTM-CRF networks with regarding the discontinuity challenge. To extend our study, we analyze the importance of data representation format on the VMWE identification task. We introduce a novel tagging scheme called bigappy-unicrossy in order to represent overlaps in sequence labelling tasks. Last, we explore different representation learning methods to rise to the variability challenge in VMWE identification.

2. LITERATURE REVIEW

2.1. Surveys on Multiword Expressions

In a preliminary study related to MWE research [1], MWEs are divided into two main categories: lexicalized phrases and institutionalized phrases. First, lexicalized phrases show the syntactic or semantic idiomaticity to some extent. They contain words which cannot be separately used. Furthermore, lexicalized phrases have three sub-groups: fixed expressions, semi-fixed expressions, and syntactically flexible expressions. Fixed expressions do not undergo morphosyntactic variation and internal modification, namely, they are immutable expressions e.g. *in short*. Semi-fixed expressions allow some degree of lexical variation like the variation in reflexive form, or determiner selection e.g. *car park*. Syntactically flexible MWEs (e.g. *look up*) display a wide range of syntactical behaviour with respect to the word order, internal modification, syntactic variation. Second, institutionalized phrases are the expressions that have a higher frequency than any alternative word combinations e.g. *fresh air*.

A great deal of previous research into MWEs describes five different types of idiomaticity [2]. First, lexical idiomaticity occurs when one or more components of an MWE do not exist in the lexicon of the language such as *ad hoc, ad hominem*. Second, syntactic idiomaticity means that the syntactical properties of an MWE cannot be derived from the syntax of its components. For instance, although the expression of *by and large* is an adverb, it consists of a preposition (*by*) and an adjective (*large*). Third, in terms of semantic idiomaticity, the meaning of an MWE cannot be predicted from its components. Idioms illustrate this point clearly, for example, *spill the beans*. Also, the expressions like *bus driver*, *traffic light* have idiomaticity at the semantic level. Fourth, pragmatic idiomaticity covers MWEs describing a fixed set of situations or used in a particular context such as *good morning*. Last, a particular combination of words tends to occur more frequently compared to alternative phrasings of the same concept, therefore these kinds of MWEs show statistical idiomaticity, including *black and white, salt and pepper*.

A recent survey about MWEs [3] determines the main challenges encountered in MWE identification as follows:

- Discontiguity: MWEs can be continuous or discontinuous. Consider the example *I made some big mistakes*; the expression *make mistake* includes other tokens as *some, big* between its components. Additionally, the tokens of the same expression can appear in a different order because of its flexibility like in another example *This is the biggest mistake that you have made at work*. This problem can vary according to language.
- Overlaps: Overlaps occur in two ways: nesting and sharing. Discontinuous MWEs can potentially overlap with other nested MWEs. To illustrate, in the sentence *I took her decision to move on seriously*, there is a nested MWE *move on* between the components of *take seriously*. Furthermore, such MWEs can share tokens in some cases. A given sentence *Pay close attention who don't clap when you win* includes two MWEs that shares a token such as *pay attention* and *close attention*.
- Ambiguous: It is needed to distinguish the non-compositional usage of a MWE from the literal usage of a word combination in the text to solve ambiguity in MWE identification. For example, *spill the beans* can occur in both usage: *He spilled the beans about the surprise* vs. *I spilled the beans on the floor.*
- Variability: Flexible MWEs show variation in the form. To detect the variants of MWEs can be challenging for MWE identification. For example, *make mistake* can undergo syntactic variations like *make/made/making mistake*. Morphological and syntactic analysis can be useful for this problem.

2.2. PARSEME

The PARSEME initiative has organized two shared tasks with the aim of identifying VMWEs in text for the last two years. The first shared task has been held in 2017, referred as edition 1.0 of the PARSEME Shared Task on automatic identification of VMWEs [34]. The second one has been held in 2018, referred as edition 1.1 of the PARSEME Shared Task on automatic identification of VMWEs [4]. Edition 1.1 of the

PARSEME Shared Task has taken place in two phases. In the first phase, the organization has released a new annotation guideline that describes the types of VMWEs to the language teams. For languages covered by edition 1.0 of the PARSEME corpora, the language teams have updated the existing training and development corpora and developed new test corpora according to the guideline. Also, new language teams have involved in edition 1.1 of the PARSEME corpora. They have developed new training, development and test corpora. In the second phase, the organization has publicly released the PARSEME training, development and test corpora, and several systems have participated in the shared task. The competition has been divided into two tracks as closed track and open track. The systems using only the provided data have competed in the closed track. The systems using external resources as well as the provided data have competed in the open track. Additionally, the organization has introduced new evaluation metrics focusing on MWE-specific challenges to compare the systems in the shared task in a systematic way.

The PARSEME organization assessed the quality of a VMWE identification system with two general metrics: MWE-based score and token-based score. The MWE-based score represents the percentage of VMWE sequences that are fully predicted. However, the token-based score considers also the partial matches, namely the percentage of the predicted tokens in VMWE sequences. Besides the general metrics, the MWE-specific evaluation metrics were defined as follows:

- Continuity metric is calculated for two cases: continuous VMWEs (e.g. *take off* your coat) and discontinuous VMWEs (e.g. *take* your coat *off*).
- Length metric is calculated for two cases: single-token VMWEs (e.g. *aufmachen (to open)*) and multi-token VMWEs (e.g. *macht* es *auf* (to open it up)).
- Novelty metric is calculated for two cases: seen VMWEs and unseen VMWEs. If a VMWE in the test corpus is previously annotated with the same set of lemmas in the training corpus, it is regarded as seen; otherwise, it is regarded as unseen. For instance, the VMWE *made certain decision* in the test corpus has the same set of lemmas with the VMWE *make a good decision* in the training corpus, so it is accepted as seen.

- Variability metric is calculated for only the variants of the seen VMWEs. For example, the occurence of *decision we made* in the test corpus is different from the order of surface-form tokens in *make a decision* in the training corpus, so it is accepted as a variant of a seen VMWE.

2.3. Machine Learning Methods

Liebeskind and HaCohen-Kerner [24] target to automatically identify verb-noun MWEs which consist of a verb and a noun on Hebrew language. They implement nine supervised classification algorithms using three feature sets. Although the previous studies investigate statistical approaches and linguistic approaches, they additionally consider semantical features. The semantic, linguistic and statistical features (206 features in total) are extracted from available resources. In the evaluation, the semantic features outperform the linguistic ones and the statistical ones. Also, they test different combinations of feature sets. The combination of all features results in the highest accuracy of 80.47%. On the other side, the semantic and statistic features result in 80.29% accuracy.

Maldonado *et al.* [25] explore a sequence labelling approach for the identification of VMWEs on 12 languages. They build a prediction model having two sequential blocks: a CRF model and a semantic re-ranking model. In the first step, the CRF model labels VMWE sequences in a sentence with a confidence score. The feature set consists of the surface form, the lemma and the POS tag of a token. If syntactic dependency information is available in language corpus, the surface form, the lemma, and the POS tag of the token's head and the dependency relation tag are added to the feature set to describe the relationships between different morpho-syntactic types. In the second step, the re-ranking model receives the 10 most likely candidate VMWEs in the sentence from the CRF model. Also, the context vectors computed using an external resource is given to the model as input. This model assigns a new score to every candidate VMWE. Then, it labels the one with the highest score as a VMWE. In the experiments, they observe that the features based on syntactic dependency relation improve the performance. Additionally, after applying the re-ranking method to the

outputs of the CRF model, the averaged MWE-based score over all languages increases by 12%.

Boros *et al.* [26] develop a CRF sequence labelling model on 12 languages for the identification of VMWEs. They aim to predict transitions between labels instead of the labels. The prediction model is carried out in two stages. First, the model predicts the head words, namely verb in this case, which is called head labelling. The feature set includes the lemma and POS tag of current and previous tokens. In other words, they use a feature window size of two. Second, the model predicts the remaining parts of VMWEs, which is called tail labelling. The predicted head labels in the first stage are added to the feature set in the second one. However, they prefer to use a feature window size of four in the tail labelling. They obtain a remarkable result for only the Romanian data set.

Waszczuk [27] designs a VMWE identification system based on tree-structured CRF that makes use of two sequential algorithms. The study assumes that CRF is successful at identifying continuous entities, but if the algorithm processes a sequence in the form of a dependency tree, it can also handle discontinuity. Accordingly, the system constructs a dependency tree with two possible tags (MWE and not-MWE). Then, it employs a multiclass logistic regression algorithm to find the optimal hyperpath encoded in the hypergraph for the dependency tree. The feature set includes the surface form, lemma and POS tag information provided in the corpora. The system ranked first with respect to the general ranking in the closed track of edition 1.1 of the PARSEME Shared Task.

2.4. Deep Learning Methods

Huang *et al.* [13] introduce a bidirectional LSTM-CRF network architecture, that uses both statistical and non-linear approaches for sequence labelling tasks. This study proposes that the BiLSTM-CRF network is capable of considering past and future input features via a BiLSTM layer as well as sentence-level tag information via a CRF layer. They conduct a number of experiments using five different architectures which

are LSTM, BiLSTM, CRF, LSTM-CRF, and BiLSTM-CRF on three NLP sequence labelling tasks which are POS tagging, chunking, and NER. They construct the same feature set for each experiment. The feature set consists of pre-trained word embeddings with the dimension of 50, spelling and context features extracted from the data sets. The spelling features generated for each token are the followings: whether a token starts with a capital letter, whether it is in all uppercase, whether it is in all lowercase, whether it contains both letters and digits, whether it contains punctuation, etc.

As a result, the BiLSTM-CRF model slightly outperforms the other models on three data sets. Also, the accuracy of the model is comparable to the state-of-the-art result for each data set. Nevertheless, when they remove the spelling and context features from the feature set, the performances of all models decrease. The removal of some features significantly affects the performance of the CRF model compared to the BiLSTM-CRF model. This demonstrates that the BiLSTM-CRF model is less dependent on the engineered features than the CRF model.

Legrand and Collobert [28] present a neural network model that uses the IOBES tagging scheme in order to perform MWE identification. They test their approach on a French corpus that includes 22600 annotated MWEs. First, a neural network computes fixed-size word and phrase representations over variable length chunks in a sentence. Then, a linear classifier takes as input the representations so that it calculates tag scores for each token in the sentence. Last, the Viterbi algorithm decodes the best tag sequence for the sentence. The model delivers similar performance to the previous studies on this data set.

Klyueva *et al.* [29] follow a deep learning approach based on Recurrent Neural Network (RNN) to identify VMWEs on 18 languages. They use randomly initialized form, lemma and POS tag embeddings with a size of 100 as input. The model is trained using Adam optimizer. Also, the batch size of 64 is chosen for all languages. Even though they succeed in predicting partial VMWE sequences (the token-based score) for a few languages, they fail to predict full VMWE sequences (the MWE-based score) when compared to the other results in the literature.

Stodden *et al.* [30] implement an arc-standard transition algorithm using neural network architecture for the identification of VMWEs in edition 1.1 of the PARSEME Shared Task. The system takes as input the surface form, lemma, POS tag, and length of each token. First, the system employs a transition-based dependency parser to find relationships between words. Then, it determines the best transition sequence for a sentence using a CNN based classifier. They prefer to use a kernel SVM layer after the CNN layer instead of a softmax layer. Whereas the system achieves better results for eight of 19 languages than the other systems in the shared task, it delivers 0% accuracy on the Turkish data set. After removing some rows that include additional information in the data set, the accuracy of the system increases from 0% to 39.34%. This contradictory result may be due to that the system is sensitive to the noise in the data set.

In edition 1.1 of the PARSEME Shared Task, Taslimipoor and Rohanian [31] design a neural network model that combines CNN and BiLSTM-CRF architectures in order to identify VMWEs. Also, they use IOB tagging scheme. The first part of model computes bigram and trigram character embeddings on a CNN layer. Then, it concatenates the character embeddings to pre-trained word embeddings so that it constructs the input representation vector. Moreover, the model exploits word shape features and POS tag information for each token as input. The word shape features include the following binary information: whether a word starts with a capital letter, it starts with #, it starts with @, it is an URL, and it contains a digit or not. The second part of the model predicts the sequence of tags for the sentences using BiLSTM-CRF classifier. Additionally, when converting the predicted tags in IOB tagging format to the annotation format, they apply a filtering technique for the predicted ones. If the tag I does not have any predecessor B tag, they mark these cases as non-VMWE. The proposed system takes first place in terms of general ranking in the open and closed track of shared task.

2.5. Tagging Schemes

Schneider *et al.* [32] offer a new tagging scheme called gappy 1-level to encode discontinuous and nested MWEs in the text. The gappy 1-level tagging scheme is a variant of the IOB tagging scheme. It augments the IOB tagging scheme including *B, I, O* tags with additional tags *b, i, o* to cover gappy tokens and/or a nested MWE between the components of a MWE sequence. They compare the gappy 1-level tagging scheme with the IOB tagging scheme on an English corpus containing 3483 annotated MWEs. They choose a structured perceptron learning algorithm as a model. The base model using IOB tagging has an F1-score of 63.20%. The other model using gappy 1-level has an F1-score of 63.50%. As a result, the models perform similarly. The small difference between the two models could be associated that only one-quarter of all MWEs constitutes the discontinuous cases in the corpus.

Zampieri *et al.* [35] examine two different tagging schemes using a deep learning approach based on an RNN for VMWE identification on 19 languages. The first tagging scheme contains the same tags as the original IOB tagging scheme. It is also enlarged with a new tag *g* in order to represent gappy tokens within a VMWE sequence. However, this tagging scheme does not make use of the VMWE category labels provided in the corpora, thereby only labelling the position of VMWE sequences in a sentence. In contrast, the second tagging scheme also concatenates the VMWE category labels to the tags. For the first tagging scheme, they add a heuristics layer to the RNN to determine the VMWE categories. For the second one, the model predicts the tags with the category labels. Although the first tagging scheme performs worse than the second one based on MWE-based score, it delivers higher performance than the second tagging scheme based on token-based score.

2.6. Representation Learning Methods

Lample *et al.* [15] present a neural network architecture that incorporates character-level representations using a BiLSTM network into a sequence tagger based on a BiLSTM-CRF network for NER task. The character-level BiLSTM network allows

the tagger to learn the prefix of a word via its forward layer as well as the suffix of the word via its backward layer. To evaluate the effectiveness of the character model, they compare the proposed architecture with the base BiLSTM-CRF architecture not using character-level representations on four languages. In addition to this, they analyse the impact of dropout on the output of the embedding layer. The findings of this study indicate that the overall F1 score on four languages increases by 0.74% and 1.79% when using the character-level representations and dropout, respectively.

Ma and End [14] propose an end-to-end model known as BiLSTM-CNN-CRF without depending on the hand-crafted features and task-specific resources except pre-trained embeddings for sequence labelling tasks. They test the methodology on two sequence labelling tasks which are POS tagging and NER . The model comprises different architectures such as CNN and BiLSTM-CRF. First, the CNN layer takes as input the characters of words and generates character-level embeddings. They choose a window size of 3 and a filter size of 30, thus the CNN learns character trigrams, and produces the character vectors with the dimension of 30. Second, the concatenation of character-level embeddings with word embeddings is given to the BiLSTM-CRF layer. In this model, dropout with the rate of 0.5 is applied to the input of CNN layer and BiLSTM layer as well as the output of BiLSTM layer. In the experiments on the effect of dropout, they observe that applying dropout on such layers improves the performance of the model on the development set. They state that the proposed model slightly improves the accuracy of state-of-the-art models by 0.05% and 0.01% for POS tagging and NER tasks, respectively.

Gungor *et al.* [23] seek to address the problem of NER on morphologically rich languages. They investigate four different configurations to construct morphological embeddings over morphological tags obtained by morphological analysis. The first configuration called *WITH ROOT* learns the morphological representations over the lemma and the morphological tags of a word. The second one called *WITHOUT ROOT* does not make use of the lemma of the word, thus just relying on the morphological tags. The third one is developed for only Turkish. The fourth one called *CHAR* treats the concatenation of lemma and morphological tags character by character. A

BiLSTM network is trained for all types of embeddings. Then, the combination of these embeddings with the word and character embeddings is fed to a BiLSTM-CRF network to predict named entity tags. The experimental setup suggests that the configurations *WITH ROOT* and *CHAR* yield better performance than the other two configurations. Additionally, the usage of morphological embeddings with the word and character embeddings delivers state-of-the-art results in comparison to the other studies in the literature.

3. DATA SET

The availability of high-quality data sets is a vital factor in the success of deep learning approaches for NLP. It is important to evaluate the deep learning approaches on such data sets constructed with gold standard techniques, thus achieving high standards. Furthermore, the generalization of these approaches to several languages helps push NLP research forward, but this condition requires multilingual data sets containing diverse examples.

A survey on MWE language resources conducted in 2016 demonstrated that the MWE languages resources encompass MWE lists, MWE lexicons, treebanks and corpora with annotated MWEs, yet each resource comes in certain kinds of MWEs for one language or a few languages [36]. In 2017, a major step has been taken by the PARSEME network to meet the need for a common data set including MWEs. Edition 1.0 of the PARSEME Shared Task has provided the annotated corpora prioritizing VMWEs for 18 languages [34]. For edition 1.1 of the PARSEME Shared Task in 2018, the corpora have been revised with the enhanced annotation methodology and enlarged with additional annotated data [4].

3.1. The PARSEME Corpora Edition 1.1

This section explains the details of the annotation methodology presented in the PARSEME Shared Task edition 1.1 and the PARSEME corpora edition 1.1 [4]. According to the annotation guideline edition 1.1, a VMWE comprises a head word which is a verb and at least one syntactically related word. VMWEs are divided into four main categories as follows:

- Universal categories exist in all languages participated in the shared task. Light Verb Constructions (LVCs) and Verbal Idioms (VIDs) belong to universal categories. LVCs formed by a verb and a noun has two subtypes: LVC.full and LVC.cause. LVC.full refers to LVCs in which the verb only affects morphological

features of the whole expression e.g. *give a lecture*. LVC.cause refers to LVCs in which the subject of the verb indicates the cause of the noun e.g. *give a headache*. VIDs are semantically noncompositional expressions e.g. *to go bananas*.

- Quasi-universal categories exist in some languages. Inherently Reflexive Verbs (IRVs), Verb-particle Constructions (VPCs), and Multi-verb Constructions (MVCs) constitute the quasi-universal categories. IRVs (e.g. *wet oneself*) are formed by a reflexive verb and a clitic pronoun. IRVs are annotated when the reflexive verb always occurs with the clitic or the clitic affects the meaning or subcategorization frame of the reflexive verb. VPCs formed by a verb and a particle has two subtypes: VPC.full and VPC.semi. VPC.full refers to fully non-compositional VPCs e.g. *do in*. VPC.semi refers to semi-non-compositional VPCs e.g. *wake up*. MVCs are formed by two adjacent verbs that act as a single predicate e.g. *let go*.
- Language-specific categories include VMWE types that are specific to one language. Inherently Clitic Verbs (LS.ICVs) constitute language-specific categories. LS.ICVs are observed in only Italian. LS.ICVs are formed by a verb and at least one non-reflexive clitic e.g. *IT: prenderle (EN: get beaten up)*.
- Optional experimental category is introduced after the annotation process. Inherently adpositional verbs (IAVs) constitute this category. IAVs are formed by a verb or VMWE and a preposition or a postposition e.g. *put up with*.

The PARSEME corpora edition 1.1 are constructed for 20 languages as follows: Arabic (AR), Bulgarian (BG), German (DE), Greek (EL), English (EN), Spanish (ES), Basque (EU), Farsi (FA), French (FR), Hebrew (HE), Hindu (HI), Croatian (HR), Hungarian (HU), Italian (IT), Lithuanian (LT), Polish (PL), Portuguese (PT), Romanian (RO), Slovenian (SL), and Turkish (TR). The corpora are publicly available for 19 languages [37]. Only the Arabic corpus is not publicly available because it does not have an open licence. Therefore, we consider 19 languages in this Thesis. Each language corpus is split into three sets: training set, development set and test set. But the English corpus does not have the test set.

The participating languages are grouped into four language families as followings:

- Germanic languages: DE, EN
- Romance languages: ES, FR, IT, PT, RO
- Balto-Slavic languages: BG, HR, LT, PL, SL
- Other languages: EL, EU, FA, HE, HI, HU, TR

The corpora are provided in cupt format [4] which is an extension of CoNLL-U format [38]. The cupt format includes three types of lines as the CoNLL-U format. First, each token in a sentence appears in a line and is represented with 11 fields. The first 10 fields are the same as the CoNLL-U format. The fields contain the word index (ID), the form (FORM), the lemma (LEMMA), the universal POS tag (UPOS), the language-specific POS tag (XPOS), the morphological features (FEATS), the syntactic dependencies (HEAD, DEPREL, DEPS), and any other annotation (MISC) of the token, respectively. The eleventh field (PARSEME:MWE) introduces the VMWE annotation. Second, comment lines start with #. Third, blank lines represent sentence boundaries. If an underscore _ occurs in a field, it means that the corresponding annotation is underspecified. If a star * occurs in a field, it means that the corresponding annotation is empty. Figure 3.1 illustrates an example sentence from the English training corpus in the cupt format.

```
# global.columns = ID FORM LEMMA UPOS XPOS FEATS HEAD DEPREL DEPS MISC PARSEME:MWE
# source_sent_id = . . 2289
# text = When he joined the group, they were listening to her.
1       When    when    SCONJ   _       _       3       mark    _       _       *
2       he      he      PRON    PERS-P3SG-NOM   _       3       nsubj   _       _       *
3       joined  join    VERB    PAST    _       9       advcl   _       _       *
4       the     the     DET     DEF     _       5       det     _       _       *
5       group   group   NOUN    SG-NOM  _       3       obj     _       SpaceAfter=No   *
6       ,       ,       PUNCT   Comma   _       3       punct   _       _       *
7       they    they    PRON    PERS-PL-NOM     _       9       nsubj   _       _       *
8       were    be      AUX     PAST    _       9       aux     _       _       *
9       listening       listen  VERB    ING     _       0       root    _       _       *
10      to      to      ADP     _       _       11      case    _       _       *
11      her     she     PRON    PERS-P3SG-ACC   _       9       obl     _       SpaceAfter=No   *
12      .       .       PUNCT   Period  _       9       punct   _       _       *
```

Figure 3.1. An example sentence from the English training corpus in the cupt format.

3.2. Turkish Verbal Multiword Expressions Corpus

In 2018, I participated in the annotation phase of the PARSEME Shared Task edition 1.1 [4] as a member of the Turkish language team. The Turkish language team consisted of a language leader and two annotators. The team members were native speakers of Turkish. I was one of the annotators. The annotation process proceeded in two stages. First, we updated the Turkish training and test corpora edition 1.0 [34] according to the annotation guideline edition 1.1. Second, we created a new Turkish test corpus for edition 1.1. The process took four months. The updated version of the PARSEME Turkish training and test corpora edition 1.0 was renamed to the PARSEME Turkish training and development corpora edition 1.1. The new Turkish test corpus was named the PARSEME Turkish test corpus edition 1.1. This section explains the updates in the PARSEME Turkish training and development corpora edition 1.1, and the construction and annotation of the PARSEME Turkish test corpus edition 1.1 [4, 39].

3.2.1. Categories of Turkish Verbal Multiword Expressions

To gain a better understanding of Turkish VMWEs, we examined a key study categorising Turkish MWEs in the literature [40]. In the light of this study, we focused on the following definitions to classify Turkish VMWEs:

- Verbal Compound MWEs: A verbal compound MWE is composed of a verb and a noun *TR: karar vermek (EN: to decide)*. The component words do not undergo a significant change in their meanings.
- Light Verb Construction MWEs: Light verbs in Turkish are composed of six auxiliary verbs: *TR: olmak (EN: to be)*, *TR: etmek (EN: to do)*, *TR: yapmak (EN: to make)*, *TR: kılmak (EN: to render)*, *TR: eylemek (EN: to make)*, and *TR: buyurmak (EN: to order)*. The light verbs with a proceeding nominal constitute light verb construction MWEs e.g. *TR: ifade etmek (EN: to state)*.
- Idiomatic Expression MWEs: Idiomatic expressions occur when the semantics of expression is different from the semantics of its components e.g. *TR: piyasaya*

sürmek (EN: to launch to the market).

- Formulaic Expression MWEs: Formulaic expressions are MWEs that have the meaning of well-wishing or gratitude e.g. *TR: Hoşça kal (EN: Good bye).*
- Proverb MWEs: Idiomatic and fixed sentences constitute proverb MWEs e.g. *TR: Damlaya damlaya göl olur (EN: Many a little makes a mickle).*

According to the PARSEME annotation guideline edition 1.0 [34], VMWEs were divided into five categories, including LVCs, Idioms (IDs), Inherently Reflexive Verbs (IReflVs), Verb-particle Constructions (VPCs), and Other Verbal MWEs (OTH). The previous edition of the Turkish corpus had three categories which are ID, LVC and OTH.

Considering the key study about MWEs [40] and the annotation guideline edition 1.0 and 1.1, we decided to use three categories to annotate Turkish VMWEs: LVC.full, VID and MVC. It is worth mentioning that the LVC.cause category does not exist in Turkish.

3.2.2. Annotation Phase of Turkish Verbal Multiword Expressions Corpus

In the first stage, we updated the PARSEME Turkish training and test corpora edition 1.0 based on the changes in the annotation guideline edition 1.1. There were several differences between the annotation guideline edition 1.0 and the annotation guideline edition 1.1. The ID and IReflV categories were renamed to VID and IRV, respectively. The OTH category was removed, the new VMWE categories which are MVC and IAV were introduced. Additionally, the LVC category was split into two subcategories which are LVC.full and LVC.cause. Similarly, the VPC category was split into two subcategories which are VPC.full and VPC.semi. The previous edition of the Turkish corpus had three categories which are ID, LVC and OTH. Therefore, we changed the categories of VMWEs annotated as ID and LVC to VID and LVC.full, respectively. Then, we reviewed the VMWEs annotated as OTH. To determine the new categories of VMWEs annotated as OTH, we followed the annotation decision tree constructed by the PARSEME network [4] that is shown in Figure 3.2. However,

we encountered some issues while categorizing verbal compound constructions. After we discussed the issues that we came across, we drew up three specifications for the categorization of Turkish VMWEs. During the annotation process, we made use of the Turkish Language Institution dictionary [41] to check the meaning of a word.

Figure 3.2. The annotation decision tree constructed by the PARSEME network.

In the second stage, we constructed the new Turkish test corpus for edition 1.1 of the PARSEME Shared Task. We gathered the data from newspaper articles in several genres such as politics, world, life, and art and columns. Then, we used ITU Turkish NLP Web Service [42] to parse and tokenize the data. ITU Turkish NLP Web Service returned the output in the CoNLL-U format with the POS tags, the morphological features, and the syntactic dependency tags for each token. The universal dependency relation tags which occur in the DEPREL field also contained the MWE annotations,

therefore we used this information to identify candidate VMWEs in the test corpus. After we annotated the VMWEs in the test corpus, we converted it to the cupt format by adding the VMWE annotations.

The annotation process starts with identifying a candidate in the text. If the candidate is formed by a verb and a noun, we initially use the LVC-specific tests in the decision tree to check whether it is a VMWE or not. If the candidate VMWE passes all the LVC-specific tests, it is categorized as an LVC. The categorization of an example expression *TR: karar vermek (EN: to decide)* is shown as follows:

- LVC.0 - [N-ABS]: The noun should be abstract. *TR: karar (EN: decision)* is an abstract noun, so the example passes this test.
- LVC.1 - [N-PRED]: The noun should be predicative. In other words, it should have a semantic argument. The example *TR: karar vermek (EN: to decide)* refers to an event. There is one semantic argument *TR: karar veren (EN: the decider)*. So, the example passes this test.
- LVC.2 - [N-SUBJ-N-ARG]: The subject of the verb should be a semantic argument of the noun. In a given sentence *TR: Ali ders çalışmaya karar verdi (EN: Ali decided to study)*, *Ali* is the subject of the verb *TR: vermek (EN: to give)* and linked to the noun *TR: karar (EN: decision)*. So, the example passes this test.
- LVC.3 - [V-LIGHT]: The verb should be semantically light. That is to say, the verb changes only morphological features of the noun, it does not add any meaning to the noun. The verb *TR: vermek (EN: to give)* adds no meaning to the noun *TR: karar (EN: decision)*, it just makes the event happen. So, the example passes this test.
- LVC.4 - [V-REDUC]: When a phrase is constructed using the noun and the subject without the verb, the phrase should refer to the same event or state. The constructed phrase *TR: Ali'nin kararı (EN: Ali's decision)* refers to the same event in the example *TR: karar vermek (EN: to decide)*. So, the example passes the last test, it is categorized as an LVC.

If the candidate formed by a verb and a noun fails at the LVC-specific tests or there is a candidate VMWE formed by a verb and at least one other word, we apply the VID-specific tests in the decision tree for such candidate. If the candidate passes one of the VID-specific tests, it is categorized as a VID. Otherwise, we accept that it is not a VMWE. This case is shown by the categorization of an example expression *TR: ifade vermek (EN: to testify)* as follows:

- VID.1 - [CRAN]: If the candidate includes a cranberry word, it is marked as a VID. The components of the example *TR: ifade vermek (EN: to testify)* are standalone Turkish words. So, the example does not pass this test.
- VID.2 - [LEX]: VIDs can be lexically inflexible. Considering the example, when the component *TR: ifade (EN: statement)* is replaced with its synonym *TR: açıklama (EN: explanation)*, *TR: açıklama vermek (EN: to give an explanation)* does not exist in Turkish. This shows that the example is inflexible. So, it is categorized as a VID.
- VID.3 - [MORPH]: VIDs can be morphologically inflexible. If the component *TR: ifade (EN: statement)* is inflected in the plural form, *TR: ifadeler vermek (EN: to give statements)* breaks general grammar rules of Turkish. This means that the candidate is inflexible. So, it is categorized as a VID.
- VID.4 - [MORPHSYNT]: VIDs can be morpho-syntactically inflexible. In the example sentence *TR: Karakolda ifade verdim (EN: I testified at the police station)*, if the component *TR: ifade (EN: statement)* takes a possessive suffix, the new sentence *TR: Karakolda ifadeni verdim (EN: I gave your statement at the police station)* is not grammatically correct in Turkish. This shows that the example is inflexible. So, it is categorized as a VID.
- VID.5 - [SYNT]: VIDs can be syntactically inflexible. Considering the previous example sentence, it is possible to change the order of the components like *TR: Karakolda verdiğim ifade tekrar kontrol edildi (EN: My statement that I gave at the police station was checked again)*. This means that the example is syntactically flexible. So, it fails at this test.

As a result, the example passes the VID.2 test, therefore it is categorized as a VID. There is no need to apply the remaining VID-specific tests for this example.

If there is a candidate formed by a sequence of two adjacent verbs, we apply the MVC-specific tests for it. For example, in the sentence *TR: İki karar arasında gidip geliyorum (EN: I go and come between two decisions)*, there is an example expression *TR: gidip gelmek (EN: to go and come).* The details of the MVC-specific tests for this example are given below:

- The verbs usually have the same subject. *TR: gitmek (EN: to go)* and *TR: gelmek (EN: to come)* have the same subject which is "I" here.
- The verbs usually belong to the connected actions or the same event. *TR: gitmek (EN: to go)* and *TR: gelmek (EN: to come)* refer to the same event.
- The verbs act together as a single predicate. The sequence *TR: gidip gelmek (EN: to go and come)* functions together as the verb of the sentence.

In addition to the tests provided in the annotation guideline edition 1.1, there are three specifications for the annotation of Turkish VMWEs. First, as mentioned before, the light verbs cover six auxiliary verbs in Turkish. However, LVCs are not limited to be formed by only these verbs. There are also some supportive verbs that behave like the semantically light verbs in some cases. For instance, the combination of the noun *TR: oy (EN: vote)* and the verb *TR: vermek (EN: to give)* forms an LVC *TR: oy vermek (EN: to vote).* The verb does not add any meaning to the noun, it just performs an activity of voting. Second, if a noun and the verb *TR: vermek (EN:to give)* constitute an LVC, the same noun and the verb *TR: almak (EN: to take)* may constitute a VID. For instance, the noun *TR: oy (EN: vote)* and the verb *TR: almak (EN: to take)* form the VID *TR: oy almak (EN: to receive vote).* This decision stems from the fact that the verb adds a new meaning to the noun. Also, the noun cannot be replaced with its synonym *TR: rey (EN:vote)* in this case, the expression is lexically inflexible as well. Third, the constructions including the verbs that are *TR: sağlamak (EN: to provide)*, *TR: duymak (EN: to get)*, *TR: ulaşmak (EN: to reach)*,

TR: karşılamak (EN: to meet), *TR: kullanmak (EN: to use)*, *TR: uygulamak (TR: to perform)* are usually not VMWEs.

3.2.3. Turkish Verbal Multiword Expressions Corpus Results

In this section, we present the results of the PARSEME Turkish corpus edition 1.1. The revised version of the PARSEME Turkish training corpus edition 1.0 was named the PARSEME Turkish training corpus edition 1.1. The revised version of the PARSEME Turkish test corpus edition 1.0 was named the PARSEME Turkish development corpus edition 1.1. The new Turkish test corpus was released under the name the PARSEME Turkish test corpus edition 1.1. Additionally, the LVC and ID categories in the previous corpus were converted to the LVC.full and VID categories, respectively. Table 3.1 shows the statistics about the PARSEME training corpus 1.0 and the PARSEME training corpus 1.1. Table 3.2 shows the statistics about the PARSEME test corpus 1.0 and the PARSEME development corpus 1.1. Table 3.3 shows the statistics about the PARSEME test corpus 1.1. The number of sentences, the number of annotations per category, and the total number of VMWE annotations for each subset are presented.

As seen in Table 3.1, the PARSEME Turkish training corpus 1.1 contains 6120 VMWEs in total. As seen in Table 3.2, the PARSEME Turkish development corpus 1.1 contains 500 VMWEs in total. Out of the total of 687 OTHs in the previous training and development sets, 354 were categorized as LVC.fulls, 282 were categorized as VIDs, 1 was categorized as MVC and 50 were categorized as non-MWEs.

As seen in Table 3.3, the new corpus consists of 577 sentences and 509 annotated VMWEs. The test corpus contains 273 LVC.fulls, 235 VIDs and 1 MVC. The MVC category is the least annotated category among Turkish VMWEs.

Table 3.1. Statistics about the Turkish training corpus.

	Sentences	LVC.full	VID	OTH	MVC	VMWE
PARSEME 1.0	16715	2624	2911	634	0	6169
PARSEME 1.1	16715	2950	3169	0	1	6120

Table 3.2. Statistics about the Turkish development corpus.

	Sentences	LVC.full	VID	OTH	MVC	VMWE
PARSEME 1.0	1320	199	249	53	0	501
PARSEME 1.1	1320	227	273	0	0	500

Table 3.3. Statistics about the Turkish test corpus.

	Sentences	LVC.full	VID	OTH	MVC	VMWE
PARSEME 1.1	577	273	235	0	1	509

4. METHODOLOGY

In this Thesis, we model VMWE identification as a sequence labelling task. Following the state-of-the-art models proposed for sequence labelling tasks, we develop a VMWE identification system based on a BiLSTM-CRF network. We accompany the BiLSTM-CRF network with the gappy 1-level tagging scheme to recognize discontinuous VMWEs. In addition to this, we introduce a new tagging scheme called bigappy-unicrossy to recognize nested and crossing VMWEs as well as discontinuous VMWEs. Last, we enhance the proposed neural network architecture with different feature representations to capture morphological information, thereby handling the variability challenge.

4.1. Deep Learning Architectures

In this section, we provide a brief description of LSTM, BiLSTM and BiLSTM-CRF networks and the dropout technique.

4.1.1. Long Short-term Memory

RNNs are a type of neural networks that was designed to process sequential data. They have been broadly utilized in various tasks ranging from sequence labelling to sequence prediction. A standard RNN consists of three layers: an input layer, a hidden layer and an output layer. For a given vector x_t at time t, the RNN computes the hidden state vector h_t and the output vector o_t as follows:

$$h_t = tanh(Ux_t + Wh_{t-1} + b_h)$$
$$o_t = softmax(Vh_t + b_o)$$

where U, W, and V are the weight matrices, b_h and b_o are the bias vectors.

In theory, RNNs take as input the current input and the previous hidden state vectors, thereby connecting the previous information to the current information. In practice, RNNs, however, fail at learning long time dependencies, which results in the vanishing gradient problem [43].

LSTM networks were introduced to overcome the vanishing gradient problem of RNNs [44]. LSTM networks are an extension of RNNs, however they differ from RNNs in that they make use of special units to compute the hidden state and the output vectors. An LSTM architecture consists of a memory cell, an input gate, a forget gate and an output gate. The implementation of the LSTM architecture is formulated as follows:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + R_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + R_f h_{t-1} + b_f) \\
\tilde{c_t} &= tanh(W_c x_t + R_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \\
o_t &= \sigma(W_o x_t + R_o h_{t-1} + b_o) \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

where σ is the sigmoid function and $\odot$ is the element-wise product. W_i, W_f, W_c, W_o denote the input weight matrices. R_i, R_f, R_c, R_o denote the recurrent weight matrices. b_i, b_f, b_c, b_o denote the bias vectors. The input gate vector i_t, the forget gate vector f_t, and the output gate vector o_t are computed using the current input vector x_t and the previous hidden state vector h_t. The current cell state vector c_t is computed using the previous cell state vector c_{t-1}, the candidate cell vector $\tilde{c_t}$, the input gate i_t and the forget gate f_t. The hidden state vector h_t is computed based on the output gate o_t and the current cell state c_t.

4.1.2. Bidirectional Long Short-term Memory

BiLSTM networks is a type of neural networks that combines an LSTM network processing sequence in forward order with another LSTM network processing the sequence in backward order [45]. In this way, they become capable of capturing past and future information. A BiLSTM network computes the forward hidden state vector $\vec{h_t}$ and the backward hidden state vector $\overleftarrow{h_t}$. The concatenation of the two vectors forms the output of the BiLSTM network, $h_t = [\vec{h_t}; \overleftarrow{h_t}]$.

4.1.3. Bidirectional Long Short-term Memory with Conditional Random Fields

BiLSTM-CRF networks is a hybrid architecture that is composed of a BiLSTM component and a CRF component [13]. They can learn past and future information in a sentence via the BiLSTM component as well as the dependencies between the consecutive tags in the sentence via the CRF component [45, 46].

Formally, $X = (x_1, x_2, ..., x_n)$ represents an input sentence consisting of n words. Each x_i is a d-dimensional vector that represents the i^{th} word (e.g. embedding). $y = (y_1, y_2, ..., y_n)$ represents a sequence of predictions for the input sentence X. Each y_i is a k-dimensional vector where k is the number of unique tags. The BiLSTM layer of a BiLSTM-CRF network takes as input a sentence X. Then, it computes the output vector $h_i = [\vec{h_i}; \overleftarrow{h_i}]$ for each word. The output vectors are fed to the fully connected layer of the network. The fully connected layer computes the tag score matrix P of size $n \times k$. $P_{i,j}$ denotes the score of the j^{th} tag for the i^{th} word. The CRF layer of the network computes the most likely tag sequence y^* for the input sentence X as follows:

$$s(X, y) = \sum_i A_{y_i, y_{i+1}} + \sum_i P_{i, y_i}$$

$$y^* = \arg\max_{\tilde{y}} s(X, y')$$

where $s(X, y)$ is the score of a sentence X along with a tag sequence y, and A is a transition score matrix of size $k \times k$. $A_{i,j}$ is the transition score from the tag i to the tag j.

In this Thesis, we propose several VMWE identification models based on the described architecture. We change only the input representation of a word x_i and IOB encoding format for each model. More details on this will be discussed in the followings sections.

4.1.4. Dropout

Dropout is a regularization technique that is commonly used to reduce overfitting for neural networks. The dropout technique randomly deactivates some neurons in a neural network with a probability p during training [47].

4.2. Deep-BGT at PARSEME Shared Task 2018

In this section, we present the Deep-BGT system developed for edition 1.1 of the PARSEME Shared Task on automatic identification of VMWEs [48]. The proposed system employs a BiLSTM-CRF neural network using the gappy 1-level tagging scheme. To address the discontinuity problem in MWE identification, we exploit the gappy 1-level tagging scheme instead of the IOB tagging scheme which is commonly used for sequence labelling tasks.

The tag set of the gappy 1-level tagging scheme is composed of six types of tags which are *B, I, O, b, i, o*. The term chunk used here refers to a MWE sequence. The term gappy chunk refers to a discontinuous MWE sequence. The term gap refers to a token that is in between a gappy chunk but does not belong to that chunk. The uppercase tags are similar to the tags in the IOB tagging schemes. B labels the first token of a chunk. I labels the other tokens belonging to the chunk. O labels tokens outside of the chunk. The lowercase tags are used to represent continuous nested chunks within gappy chunks. b labels the first token of a continuous nested chunk. i labels

the other tokens belonging to the continuous nested chunk. *o* labels tokens outside of any chunk within a gappy chunk. This tagging scheme ignores discontinuous nested chunks, so it labels these cases as *o*.

To properly use the gappy 1-level tagging scheme for VMWE identification, we modify some rules of this tagging scheme. We append the VMWE category tags to the tags *B, I, b, i*. For practical reasons, we consider only discontinuous cases made up of two tokens. If a discontinuous VMWE consists of more than two tokens, we label the tokens between the first token and the last token of the discontinuous VMWE as *o*. Also, we label nested VMWEs within such discontinuous VMWEs as *o*. For the VMWEs that share tokens, we remove the tags of the second VMWE that follows the first VMWE. Moreover, we label single-token VMWEs despite of that the gappy 1-level tagging scheme considers only multi-token VMWEs. Figure 4.1 shows an example sentence labelled with the gappy 1-level tagging scheme. The example sentence contains two VMWEs: *took seriously* as a VID and *move on* as a VPC.full.

We develop a VMWE identification model based on the BiLSTM-CRF architecture that we described in Subsection 4.1.3. In this study, we use the pre-trained word embeddings released by fastText. The fastText embeddings were trained on Common Crawl and Wikipedia. The vocabulary size of the embeddings is 2M words and the embedding vector dimension is 300. Additionally, we use POS and DEPREL tags provided in the data set to extract dependencies at sentence level. The model takes as input the concatenation of the pre-trained word embeddings, the one-hot encoding representation of the POS tags, and the one-hot encoding representation of the DEPREL tags. Figure 4.1 shows the Deep-BGT system developed for the PARSEME Shared Task Edition 1.1.

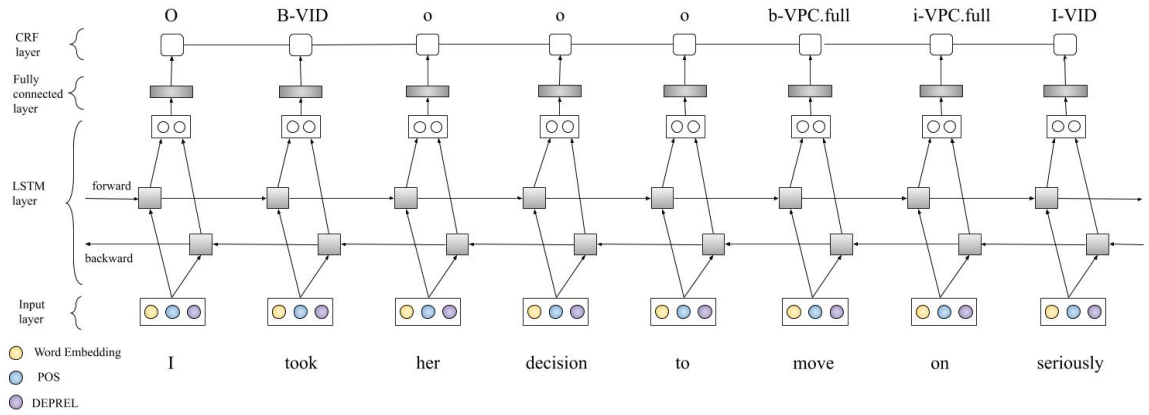

Figure 4.1. Architecture of the Deep-BGT system for VMWE identification.

4.3. Representing Overlaps in Sequence Labeling Tasks with a Novel Tagging Scheme: bigappy-unicrossy

In this section, we introduce a novel tagging scheme called bigappy-unicrossy in order to resolve the challenge of overlaps in MWE identification [49]. To evaluate the bigappy-unicrossy tagging scheme, we compare it with the IOB2 tagging scheme and the gappy 1-level tagging scheme using the BiLSTM-CRF model that we proposed in Section 4.2 on the VMWE identification task. The bigappy-unicrossy tagging scheme can be applied to various sequence labelling tasks in which overlapping sequences are frequently observed.

We approach the VMWE identification task as a sequence labelling problem using IOB encoding. Over time, several data representation formats have been developed for sequence labelling tasks. However, the popular formats which are the IOB1 and IOB2 tagging schemes [50, 51] are not suitable for the VMWE identification task with regard to the MWE-specific challenges which are discontinuity and overlaps. Although the gappy 1-level tagging scheme proposes a solution to represent discontinuous MWEs and continuous nested MWEs, the other types of overlaps such as crossing and shared tokens remains to be addressed. To illustrate this, we examine the IOB1 tagging scheme, the IOB2 tagging scheme and the gappy 1-level tagging scheme in detail.

The IOB1 tagging scheme was proposed to treat chunking as a tagging problem [50]. The tag set of this tagging scheme consists of three uppercase letters: *I, O, B*. *I* stands for a token inside a chunk. *O* stands for tokens outside of any chunk. However, *B* stands for only the first token of a chunk that immediately follows another chunk of the same type. In this way, *B* is only used to separate two neighbouring chunks of the same type.

The IOB2 tagging scheme was introduced to treat initial tokens of chunks independently [51, 52]. Although the tag set of the IOB2 tagging scheme is exactly the same as that of the IOB1 tagging scheme, the IOB2 tagging scheme assigns a different meaning to the tags. *B* stands for the first token of a chunk. *I* stands for the other tokens belonging the chunk. *O* stands for tokens outside of any chunk. In this way, every chunk starts with the *B* tag. If a chunk contains more than one token, *I* is used for the remaining tokens of the chunk. Hence, a single-token chunk receives the *B* tag.

In addition to these schemes, the gappy 1-level tagging scheme was proposed to treat discontinuous MWEs and continuous nested MWEs in the text. To this purpose, it enlarges the tag set of the IOB tagging scheme with the new tags *b, i, o* as we mentioned in Section 4.2.

In the context of MWEs, the IOB1 and IOB2 tagging schemes cannot represent discontinuos MWEs in the text. However, the IOB2 tagging schemes can label single-token MWEs. As regarding the gappy 1-level tagging scheme, it accepts continuous nested MWEs, but it ignores discontinuous nested MWEs. Also, it is not specifically designed to recognize the other types of overlaps such as crossing and shared tokens. In order to properly address the challenges in MWE identification, we propose the bigappy-unicrossy tagging scheme that can represent both nested and crossing MWEs besides discontinuous MWEs.

The tag set of the bigappy-unicrossy tagging scheme consists of the following tags: *B, I, O, b, i, o*. *B* labels the first token of a chunk. It is also used for single-token chunks. *I* labels the other tokens belonging to the chunk if the chunk contains more

than one token. *O* labels tokens outside of the chunk. We use the lowercase tags to represent discontinuous chunks, nested chunks, and crossing chunks. The crossing chunk term refers to chunks with crosswise positioned tokens. *b* labels the first token of a nested or crossing chunk. It is also used for single-token ones. *i* labels the other tokens belonging to the nested or crossing chunk. *o* labels tokens outside of any chunk within a gappy chunk. It is also used for gaps within the nested chunk.

The identification of crossing cases is similar to that of nesting cases, therefore this tagging scheme treats crossing cases and nesting cases in a similar way. For example, there is a sentence including two chunks X and Y. X is the first chunk seen in the sentence. Y is the second chunk that follows the chunk X in the sentence. If the index of the last token of the chunk Y is smaller than the index of the last token of the chunk X, this case is called nesting. But if the middle tokens of them are positioned crosswise, this is called crossing. Also, if the index of the last token of the chunk Y is bigger than the index of the last token of the chunk X, this case is called crossing.

To explain the tagging of nested cases, an example sentence including a discontinuous nested MWE is illustrated in Table 4.1. This sentence is labelled with the IOB tagging scheme, the gappy 1-level tagging scheme and the bigappy-unicrossy tagging scheme. The sentence contains two MWEs: *take seriously* and *make changes*. While *take seriously* is a discontinuous MWE, *make changes* is a discontinuous nested MWE. The IOB tagging scheme labels only the first chunk *take seriously*. Similarly, the gappy 1-level tagging scheme marks only *take seriously*. But it marks the gap tokens with *o*. The bigappy-unicrossy tagging scheme recognizes both *take seriously* and *make changes*. The bigappy-unicrossy tagging scheme labels the gappy token *some* inside *make changes* as well as the gappy tokens *her*, *decision*, *to* inside *take seriously*.

To explain the tagging of crossing cases, an example sentence including crossing MWEs is illustrated in Table 4.1. The sentence is labelled with the IOB tagging scheme, the gappy 1-level tagging scheme and the bigappy-unicrossy tagging scheme. The sentence contains three MWEs: *made changes*, *not only but also*, and *made additions*. The index of the last token of *make changes* is smaller than the index of the last

Table 4.1. An example sentence including a discontinuous nested MWE.

	I	take	her	decision	to	make	some	changes	seriously
IOB	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

token of *not only but also*. The two MWEs are positioned crosswise. The IOB tagging scheme marks only *made changes*. Similarly, the gappy 1-level tagging scheme marks only *made changes*. But it marks the gap tokens with *o*. According to the the bigappy-unicrossy tagging scheme, the tokens of *made changes* are labelled with the uppercase tags, the tokens of *not only but also* are labelled with the lowercase tags. However, the token *additions* inside *made additions* is ignored because the bigappy-unicrossy tagging scheme does not consider the chunks sharing tokens.

Table 4.2. An example sentence including crossing MWEs.

	I	made	not	only	changes	but	also	additions
IOB	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

As a result, the bigappy-unicrossy tagging scheme allows two levels of discontinuity, one level of nesting, and one level of crossing.

To validate our approach, we compare the bigappy-unicrossy tagging scheme with the IOB2 tagging scheme and the gappy 1-level tagging scheme using the BiLSTM-CRF model that we described in Section 4.2 on the VMWE identification task. To obtain baseline results, we choose the IOB2 tagging scheme instead of the IOB1 tagging scheme because it can handle single-token VMWEs that are observed in the text. For

the gappy 1-level tagging scheme, we follow the same rules that we mentioned in Section 4.2. However, we label all tokens of the discontinuous VMWEs consisting of more than two tokens and the nested VMWEs within these VMWEs as the gappy 1-level tagging suggests.

4.4. Representation Learning Methods

In recent years, the advances in representation learning methods have enabled NLP models to automatically discover useful information from data instead of relying on traditional feature engineering methods. In this section, we examine different representation learning methods to respond to the variability challenge in VMWE identification. Considering the state-of-the-art sequence labelling models [14, 15, 23], we focus on character embeddings extracting morphological information from character sequences, morphological embeddings extracting morphological information from morphological features, and the handcrafted features extracting spelling patterns. To encode character sequences into character embeddings, we exploit two different neural networks: CNN and BiLSTM. To encode morphological features into morphological embeddings, we exploit two different input configurations using a BiLSTM network. Moreover, we examine the effect of dropout training on embeddings.

4.4.1. Word Embeddings

We build a baseline model based on the BiLSTM-CRF architecture that we described in Subsection 4.1.3 to show the level of performance that we can obtain without using additional information. This model is called *Base*. The BiLSTM-CRF architecture takes as input only word embeddings. Figure 4.2 shows the architecture of the *Base* model. The sentence *Seçimi salı günü yapacağız* means that *We will make the election on tuesday.*

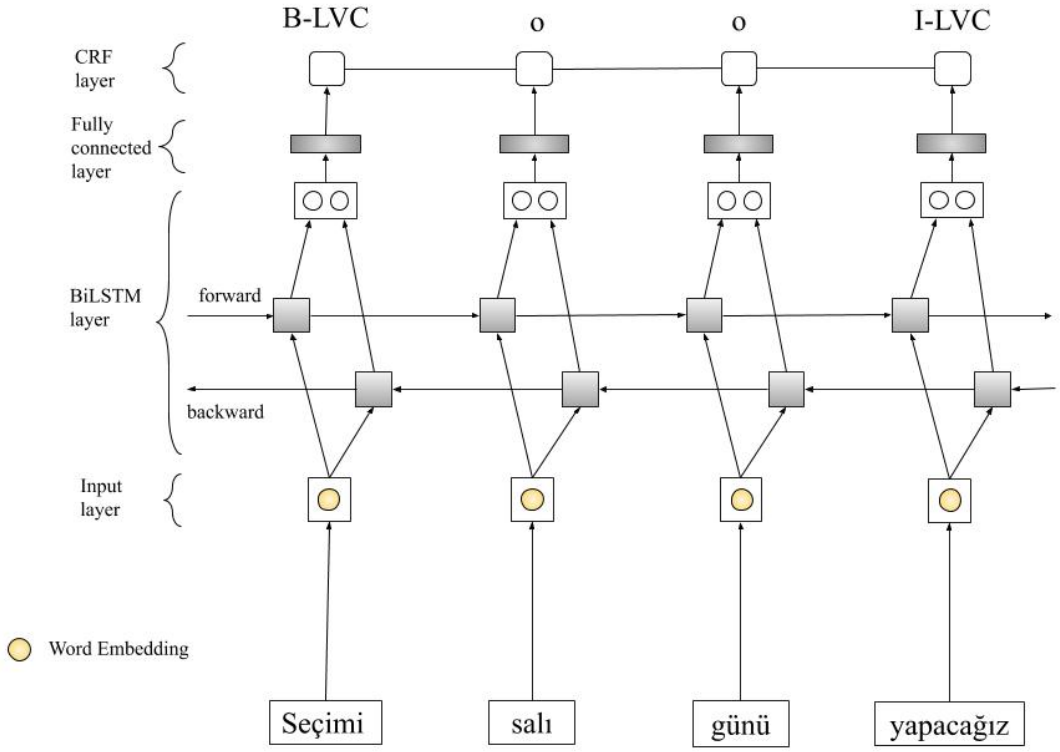

Figure 4.2. Architecture of the *Base* model using word embeddings.

4.4.2. Spelling Features

Despite of the advances in representation learning, some sequence labelling models have utilized the handcrafted features [13, 31]. We build a model to evaluate the effect of the handcrafted features on VMWE identification. This model is called *Spelling*. The following binary features are extracted for each word in a sentence:

- Whether it starts with an uppercase letter
- Whether all letters of it are in uppercase
- Whether all letters of it are in lowercase
- Whether it is numeric
- Whether it includes any digit
- Whether it includes any punctuation

- Whether it includes @
- Whether it is an URL

The binary feature vector is concatenated to the word embedding for each word. The result vector is then given to the BiLSTM-CRF network. Figure 4.3 shows the architecture of the *Spelling* model.

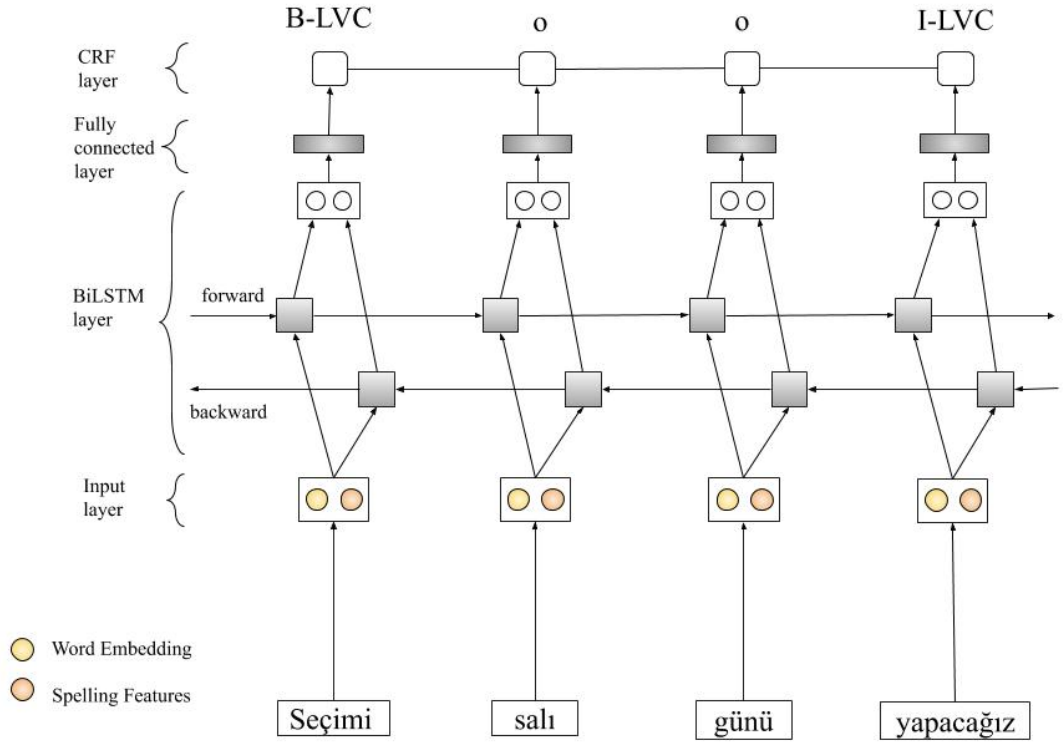

Figure 4.3. Architecture of the *Spelling* model using word embeddings and spelling features.

4.4.3. Character Embeddings

It has been shown that the use of word embeddings with character embeddings has improved the performance of sequence labelling models. Some studies have used CNNs to learn character-level embeddings for sequence labelling tasks [14], others have used BiLSTM networks for the same purpose [15]. Reimers and Gurevych [21]

and Yang *et al.* [22] have reported that the difference between the two approaches is statistically insignificant in general, but the two approaches can give different results for different tasks. Also, it is worth noticing that while character-level BiLSTM networks are designed to extract the prefix and the suffix of a word, character-level CNNs are designed to extract n-gram features of characters. Regarding the findings, we aim to evaluate the impact of network choice on learning character-level representations for VMWE identification. We incorporate two different architectures into the BiLSTM-CRF architecture that we described in Section 4.1.3. Also, we test the effect of adding a dropout layer after the input layer of the BiLSTM-CRF architecture.

First, we integrate a character-level CNN into the BiLSTM-CRF architecture. The character-level CNN consists of three layers: an input layer, a convolutional layer and a max pooling layer. Each character in a word is transformed into a character embedding. The input layer takes as input the character embeddings of the word. The second layer performs convolution operation on the character embeddings. The output of the convolutional layer is fed to the max pooling layer. The max pooling layer generates the character-level representation of the word. Then, the concatenation of character-level representations and word embeddings is given to the BiLSTM-CRF architecture. This model is called *CharCNN*. Figure 4.4 shows the character-level CNN. The word *yapacağız* means that *we will do*. Figure 4.5 shows the architecture of the *CharCNN* model using the character-level CNN.

Second, we apply a dropout mask to the input layer of the BiLSTM-CRF architecture using the character-level CNN. This model is called *CharCNND*. Figure 4.6 shows the architecture of the *CharCNND* model using the character-level CNN with the dropout layer. Dashed arrows represent the dropout layer of the model.

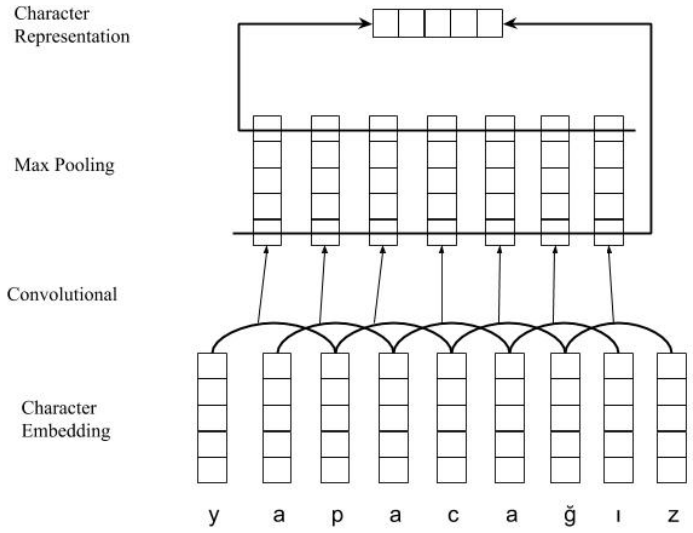

Figure 4.4. Character-level CNN.

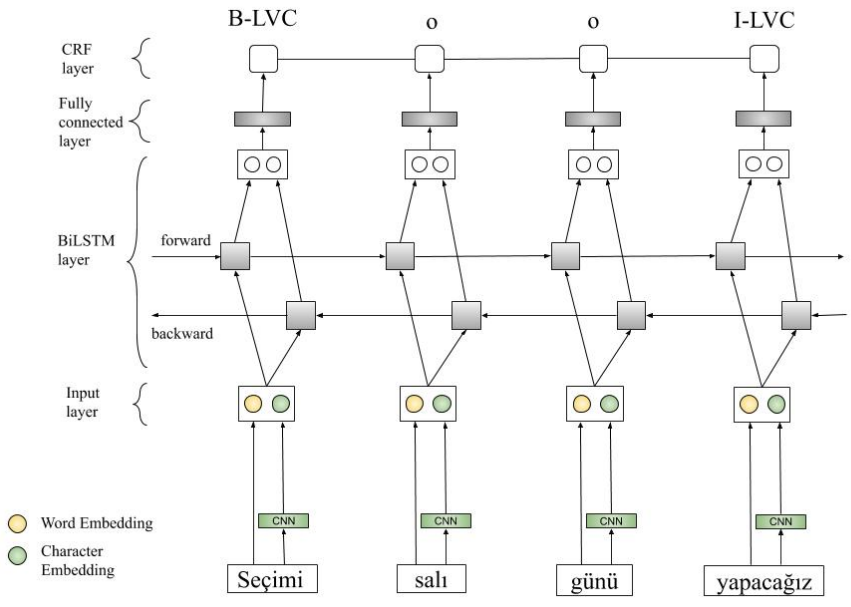

Figure 4.5. Architecture of the *CharCNN* model using word embeddings and character embeddings.

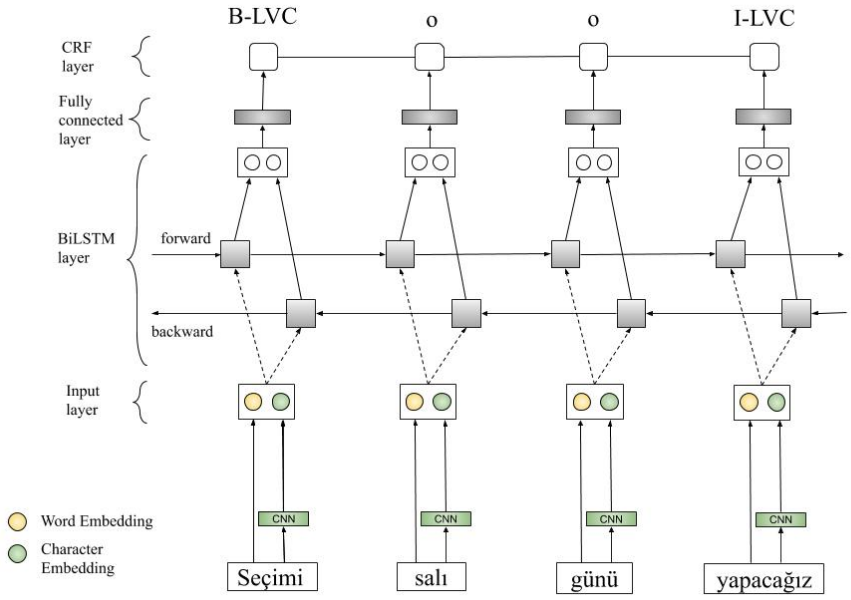

Figure 4.6. Architecture of the *CharCNND* model using word embeddings and character embeddings with dropout layer.

Third, we integrate a character-level BiLSTM network into the BiLSTM-CRF architecture. The character-level BiLSTM network takes as input the character sequences of a word. The output vectors of the forward and backward LSTM units are concatenated to construct character-level representations. Then, the concatenation of word embeddings and character-level representations is given as input to the BiLSTM-CRF architecture. This model is called *CharBiLSTM*. Figure 4.7 shows the character-level BiLSTM network. Figure 4.8 shows the architecture of the *CharBiLSTM* model using the character-level BiLSTM network.

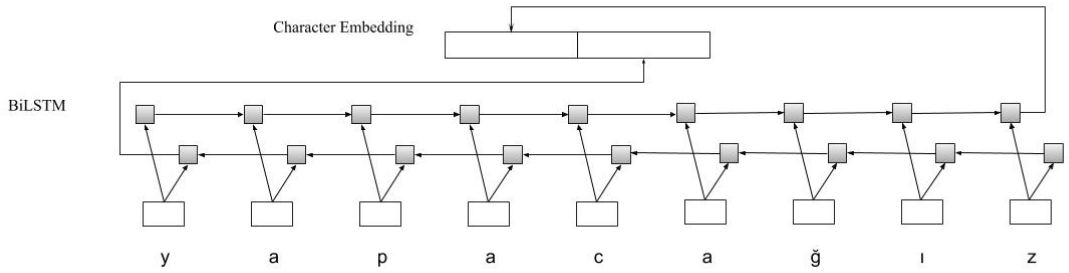

Figure 4.7. Character-level BiLSTM network.

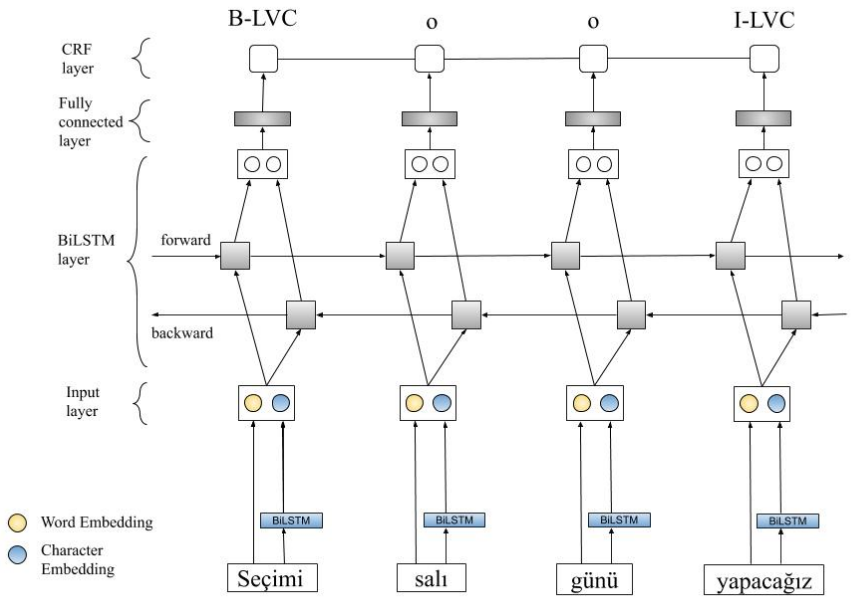

Figure 4.8. Architecture of the *CharBiLSTM* model using word embeddings and character embeddings.

Fourth, we apply a dropout mask to the input vectors of the BiLSTM-CRF architecture using the character-level BiLSTM network. This model is called *CharBiLSTMD*. Figure 4.9 shows the architecture of the *CharBiLSTMD* model using the character-level BiLSTM network with the dropout layer. Dashed arrows represent the dropout layer of the model.

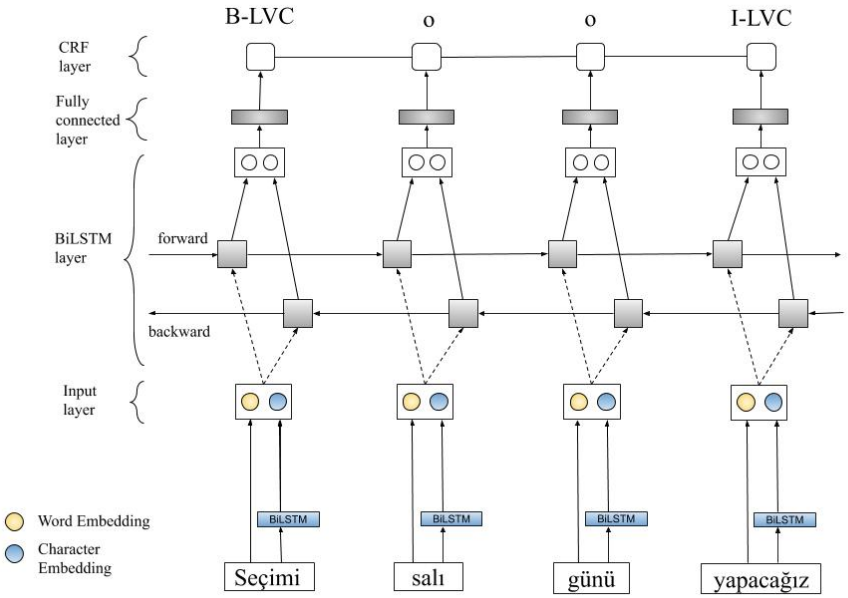

Figure 4.9. Architecture of the *CharBiLSTMD* model using word embeddings and character embeddings with dropout layer.

4.4.4. Morphological Embeddings

Gungor *et al.* [23] have demonstrated that morphological embeddings computed using morphological tags improve NER performance for morphologically rich languages. The main idea beyond this study is that morphologically rich languages can carry additional information in the morphology of the surface forms of words. For example, the Turkish word *yapacağız* means *we will do*. Its morphological representation which is *'Pos+Fut+A1pl'* contains the tense and the number/person agreement information. In the light of the study proposed by Gungor *et al.* [23], we compute morphological embeddings in two ways using a BiLSTM network.

In general, the representation of the morphological analysis of a word is a string that contains a list of morphological features with a list separator like vertical bar (|), plus (+) etc. Accordingly, we form two different morphological embedding configurations as follows:

- *Mor*: This configuration uses the morphological analysis of a word. The morphological analysis of the word is converted into a sequence of morphological features by splitting a symbol.
- *MorChar*: This configuration uses both the lemma and the morphological analysis of a word. It appends the morphological analysis of the word to the lemma of the word. Then, the output string is decomposed into a sequence of characters.

Table 4.3 illustrates the *Mor* and *MorChar* embedding configurations. The morhological analysis of the word *yapacağız* is *'Pos+Fut+A1pl'*. The lemma of the word *yapacağız* is *yap*. In the *Mor* configuration, the morhological analysis *'Pos+Fut+A1pl'* is transformed into the sequence of features *('Pos', 'Fut', 'A1pl')*. In the *MorChar* configuration, the combination of the lemma *yap* and the morhological analysis *'Pos+Fut+A1pl'* is transformed into the sequence of characters *('y', 'a', 'p', '+', 'P', 'o', 's', '|', 'F', 'u', 't', '|', 'A', '1', 'p', 'l')*.

Table 4.3. Morphological Embedding Configurations.

	Morphological Embedding Configuration
Mor	('Pos', 'Fut', 'A1pl')
MorChar	('y', 'a', 'p', '+', 'P', 'o', 's', '|', 'F', 'u', 't', '|', 'A', '1', 'p', 'l')

Based on the two configurations, we develop two different models using morphological embeddings as well as word embeddings. We use the BiLSTM-CRF architecture that we described in Section 4.1.3 as a classifier. For each model, we integrate a BiLSTM network into the BiLSTM-CRF architecture to generate morphological embeddings. The BiLSTM network takes as input the corresponding embedding configuration. Then, it outputs morphological embeddings. The concatenation of morphological embeddings and word embeddings is fed to the BiLSTM-CRF architecture. Also, we add a dropout layer after the input layer of the BiLSTM-CRF architecture and we develop two more models.

Figure 4.10 illustrates the BiLSTM network using the *Mor* embedding configurations to generate morphological embeddings.

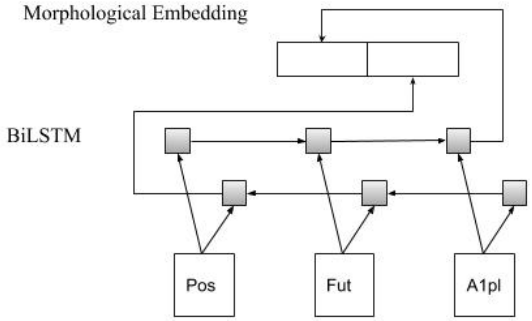

Figure 4.10. BiLSTM network using the *Mor* configuration.

Figure 4.11 illustrates the BiLSTM network using the *MorChar* embedding configuration to generate morphological embeddings. The network treats the combination of the lemma and the morphological analysis of a word as a sequence of characters. This configuration enables the network to learn the relationship between lemmas with the same prefix. Additionally, it allows the network to extract the common parts in relevant morphological features. For example, the morphological feature *A2sg* stands for second person singular, the morphological feature *A2pl* stands for second person plural in Turkish. The common part *A2* means second person agreement. When the morphological features are decomposed into characters, the network can learn this information.

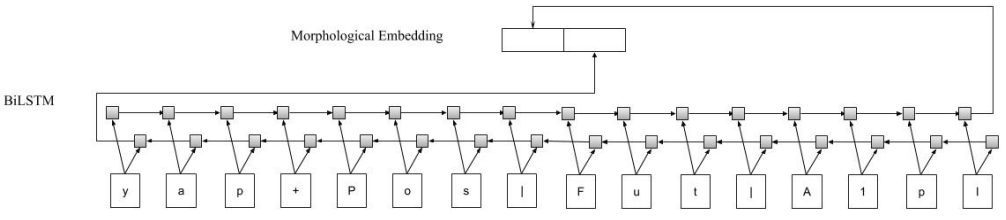

Figure 4.11. BiLSTM network using the *MorChar* configuration.

The first model called *MorBiLSTM* uses the *Mor* configuration. Figure 4.12 shows the architecture of the *MorBiLSTM* model.

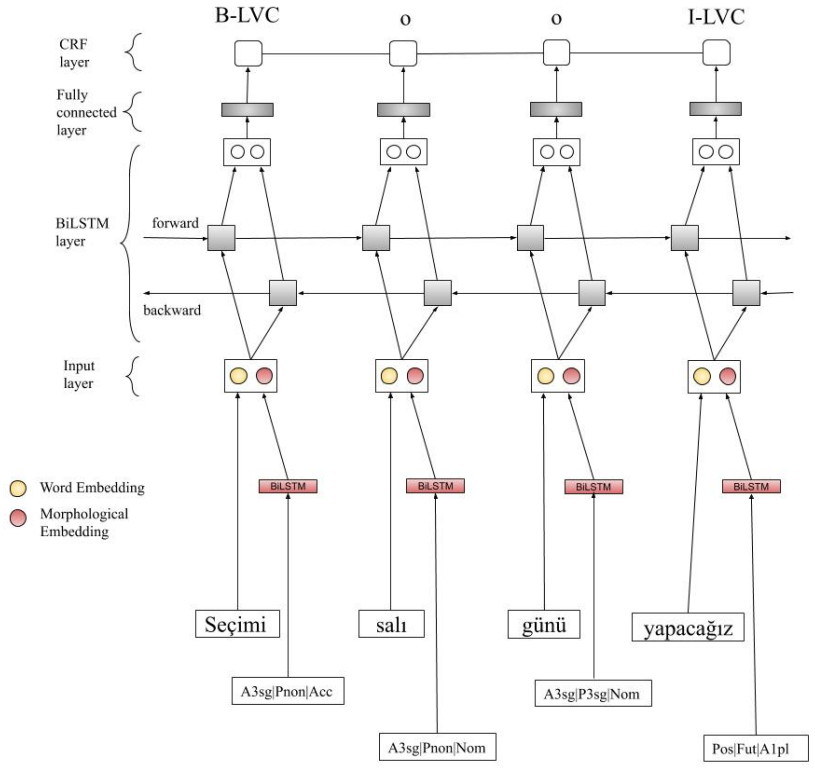

Figure 4.12. Architecture of the *MorBiLSTM* model using word embeddings and morhological embeddings.

The second model called *MorBiLSTMD* uses the *Mor* configuration and includes a dropout layer on the input vectors of the BiLSTM-CRF architecture. Figure 4.13 shows the architecture of the *MorBiLSTMD* model. Dashed arrows represent the dropout layer of the model.

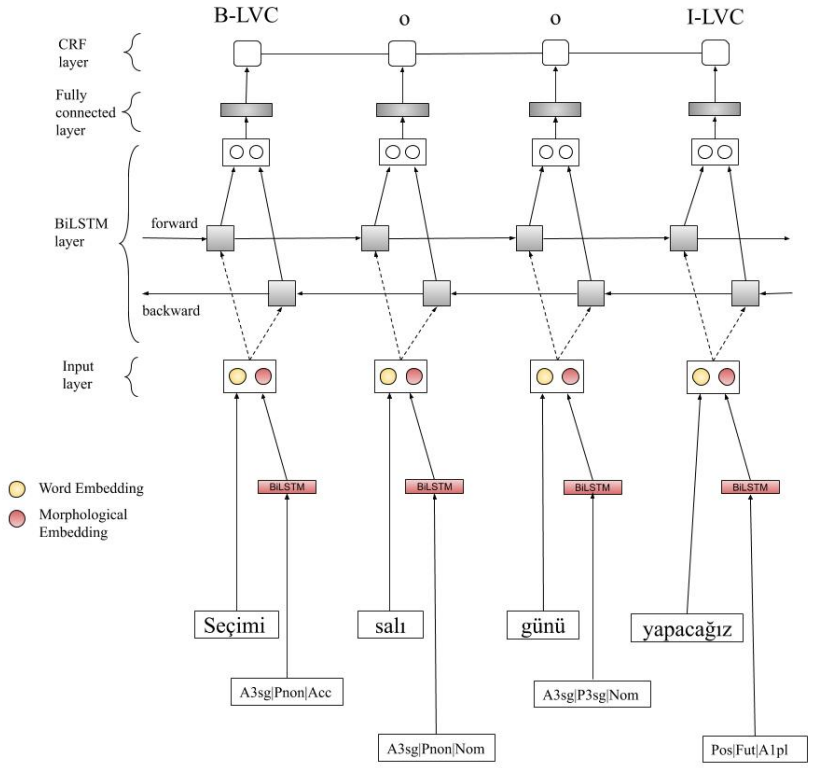

Figure 4.13. Architecture of the *MorBiLSTMD* model using word embeddings and morhological embeddings with dropout layer.

The third model called *MorCharBiLSTM* uses the *MorChar* configuration. Figure 4.14 shows the architecture of the *MorCharBiLSTM* model.

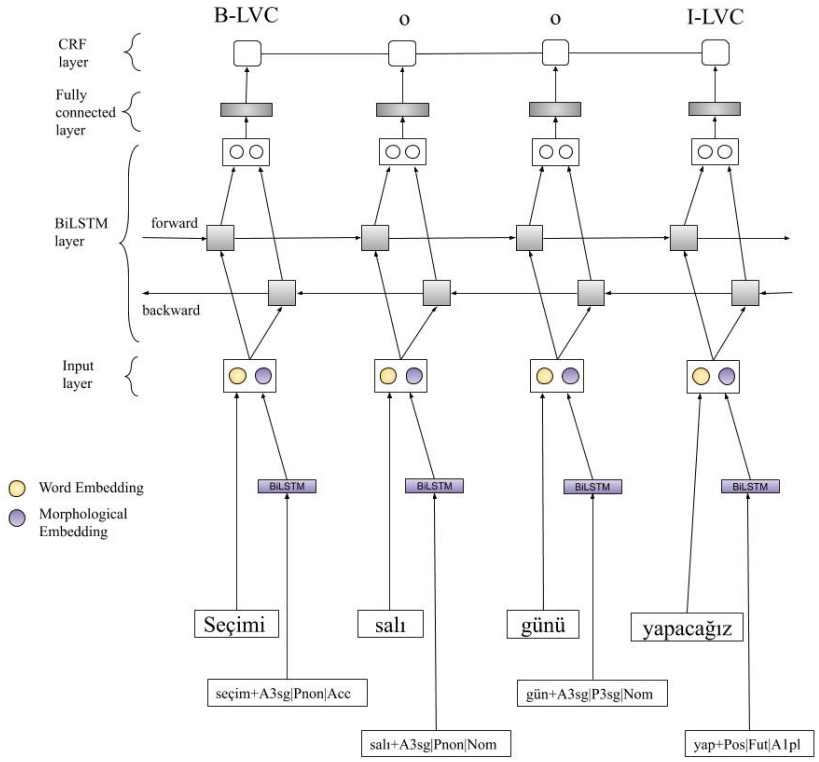

Figure 4.14. Architecture of the *MorCharBiLSTM* model using word embeddings and morhological embeddings.

The fourth model called *MorCharBiLSTMD* uses the *MorChar* configuration and includes a dropout layer on the input vectors of the BiLSTM-CRF architecture. Figure 4.15 shows the architecture of the *MorCharBiLSTMD* model. Dashed arrows represent the dropout layer of the model.

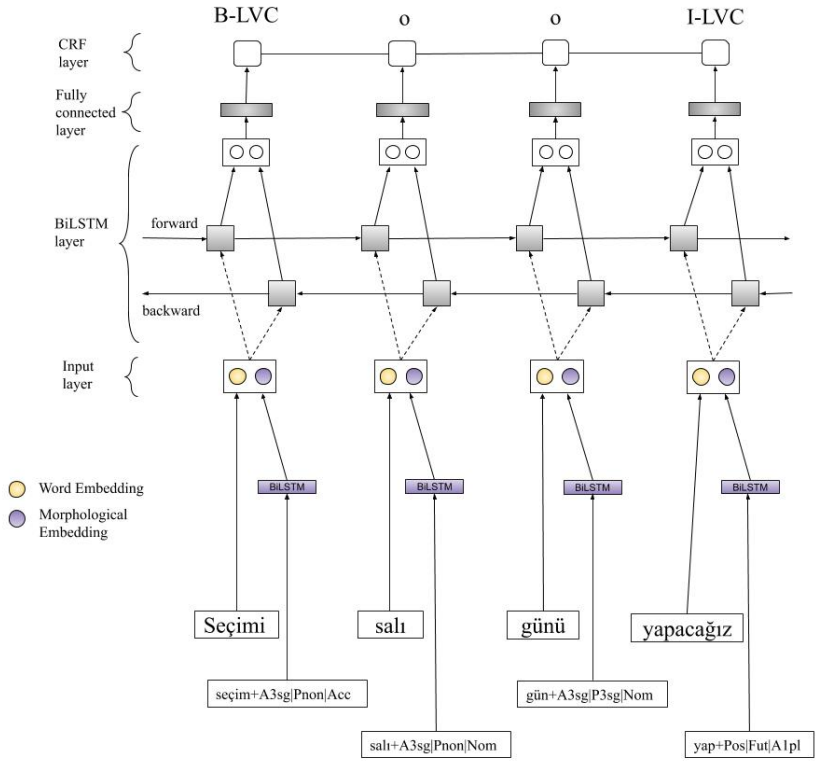

Figure 4.15. Architecture of the *MorCharBiLSTMD* model using word embeddings and morhological embeddings with dropout layer.

5. EXPERIMENTS AND RESULTS

5.1. Statistics About The PARSEME Corpora Edition 1.1

In this Thesis, we use the PARSEME Corpora edition 1.1 in all experiments. Therefore, we first present the statistical results of the PARSEME Corpora edition 1.1 for 19 languages. Each language corpus consists of three sets: the training set, the development set and the test set. If the development set of a language is available, we append its development set to its training set to have more training data. So, the training set refers to the combination of the training set and development set for each language.

Table 5.1 illustrates the number of sentences (Sentences), the number of tokens (Tokens), the total number of the annotated VMWEs (VMWEs) in each language corpus for each set. With respect to the number of sentences, the Romanian training corpus consisting of 49769 sentences is the largest data set among 19 languages. With respect to the total number of VMWEs, the largest training corpus is Hungarian with 6984 VMWEs, the smallest training corpus is English with 331 VMWEs. For all languages, the size of the training corpus is larger than the size of the test corpus except the English and Lithuanian languages. While the English test corpus contains 331 VMWEs, the English training corpus contains 501 VMWEs. Similarly, the Lithuanian test corpus contains 312 VMWEs, but the Lithuanian training corpus contains 500 VMWEs.

The percentages of the discontinuous VMWEs, the variant-of-train VMWEs, the unseen-in-train VMWEs and the single-token VMWEs in the test set for each language are given in Table 5.2. The percentages of the VMWE categories (e.g. IAV, IRV etc.) in the test set for each language are provided in Table 5.3. The symbol ”-” indicates that the language does not have the corresponding category in the test set.

Table 5.1. Statistics about The Parseme Corpora Edition 1.1.

	Training Set			**Test Set**		
Lang	**Sentences**	**Tokens**	**VMWEs**	**Sentences**	**Tokens**	**VMWEs**
BG	19767	441193	6034	1832	39220	670
DE	7918	152734	3323	1078	20559	500
EL	6989	188889	1904	1261	35873	501
EN	3471	53201	331	3965	71002	501
ES	3469	122741	2239	2046	59623	500
EU	9754	138769	3323	1404	19038	500
FA	3258	54076	2952	359	7492	501
FR	19461	488643	5179	1606	39489	498
HE	15491	303315	1737	3209	65698	502
HI	856	17850	534	828	17580	500
HR	3129	73107	1950	708	16429	501
HU	5404	135577	6984	755	20759	776
IT	14472	393496	3754	1256	37293	503
LT	4895	90110	312	6209	118402	500
PL	14821	246495	4637	1300	27823	515
PT	25134	575354	4983	2770	62648	553
RO	49769	900626	5302	6934	114997	589
SL	11517	239999	2878	1994	40523	500
TR	18035	362076	6635	577	14388	506

Table 5.2. Percentages of the discontinuous, the variant-of-train, the unseen-in-train and the single-token VMWEs in the test set for each language.

Lang	Discontinuous	Variant-of-train	Unseen-in-train	Single-token
BG	29	36	33	0
DE	46	59	50	30
EL	45	68	42	0
EN	41	53	71	1
ES	28	52	44	0
EU	19	39	16	0
FR	44	50	50	0
HE	24	41	65	0
HI	7	49	43	0
HR	42	73	45	0
HU	8	21	9	67
IT	33	62	40	0
LT	40	83	55	0
PL	30	60	28	0
PT	43	59	28	0
RO	33	12	5	0
SL	51	73	27	0
TR	59	60	75	1

Table 5.3. Percentages of the VMWE categories in the test set for each language.

Lang	IAV	IRV	LS.ICV	LVC.cause	LVC.full	MVC	VID	VPC.full	VPC.semi
BG	1	38	-	8	41	-	12	-	-
DE	-	8	-	0	8	-	37	42	5
EL	-	-	-	2	61	0	34	2	-
EN	9	-	-	7	33	1	16	29	5
ES	13	24	-	6	17	21	19	0	-
EU	-	-	-	3	82	-	15	-	-
FR	-	22	-	3	32	1	43	-	-
HE	-	-	-	10	42	-	36	12	-
HI	-	-	-	2	64	26	8	-	-
HR	38	24	-	6	26	-	7	-	-
HU	-	-	-	4	21	-	1	63	11
IT	8	19	2	5	21	1	41	5	-
LT	-	-	-	3	57	-	40	-	-
PL	6	48	-	3	29	-	14	-	-
PT	-	16	-	1	61	-	21	-	-
RO	-	62	-	3	6	-	29	-	-
SL	20	49	-	3	7	-	21	-	-
TR	-	-	-	-	54	0	46	-	-

5.2. Deep-BGT at PARSEME Shared Task 2018

In this section, we present the results of the Deep-BGT system in the PARSEME Shared Task edition 1.1. In the shared task, we covered the Romance languages, which are ES, FR, IT, PT, and RO and the languages with the higher frequency of VMWEs which are BG, DE, HU, PL, SL. We did not cover TR not to introduce a bias to system evaluation because we were in the Turkish annotation team. The Deep-BGT system was ranked the second in terms of the general ranking metric in the open track of the PARSEME Shared Task edition 1.1. This work was our initial step towards building a multilingual VMWE identification system.

In this work, we follow the evaluated network configurations for many sequence labelling tasks by Reimers and Gurevych [21] because hyperparameter optimization is time intensive. We apply a dropout rate of 0.1 to the input and recurrent units of the BiLSTM layer for each language. We use the Nadam optimizer without exceeding batch size 32 for each language. We set the node size of the network to 20 for each language. Table 5.4 shows the hyperparameters of the BiLSTM-CRF model for each language.

Table 5.4. Hyperparameters of the BiLSTM-CRF model proposed for the PARSEME Shared Task 1.1.

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

Table 5.5 illustrates the cross-lingual macro average results of the Deep-BGT system over 19 languages and 10 languages in terms of MWE-based F-measure (F1). The official shared task results in the second column are calculated by averaging the success rates for 19 languages. The unofficial shared task results in the third column are calculated by averaging the success rates for 10 languages that we covered. Each

row in the table represents a metric, including the general metrics and the phenomena-specific metrics that we explained in Section 2.2. The system identifies multi-tokens VMWEs better than single-token VMWEs. It is difficult to detect unseen-in-train VMWEs compared to seen-in-train ones, accordingly, the performance of the system for VMWEs unseen in the training data is lower compared to those that occur in both train and test data. With respect to the variability challenge, the success rate for the identical-to-train VMWEs is higher than the variant-of-train VMWEs. Finally, the performance of discontinuous VMWEs is lower than that of continuous VMWEs, as expected.

Table 5.5. Cross-lingual macro average results of Deep-BGT

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

Table 5.6 illustrates the results of Deep-BGT for each language in terms of MWE-based and token-based precision (P), recall (R), F-measure (F1), and rankings in the open track. According to the shared task results, Deep-BGT was ranked first in BG in terms of both MWE-based and token-based F-measure, and was ranked first in DE in terms of MWE-based F-measure. Deep-BGT was ranked first in FR and PL in terms of token-based F-measure. To conclude, the proposed system obtained comparable results to the other systems in the shared task.

Table 5.6. 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

5.3. Representing Overlaps in Sequence Labeling Tasks with a Novel Tagging Scheme: bigappy-unicrossy

In this study, we extended the initial work that we tested on 10 languages to 19 languages. We use the same hyperparameters as the previous work. Similarly, we apply a dropout rate of 0.1 to the input and recurrent units of the BiLSTM layer for each language. We use the Nadam optimizer and the node size of 20 for each language. The parameters of the BiLSTM-CRF model for each language are given Table 5.7. We run our experiments five times in order to maintain reproducible and reliable results and take the average.

Table 5.8 displays the language-specific results for the IOB2, the gappy 1-level and the bigappy-unicrossy tagging schemes. MWE-based and token-based F-measure (F1) are presented for all tagging schemes. The results cover 19 languages. The *shared task* column indicates the F1 score of the system which achieves the best result for each language in the open track of PARSEME Shared Task edition 1.1. The cross-

Table 5.7. Hyperparameters of the BiLSTM-CRF model using different IOB encoding formats.

Languages	**Batch Size**	**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

lingual macro-averages in the last row is calculated by averaging the F1 scores for 19 languages.

In terms of the MWE-based F-measure score, both the bigappy-unicrossy and the gappy 1-level tagging schemes outperform the IOB2 tagging scheme. In terms of the token-based F-measure score, all the three tagging schemes deliver similar performances. The observed increase in the MWE-based F-measure score could be attributed to that representing discontinuous cases with different tags enables a supervised algorithm to capture discontinuous VMWEs.

In addition, while the bigappy-unicrossy tagging scheme is the best in 8 languages, the gappy 1-level tagging scheme is the best in 11 languages. There is a slight difference of 0.38 between gappy 1-level and bigappy-unicrossy. This could be associated with that the frequency of overlapping cases is low in the data set. We claim that if the data set includes all other types of MWEs as well as VMWEs, the overlap frequency will be higher and the bigappy-unicrossy tagging scheme will show better performance. The bigappy-unicrossy tagging scheme can be also used for other sequence labelling tasks including overlapping cases.

Figure 5.1 presents the relationship between the percentage of discontinuous VMWEs in the test corpora for all languages and the relative success of the bigappy-unicrossy tagging scheme over the IOB2 tagging scheme. The language-specific dis-

continuity percentages are given in Table 5.3. The relative success is calculated by subtracting the MWE-based F1 score of IOB2 from that of bigappy-unicrossy. It is observed that the increase in the success with the bigappy-unicrossy scheme is correlated with the discontinuity ratio to some extent. It is possible to hypothesise that the performance of the bigappy-unicrossy tagging scheme increases when the proportion of discontinuous MWEs in a data set increases.

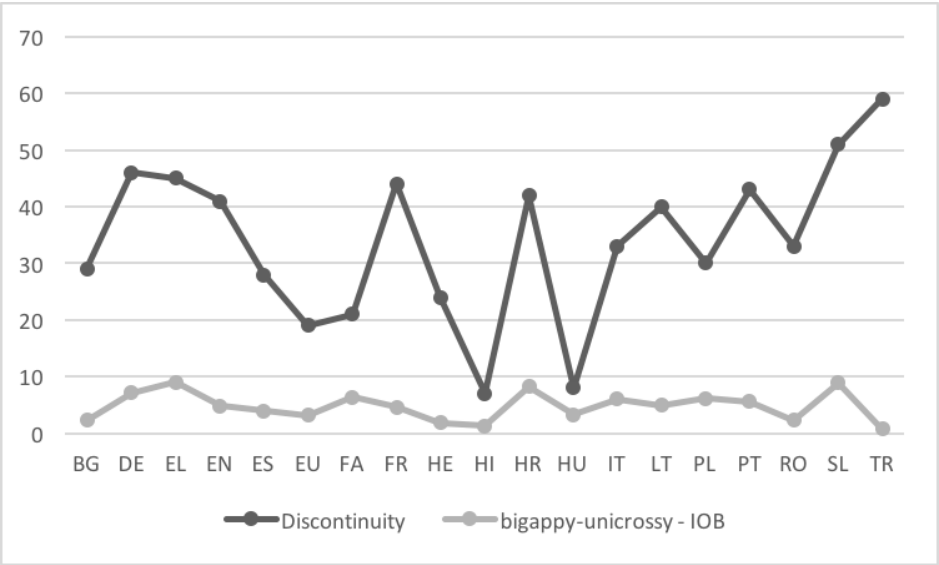

Figure 5.1. Discontinuity percentages versus MWE-based F1 score differences between the bigappy-unicrossy and the IOB tagging schemes.

The experiments demonstrate that the systems using the gappy 1-level and bigappy-unicrossy tagging schemes deliver comparable results to the best shared task systems. The gappy 1-level tagging scheme outperforms the best shared task results in four languages consisting of BG, DE, FR, HI. The bigappy-unicrossy tagging scheme outperforms the best shared task results in five languages consisting of EL, FA, HR, LT, SL. In the case of token-based results, gappy 1-level is the best in BG, DE, EL and bigappy-unicrossy is the best in FA.

Table 5.8. Language-specific results for the IOB2, the gappy 1-level, the bigappy-unicrossy tagging schemes and the best PARSEME shared task results in the open track.

	MWE-based				**Token-based**			
Lang.	**IOB2**	**gappy 1-level**	**bigappy-unicrossy**	**shared task**	**IOB2**	**gappy 1-level**	**bigappy-unicrossy**	**shared task**
BG	64.60	67.03	66.89	65.56	67.21	67.72	67.24	66.85
DE	42.62	50.75	49.73	45.53	54.28	55.10	53.31	54.65
EL	52.10	60.54	61.11	58.00	63.73	66.97	65.61	66.79
EN	26.97	31.60	31.73	33.27	30.15	31.19	30.86	34.36
ES	31.07	33.59	35.00	38.39	37.54	38.64	39.76	44.69
EU	69.91	72.62	73.07	77.04	75.38	75.03	76.06	80.21
FA	75.07	79.31	81.37	78.35	82.01	81.33	84.48	82.95
FR	53.92	61.96	58.55	60.88	65.05	64.78	61.57	65.80
HE	24.93	27.45	26.74	38.91	28.56	29.30	28.74	44.02
HI	71.28	73.35	72.54	72.71	74.06	74.78	74.35	75.62
HR	44.62	51.85	52.83	47.84	53.68	54.58	56.18	58.19
HU	70.53	74.83	73.84	85.83	73.90	76.48	76.13	86.73
IT	31.52	38.17	37.58	45.40	40.28	42.73	43.61	55.13
LT	19.15	22.85	24.04	22.86	25.31	22.89	24.49	28.13
PL	58.54	65.87	64.65	63.60	64.78	67.70	66.41	67.23
PT	54.62	61.32	60.21	68.17	62.29	62.91	62.39	73.51
RO	82.34	85.89	84.60	87.18	85.31	86.33	85.19	88.69
SL	45.30	54.06	54.22	52.27	56.30	56.46	57.50	61.55
TR	52.26	55.95	52.93	58.66	58.12	57.52	54.30	61.63
AVG	*51.12*	*56.26*	*55.88*	*57.92*	*57.79*	*58.55*	*58.33*	*62.99*

5.4. Represention Learning Methods

In this section, we present the results of the following models:

- The *Base* model using the word embeddings.
- The *Spelling* model using the handcrafted spelling features.
- The *CharCNN* model using the word embeddings and the character embeddings obtained by the character-level CNN.
- The *CharCNND* model using the word embeddings and the character embeddings obtained by the character-level CNN with the dropout layer.
- The *CharBiLSTM* model using the word embeddings and the character embeddings obtained by the character-level BiLSTM.
- The *CharBiLSTMD* model using the word embeddings and the character embeddings obtained by the character-level BiLSTM with the dropout layer.
- The *MorBiLSTM* model using the word embeddings and the morphological embeddings obtained by the BiLSTM network using *Mor* embedding configuration.
- The *MorBiLSTMD* model using the word embeddings and the morphological embeddings obtained by the BiLSTM network using *Mor* embedding configuration with the dropout layer.
- The *MorCharBiLSTM* model using the word embeddings and the morphological embeddings obtained by the BiLSTM network using *MorChar* embedding configuration.
- The *MorCharBiLSTMD* model using the word embeddings and the morphological embeddings obtained by the BiLSTM network using *MorChar* embedding configuration with the dropout layer.

We conducted all the experiments for 19 languages using the PARSEME Corpora edition 1.1. Table 5.9 displays the chosen network configurations of the BiLSTM-CRF component for each language. Language (Lang), batch size (Batch Size), number of epochs (Epochs), number of units (Units), and number of epochs for early stopping (Early Stopping) are presented. Additionally, we use the Nadam optimizer and apply a dropout rate of 0.1 to the input units and the recurrent units of the BiLSTM layer in

the BiLSTM-CRF architecture as in our previous works because optimizing parameters for 19 languages is time intensive. We use the same parameters for all models except that we use a batch size of 32 in the *MorCharBiLSTM* and the *MorCharBiLSTMD* models for the RO language due to the limitations of memory to train the RO corpus.

For *CharCNND*, *BiLSTMCNND*, *MorBiLSTMD*, *MorBiLSTMD*, *MorCharBiLSTMD*, we apply a dropout rate of 0.2 to the final embedding layer just before inputting to the BiLSTM-CRF architecture. It should be noted that the dropout technique that we applied to the BiLSTM units in the BiLSTM-CRF architecture is different from this dropout technique. The first one masks the input vectors of each gate in an LSTM unit separately.

For the character-level CNN, we use 30 filters and a window size of 3, which outputs 30-dimensional character embeddings for each word. For the character-level BiLSTM network, we set the number of units to 25, which outputs 50-dimensional character embeddings for each word. For the BiLSTM network learning morphological embeddings, we set the number of units to 25, which outputs 50-dimensional morphological embeddings for each word. Each type of embedding is initialized as a random vector of size 30 with values sampled from a uniform distribution within $[-\sqrt{\frac{3}{30}}, +\sqrt{\frac{3}{30}}]$ [14, 15].

We use the bigappy-unicrossy tagging scheme, and the pre-trained fastText word embeddings. We obtain the morphological features of words from the FEATS field provided in the data set which is in the cupt format. However, the FA corpus does not contain the morphological features, we could not run the *MorBiLSTM*, *MorBiLSTMD*, *MorCharBiLSTM*, and *MorCharBiLSTM* models for the FA language.

Table 5.9. Hyperparameters of the BiLSTM-CRF component of the models using different feature representations for each language.

Lang	Batch Size	Epochs	Units	Early Stopping
BG	32	15	20	5
DE	16	20	20	5
EL	16	20	20	5
EN	8	20	20	5
ES	16	20	20	5
EU	16	15	20	5
FA	16	15	20	5
FR	32	15	20	5
HE	32	15	20	5
HI	16	20	20	5
HR	8	20	20	5
HU	16	20	20	5
IT	16	15	20	5
LT	32	15	20	5
PL	16	15	20	5
PT	32	15	20	5
RO	64	15	20	5
SL	16	15	20	5
TR	64	15	20	5

We run the experiments three times and take the average. Table 5.10 illustrates the language-specific MWE-based F-measure (F1) score for each model. Table 5.11 illustrates the language-specific Token-based F-measure (F1) score for each model. The scores with * indicate the best result for each language. The last row shows the cross-lingual averages for each model. For *Base*, *Spelling*, *CharCNN*, *CharCNND*, *CharBiLSTM*, *CharBiLSTM*, the cross-lingual macro-averages are calculated by averaging the F-measure scores for 19 languages. For *MorBiLSTM*, *MorBiLSTMD*, *MorCharBiLSTM*, *MorCharBiLSTMD*, the cross-lingual macro-averages are calculated by averaging the F-measure scores for 18 languages because the morhological features are not available in the FA corpus and it is excluded.

As seen in Table 5.10, the performance of each model varies according to language in terms of MWE-based F-measure score. For some languages, it is difficult to choose the best representation learning method. For EL, while the *MorBiLSTMD* model outperforms the *Base* model by 2.56% F-measure, the performance of the *MorBiLSTM* model is lower than that of the *Base* model. For EN, the *MorCharBiLSTMD* model has a F-measure score of 31.68%, but the performance of the *MorCharBiLSTM* is at 27.23% F-measure. For FA, *Spelling*, *CharCNND* and *CharBiLSTM* produce similar results. For HE, *CharCNN* delivers the best performance. For TR, each model achieves an improvement over the *Base* model except for *Spelling* and *MorCharBiLSTM*. The *CharBiLSTMD* model yields a 6.17% F-measure improvement over the *Base* model. For the IT and FR languages, the use of morhological information produces the worst results. For IT, the *CharBiLSTM* model brings an 1.80% F-measure improvement over the *Base* model. However, the performance decreases when morhological embeddings computed over morphological features are used. For FR, the *CharBiLSTM* obtains 62.24% F-measure, but the performance sharply decreases when morhological information is used. In addition to these findings, for ES, the character-level information obtained using a CNN and the morphological information computed over morhological features lead to better results than the other representations, but the *CharCNN* yields the highest score for 36.97%. For EU, all models obtain good results compared to the *Base* model, but the *CharBiLSTMD* model gives the best performance. For the HI and PL languages, the use of additional information except the spelling features

results in higher performance compared to the *Base* model. For HI, *MorBiLSTM* and *MorBiLSTMD* produce similar results with the best F-measure scores of 72.49% and 72.09%, respectively. For PL, the use of character embeddings or morphological embeddings yields better results than the *Base* model. For PT, *CharCNN* achieves the best performance, but the difference between the models using character-level CNNs and the models using morhological features as character sequences is so small.

For DE, the performance considerably increases when the network uses character embeddings or morphological embeddings obtained by treating morphological features as character sequences. For HU, the use of character embeddings or morphological embeddings obtained by treating morphological features as character sequences boosts the performance, which is similar to DE. As seen in Table 5.2, this result may be explained by the fact that single-token VMWEs are more frequently observed in the DE and HU test sets.

It can be seen from the data in Table 5.2 that the variability of the expressions in the LT, HR, SL languages is higher than that in the other languages. The percentage of the variant-in-train VMWEs in the LT test corpus is 83%, which the highest proportion among 19 languages. For LT, *CharCNN* gives the best performance compared to the other models. Also, there is a large difference of 15.34% between the *CharCNN* model and the *MorCharBiLSTMD* model which is the worst model for LT. For HR, character embeddings or morphological embeddings improve the performance but the increase in the performance is averaged around 3.5%. For SL, the use of morphological embeddings computed over the characters of morphological features results in the best performance. In contrast to these languages, for RO and BG, most of the VMWEs in the test corpus are identical to the VMWEs in the train corpus. For RO, all models perform similarly. For BG, *CharBiLSTMD* gives the best performance, but there is a small difference of 1.09% between the best model *CharBiLSTMD* and the *Base* model. The impact of additional information is rather small.

Regarding the macro averages given in the last rows of Table 5.10 and Table 5.11, *CharCNN* achieves the best performance among 10 models. In terms of the MWE-

based score, these results suggest that character-level information and morhological information obtained over the characters of morphological features improve the performance of the VMWE identification system. In terms of the token-based score, the use of character-level information leads to slightly better results than the other options .

In general, applying the dropout mask to the input vectors of the BiLSTM-CRF network does not improve the performance. In particular, we observe a sharp decrease in model performance after using the dropout technique for HE and LT, which affects the average results of the models using the dropout mask. This could be linked that the LT training corpus contains the least number of annotated VMWEs among 19 languages, and the HE training corpus contains 1737 VMWEs which is less than average.

Table 5.10. Language-specific MWE-based F-measure scores for each model.

Lang	**Base**	**Spelling**	**Char CNN**	**Char CNND**	**Char BiLSTM**	**Char BiLSTMD**	**Mor BiLSTM**	**Mor BiLSTMD**	**MorChar BiLSTM**	**MorChar BiLSTMD**
BG	67.11	65.07	65.42	66.59	65.81	68.2*	66.45	66.96	67.88	67.24
DE	47.95	49.48	52.57	54.59*	54.05	53.47	48.61	48.38	51.33	52.64
EL	58.64	55.74	59.38	59.33	59.67	59.22	57.04	61.2*	60.19	61.02
EN	27.51	24.97	28.66	29.22	29.2	25.98	27.78	24.4	27.23	31.68*
ES	34.86	35.0	36.97*	36.8	33.82	34.44	36.7	36.12	34.19	32.96
EU	69.72	70.16	72.94	73.0	72.86	74.11*	71.55	71.38	71.66	72.63
FA	77.82	79.24	77.65	79.63*	79.16	78.98	-	-	-	-
FR	60.36	59.22	60.45	61.13	62.24*	59.51	53.61	53.02	53.47	53.9
HE	27.2	29.19	32.13*	22.31	25.56	25.28	27.06	22.5	29.33	24.14
HI	66.71	66.01	68.82	70.45	66.91	67.27	72.49*	72.09	68.98	68.97
HR	48.43	49.96	52.59	52.31	51.44	50.94	53.85*	52.15	52.78	51.43
HU	78.15	75.89	88.84	89.49	89.82	89.51	74.54	72.1	90.05*	89.45
IT	38.83	38.26	39.84	37.17	40.63*	33.28	35.58	35.88	39.48	39.17
LT	25.84	22.56	33.94*	24.33	28.13	26.67	30.78	24.16	28.43	18.6
PL	65.01	64.99	68.45	67.75	68.36	68.27	66.52	68.47*	68.37	68.47*
PT	60.31	59.78	63.95*	63.29	60.51	63.38	61.38	60.12	62.64	62.03
RO	85.32	85.06	85.8	84.54	85.28	85.15	84.66	83.36	86.38*	85.95
SL	53.05	53.04	55.42	55.71	56.65	56.52	53.91	53.46	58.71*	58.66
TR	49.95	49.0	55.3	53.94	53.89	56.12*	53.66	52.66	49.6	54.7
AVG	54.88	54.35	57.85*	56.92	57.05	56.65	54.23	53.24	55.59	55.2

Table 5.11. Language-specific token-based F-measure scores for each model.

Lang	Base	Spelling	Char CNN	Char CNND	Char BiLSTM	Char BiLSTMD	Mor BiLSTM	Mor BiLSTMD	MorChar BiLSTM	MorChar BiLSTMD
BG	67.95	65.34	66.22	67.36	67.2	68.53	67.18	67.55	69.16*	67.99
DE	53.7	53.93	58.02*	56.36	56.84	57.33	53.36	52.85	55.27	56.1
EL	63.4	61.08	65.99	63.98	64.77	62.91	61.83	64.75	63.87	67.21*
EN	28.9	25.66	29.01	29.82	29.65	25.91	27.78	24.39	27.6	31.94*
ES	40.29	39.38	41.79*	40.77	38.07	39.27	41.15	40.04	38.65	37.49
EU	72.11	72.46	75.98	75.65	76.53	76.64*	74.78	73.96	75.15	75.78
FA	80.76	83.37*	80.65	82.5	82.38	82.75	-	-	-	-
FR	64.9	64.77	64.47	65.33	67.14*	66.13	59.66	59.3	60.04	58.13
HE	28.37	31.2	35.97*	22.72	28.57	27.74	29.42	23.85	33.02	25.13
HI	68.5	67.7	70.33	72.29	70.07	68.44	73.14*	72.37	71.71	71.74
HR	54.09	54.77	56.58	55.36	56.84	53.35	58.48*	53.97	56.16	52.75
HU	79.71	77.4	89.11	89.09	89.5	88.4	76.52	74.08	89.74*	88.91
IT	44.81	43.07	44.6	43.39	47.85*	36.39	41.33	40.44	42.11	42.39
LT	26.67	22.32	33.83*	22.75	27.74	25.9	30.0	23.11	27.16	17.54
PL	64.69	66.04	69.29	68.29	69.21	69.12	67.7	68.78	69.83*	69.6
PT	61.96	61.45	66.02*	65.18	62.66	64.79	63.48	61.1	64.56	62.98
RO	86.17	86.12	86.83	85.19	86.13	85.58	85.12	83.94	87.24*	86.67
SL	56.02	56.51	59.26	59.24	60.01	58.38	57.12	56.77	63.19*	61.09
TR	51.87	51.0	55.9	54.46	54.64	56.66*	54.4	53.58	50.23	55.02
AVG	57.62	57.03	60.52*	58.93	59.28	58.64	56.8	55.27	58.04	57.14

The phenomenon-specific MWE-based F-measure (F1) scores are given in Table 5.12. *CharCNN* is better in predicting continuous and discontinuous VMWEs than the other models. In terms of the variant-of-train metric, the integration of the character-level CNN to the *Base* model which refers to the *CharCNN* model improves the performance by 3.93%, which demonstrates that learning n-gram features of characters can help to solve the variability challenge. All models identify seen-in-train VMWEs better than unseen-in-train VMWEs, as expected. In case of handling unseen-in-train VMWEs, the models using character embeddings outperform the other models. For all models, the recognition of single-token VMWEs is more difficult than that of multi-token VMWEs. Regarding each phenomenon-specific metric, it could be concluded that character-level information obtained by CNNs yields a better improvement on performance compared to character-level information obtained by BiLSTM networks and morphological information obtained by BiLSTM networks.

Table 5.12. Cross-lingual phenomenon-specific MWE-based F-measure scores.

Model	Continuous	Discontinuous	Identical-to-train	Variant-of-train	Seen-in-train	Unseen-in-train	Multi-token	Single-token
Base	59.06	40.26	81.34	60.13	71.32	27.23	55.25	7.14
Spelling	58.75	38.95	81.29	58.71	70.54	26.88	54.8	6.96
CharCNN	62.34*	43.19*	85.17*	64.06*	74.96*	29.08	57.86*	8.31
CharCNND	61.94	41.25	82.25	61.73	72.32	29.27	56.58	8.66
CharBiLSTM	61.67	41.89	83.55	63.27	73.73	29.58	57.14	8.54
CharBiLSTMD	61.47	40.91	81.67	61.23	71.81	29.67*	56.26	8.51
MorBiLSTM	58.58	40.53	81.41	60.17	71.33	26.3	54.83	7.39
MorBiLSTMD	57.36	40.0	78.97	57.62	68.9	26.11	53.83	6.81
MorCharBiLSTM	59.93	41.98	81.95	62.48	72.48	27.01	55.63	8.81
MorCharBiLSTMD	59.77	40.35	80.27	61.13	71.17	27.13	55.0	8.87*

Table 5.13 provides the MWE-based F-measure (F1) score per language-family for each model. The scores with * show the best result for each language-family. While morphological information is important for Balto-Slavic and Germanic language families, character-level information is valuable for Romance and Other language families. But, the use of additional information about characters or morphological features usually results in higher performance for Balto-Slavic languages.

Table 5.13. MWE-based F-measure score per language-family for each model.

Model	Balto-Slavic	Germanic	Other	Romance
Base	51.89	37.73	61.17	55.94
Spelling	51.13	37.22	60.75	55.47
CharCNN	55.16	40.62	65.01*	57.4*
CharCNND	53.34	41.9	64.02	56.58
CharBiLSTM	54.08	41.62	63.98	56.5
CharBiLSTMD	54.12	39.73	64.36	55.15
MorBiLSTM	54.3	38.2	59.39	54.39
MorBiLSTMD	53.04	36.39	58.66	53.7
MorCharBiLSTM	55.23*	39.28	61.64	55.23
MorCharBiLSTMD	52.88	42.16*	61.82	54.8

Table 5.14 provides the cross-lingual MWE-based F-measure (F1) scores per VMWE category for each model. The scores with * represent the best model for each category. For the *IAV*, *VPC.full* and *VPC.semi* categories, the models treating the morhological features as character sequences outperform the other models. The models using the character level CNN are better in identifying the *IRV, LVC.cause, LVC.full* and *VID* categories. But, the models using the character level BiLSTM network identify only the *MVC* category better than other models .

Table 5.14. Cross-lingual MWE-based F-measure score per VMWE category for each model.

Model	IAV	IRV	LS.ICV	LVC.cause	LVC.full	MVC	VID	VPC.full	VPC.semi
Base	31.03	64.97	0.0	18.16	47.54	29.67	31.17	35.08	31.59
Spelling	32.55	64.93	0.0	18.52	46.1	25.96	31.5	36.91	32.65
CharCNN	34.28	67.14*	0.0	20.23	49.72*	30.35	34.82*	40.11	34.39
CharCNND	31.87	66.33	0.0	20.99*	49.13	32.76	32.32	40.14	34.03
CharBiLSTM	33.51	66.14	0.0	20.18	49.66	34.42	33.82	39.43	34.35
CharBiLSTMD	31.98	63.86	0.0	18.79	48.59	34.75*	32.1	40.82	33.11
MorBiLSTM	32.37	64.56	0.0	17.04	45.88	32.83	32.66	38.04	30.42
MorBiLSTMD	32.35	63.85	0.0	16.27	45.14	31.87	29.66	37.69	33.11
MorCharBiLSTM	34.4*	64.73	0.0	18.33	47.82	30.88	31.92	39.37	36.02*
MorCharBiLSTMD	34.25	64.89	0.0	17.5	47.38	27.65	30.92	41.82*	33.0

Table 5.15 shows our best result and the best system result in the open and closed track of the PARSEME Shared Task Edition 1.1 for each language in terms of MWE-based F-measure (F1) score. The scores with * show the best result obtained for each language. Our results outperform the best system results for seven languages which are BG, DE, EL, FA, FR, LT, and PL among 19 languages. However, for HE, IT and SL, our results are much lower than the best system results.

Table 5.15. Comparison of our best results with shared task best results for each language

Lang	Our Best Results	Shared Task Best Results
BG	68.2*	65.56
DE	54.59*	45.53
EL	61.2*	58
EN	31.68	33.27*
ES	36.97	38.39*
EU	74.11	77.04*
FA	79.63*	78.35
FR	62.24*	60.88
HE	32.13	38.91*
HI	72.49	72.98*
HR	53.85	55.3*
HU	90.05	90.31*
IT	40.63	49.2*
LT	33.94*	32.17
PL	68.47*	66.96
PT	63.95	68.17*
RO	86.38	87.18*
SL	58.71	64.29*
TR	56.12	58.66*

6. CONCLUSION

In this Thesis, we first updated the PARSEME Turkish train and test corpora that were published in the PARSEME Shared Task 1.0 as the PARSEME Turkish train and development corpora 1.1 based on the PARSEME annotation guideline 1.1. Additionally, we created the PARSEME Turkish test corpus edition 1.1. Then, we developed a multilingual VMWE identification system based on BiLSTM-CRF networks and examined different approaches for responding to the main challenges in VMWE identification. We evaluated different data representation formats to address the discontinuity challenge. We proposed a new tagging scheme called bigappy-unicrossy to resolve overlaps in sequence labelling tasks. Moreover, we analyzed different representation learning methods to address the variability challenge. We concluded that data representation format is important to detect discontinuous cases. Additionally, we observed that character-level information and morphological information improve performance for VMWE identification. The choice of representation learning method depends on language. We validated our approaches on 19 languages which are BG, DE, EL, EN, ES, EU, FA, FR, HE, HI, HR, HU, IT, LT, PL, PT, RO, SL, TR. In future research, we plan to combine character-level information and morphological information to boost the performance of our system. In addition, we plan to integrate contextualized language models into our system such as ELMo [53] and BERT [54].

REFERENCES

1. Sag, I. A., T. Baldwin, F. Bond, A. Copestake and D. Flickinger, "Multiword expressions: A pain in the neck for NLP", *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 1–15, Springer, 2002.

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

3. 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.

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. Graves, A., "Supervised sequence labelling", *Supervised sequence labelling with recurrent neural networks*, pp. 5–13, Springer, 2012.

6. Kudo, T. and Y. Matsumoto, "Chunking with support vector machines", *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pp. 1–8, Association for Computational Linguistics, 2001.

7. Shen, H. and A. Sarkar, "Voting between multiple data representations for text chunking", *Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 389–400, Springer, 2005.

8. Passos, A., V. Kumar and A. McCallum, "Lexicon infused phrase embeddings for

named entity resolution", *arXiv preprint arXiv:1404.5367*, 2014.

9. Luo, G., X. Huang, C.-Y. Lin and Z. Nie, "Joint entity recognition and disambiguation", *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 879–888, 2015.

10. Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, "Natural language processing (almost) from scratch", *Journal of machine learning research*, Vol. 12, No. Aug, pp. 2493–2537, 2011.

11. 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.

12. Sutskever, I., O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in neural information processing systems*, pp. 3104–3112, 2014.

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

14. Ma, X. and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf", *arXiv preprint arXiv:1603.01354*, 2016.

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

16. Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality", *Advances in neural information processing systems*, pp. 3111–3119, 2013.

17. Pennington, J., R. Socher and C. Manning, "Glove: Global vectors for word rep-

resentation", *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

18. Santos, C. D. and B. Zadrozny, "Learning character-level representations for part-of-speech tagging", *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1818–1826, 2014.

19. Santos, C. N. d. and V. Guimaraes, "Boosting named entity recognition with neural character embeddings", *arXiv preprint arXiv:1505.05008*, 2015.

20. Chiu, J. P. and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs", *Transactions of the Association for Computational Linguistics*, Vol. 4, pp. 357–370, 2016.

21. 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.

22. Yang, J., S. Liang and Y. Zhang, "Design challenges and misconceptions in neural sequence labeling", *arXiv preprint arXiv:1806.04470*, 2018.

23. Güngör, O., T. Güngör and S. Üsküdarli, "The effect of morphology in named entity recognition with sequence tagging", *Natural Language Engineering*, Vol. 25, No. 1, pp. 147–169, 2019.

24. Liebeskind, C. and Y. HaCohen-Kerner, "Semantically motivated Hebrew verb-noun multi-word expressions identification", *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1242–1253, 2016.

25. Maldonado, A., L. Han, E. Moreau, A. Alsulaimani, K. Chowdhury, C. Vogel and Q. Liu, "Detection of verbal multi-word expressions via conditional random fields with syntactic dependency features and semantic re-ranking", *Proceedings of the*

13th Workshop on Multiword Expressions (MWE 2017), pp. 114–120, Association for Computational Linguistics, 2017.

26. Boroş, T., S. Pipa, V. B. Mititelu and D. Tufiş, "A data-driven approach to verbal multiword expression detection. PARSEME Shared Task system description paper", *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pp. 121–126, 2017.

27. Waszczuk, J., "TRAVERSAL at PARSEME Shared Task 2018: Identification of Verbal Multiword Expressions Using a Discriminative Tree-Structured Model", *Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, 2018.

28. Legrand, J. and R. Collobert, "Phrase representations for multiword expressions", *Proceedings of the 12th Workshop on Multiword Expressions*, EPFL-CONF-219842, 2016.

29. Klyueva, N., A. Doucet and M. Straka, "Neural Networks for Multi-Word Expression Detection", *MWE 2017*, p. 60, 2017.

30. Stodden, R., B. QasemiZadeh and L. Kallmeyer, "TRAPACC and TRAPACCS at PARSEME Shared Task 2018: Neural Transition Tagging of Verbal Multiword Expressions", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 268–274, 2018.

31. Taslimipoor, S. and O. Rohanian, "Shoma at parseme shared task on automatic identification of vmwes: Neural multiword expression tagging with high generalisation", *arXiv preprint arXiv:1809.03056*, 2018.

32. 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.

33. Boroș, T. and R. Burtica, "GBD-NER at PARSEME Shared Task 2018: Multi-Word Expression Detection Using Bidirectional Long-Short-Term Memory Networks and Graph-Based Decoding", *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pp. 254–260, 2018.

34. 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`.

35. 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, 2018.

36. Losnegaard, G. S., F. Sangati, C. P. Escartín, A. Savary, S. Bargmann and J. Monti, "Parseme survey on MWE resources", *9th International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 2299–2306, 2016.

37. 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.

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

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

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

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

42. 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*, pp. 1–4, 2014.

43. Bengio, Y., P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult", *IEEE transactions on neural networks*, Vol. 5, No. 2, pp. 157–166, 1994.

44. Hochreiter, S. and J. Schmidhuber, "Long short-term memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

45. Graves, A. and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures", *Neural Networks*, Vol. 18, No. 5-6, pp. 602–610, 2005.

46. 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`.

47. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, Vol. 15, No. 1, pp. 1929–1958, 2014.

48. 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, 2018.

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

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

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

52. 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`.

53. Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, "Deep contextualized word representations", *arXiv preprint arXiv:1802.05365*, 2018.

54. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.