Representing Overlaps in Sequence Labeling Tasks with a Novel Tagging Scheme: Bigappy-Unicrossy

Gözde Berk, Berna Erden(✉), and Tunga Güngör

Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey
{gozde.berk,berna.erden,gungort}@boun.edu.tr

Abstract. Multiword expression (MWE) identification can be handled by using sequence tagging approach accompanied with stochastic models and variants of IOB tagging scheme. In this paper, we introduce a new tagging scheme called *bigappy-unicrossy* to rise to the challenge of overlapping MWEs. The bigappy-unicrossy tagging scheme is compared with the two other well-known tagging schemes which are IOB2 and gappy 1-level in the verbal multiword expression (VMWE) identification task using bidirectional Long Short-Term Memory model with a Conditional Random Field layer on top (bidirectional LSTM-CRF). Both the bigappy-unicrossy and the gappy 1-level tagging schemes outperform the IOB2 tagging scheme. The bigappy-unicrossy tagging scheme competes with the gappy 1-level tagging scheme. We believe that our tagging scheme will show better performance on corpora with higher frequency of overlapping cases.

Keywords: IOB tagging scheme · Multiword expressions · Gappy 1-level tagging scheme · Bigappy-unicrossy tagging scheme · Long Short-Term Memory

1 Introduction

Multiword expressions (MWEs) are lexical items consisting of more than one word that show some degree of idiomaticity at the lexical, syntactic, semantic, pragmatic, and/or statistical levels. The idiomatic character of MWEs means that the properties of MWEs cannot be derived from their component items. For example, the semantics of *kick the bucket* whose lexical meaning is *to die* is not predictable from its parts [1].

The processing of MWEs is a key issue for natural language processing (NLP) tasks such as parsing and machine translation. In the context of MWEs, the automatic annotation of MWEs in running text is defined as the task of MWE identification. The techniques used in identifying MWEs can be grouped into

G. Berk and B. Erden—These authors contributed equally to the work.

A. Gelbukh (Ed.): CICLing 2019, LNCS 13451, pp. 622–635, 2023.
https://doi.org/10.1007/978-3-031-24337-0_44

four main categories: rule-based methods, classifiers, sequence tagging models, and parsing [4].

PARSEME (PARSing and Multi-word Expressions) [11], a research community which is involved in the treatment of MWEs, organizes a shared task on automatic identification of verbal multiword expressions (VMWEs) which is one of the subtypes of MWEs. The community aims to compare and evaluate language-independent VMWE identification systems using a gold standard corpus annotated by PARSEME participants. For Edition 1.1. of the PARSEME Shared Task on automatic identification of VMWEs in 2018, the PARSEME network released annotated corpora for 20 languages. The corpora for each language are sampled from various resources such as news, newswire and articles, not restricted to a specific domain. In PARSEME Shared Task, systems can submit their results in two tracks: open and closed. In the open track, systems are allowed to use additional resources like MWE lexicons, raw corpora, word embeddings and so on.

In recent years, deep neural network architectures have been broadly applied for a wide range of NLP tasks, especially for sequence tagging. The most recent sequence tagging approaches [7,8,10] have delivered the state-of-the-art results for part-of-speech (POS) tagging and Named Entity Recognition (NER). Moreover, the performance of the studies can be evaluated on various languages without requiring feature engineering methods. It is possible to address the identification of MWEs in the same sophisticated ways. However, different from other sequence labeling tasks, the discontinuity and overlapping properties inherent in MWEs necessitate developing a special procedure for the MWE identification task.

In this study, we present a novel tagging scheme called *bigappy-unicrossy* to treat MWE-specific challenges using bidirectional Long Short-Term Memory with a Conditional Random Field (BiLSTM-CRF) neural network architecture proposed by [7]. Additionally, we test our proposed solution on the corpora released in Edition 1.1. of the PARSEME Shared Task for 19 languages. In our experiments, we juxtapose *bigappy-unicrossy* with IOB2 [14] and gappy 1-level [17] tagging schemes. All in all, the difficulties in identification of MWEs can be tackled in some degree by representing all gappy words and crossing boundaries of nested MWEs in sequences.

2 Related Work

In the literature on MWEs, several variations of standard IOB tagging scheme have been evaluated and discussed so far. The IOBES tagging scheme within a supervised approach based on a neural network model is used by [9]. A new tagging scheme named as gappy (discontinuous) 1-level tagging is described by [17] in order to encode gappy MWEs.

In Edition 1.1. of the PARSEME Shared Task, the participated systems exploited several neural architectures. While SHOMA [18], which is the best performing system with respect to the overall macro-average MWE-based F1

score, employs a combination of convolutional network, bidirectional long-short term memory (BiLSTM) network and conditional random field (CRF), Deep-BGT [2] makes use of BiLSTM-CRF with the gappy-1 level tagging scheme in the open track. The two systems use pre-trained word embeddings released by fastText [6]. Mumpitz [5] uses only BiLSTM layer, while GBD-NER [3] adds a graph-based encoding layer to the BiLSTM layer. Besides Deep-BGT, Veyn [19] also presents a recurrent neural network (RNN) model that combines three different tagging schemes.

Similar to Deep-BGT and SHOMA, we decide to explore a neural network architecture and benefit from the availability of fastText word embeddings for several languages. Also, Deep-BGT adopts an advanced tagging scheme to its system, which is different from other competitors, but it does not extend it to all languages. Therefore, we choose to implement the same model which is bidirectional LSTM-CRF for all languages with our new tagging scheme.

3 Corpus

The corpora provided by the PARSEME Shared Task Edition 1.1 consist of 20 languages. We cover 19 languages which are Bulgarian (BG), German (DE), Greek (EL), English (EN), Spanish (ES), Basque (EU), Farsi (FA), French (FR), Hebrew (HE), Hindu (HI), Crotian (HR), Hungarian (HU), Italian (IT), Lithuanian (LT), Polish (PL), Portuguese (PT), Romanian (RO), Slovenian (SL), and Turkish (TR). We do not use the Arabic (AR) corpus because it is not publicly available yet. The corpus of each language is divided into training, test, and development sets. Most of the languages contain more than 2000 VMWEs but EN and LT include less than 1000 VMWEs which are the smallest corpora among the datasets. The language specific statistics are summarized in Table 1. It contains information about number of tokens and VMWEs in each training corpus. Also, percentage of VMWEs and percentage of discontinuous VMWEs in each test corpus are provided.

The corpus is released in cupt format [11], which is publicly available in [12]. The cupt format is the extension of the conllu format[1]. The current format represents each token in a sentence by 11 columns. The first 10 columns specify the rank, token, lemma, part-of-speech, morphological features, and syntactic dependencies, as in the conllu format. The 11th column introduces the VMWE annotation.

The language team members have annotated the corpora following the annotation guidelines prepared by PARSEME [11]. The categories of VMWEs are the followings:

- Universal categories which exist in all participated languages:
  - Light Verb Constructions with two subtypes (LVC.full and LVC.cause)
  - Verbal Idioms (VID)

[1] http://universaldependencies.org/format.html.

- Quasi-universal categories which some languages include:
  - Inherently Reflexive Verbs (IRV)
  - Verb-Particle Constructions with two subtypes (VPC.full and VPC.semi)
  - Multi-verb Constructions (MVC)
- Language-specific categories
  - Inherently Adpositional Verbs (IAV)
- Optional experimental category which is added after the annotation process:
  - Inherently Clitic Cerbs (LS.ICV).

Table 1. Language-specific statistics for the corpora.

Languages	# of tokens	# of VMWEs	VMWE %	Discontinuity %
BG	480413	6704	1.40	29
DE	173293	3823	2.21	46
EL	224762	2405	1.07	45
EN	124203	832	0.67	41
ES	182364	2739	1.50	28
EU	157807	3823	2.42	19
FA	61568	3453	5.61	21
FR	528132	5677	1.08	44
HE	369013	2239	0.61	24
HI	35430	1034	2.92	7
HR	89536	2451	2.74	42
HU	156336	7760	4.97	8
IT	430789	4257	0.99	33
LT	208512	812	0.39	40
PL	274318	5152	1.89	30
PT	638002	5536	0.89	43
RO	1015623	5891	0.58	33
SL	280522	3378	1.20	51
TR	376464	7141	1.90	59

4 MWE Identification

According to [4], MWE identification is the process of annotating MWE instances in a given corpus and an automatic annotation system is called MWE tagger. Most of the NLP applications such as parsing, machine translation, etc. are in need of MWE identification [4]. However, MWE identification is not straightforward since it brings about some challenges. These challenges also require specialized metrics to evaluate the success of an MWE tagger.

4.1 Challenges

Challenges for MWE identification are listed as discontinuity, overlaps, ambiguity, and variability [4]. These challenges stem from the nature of MWEs. There may be tokens other than the ones belonging to MWE between the tokens of the MWE. For example, *take seriously* can have other tokens in between to be in the form of *take someone/something seriously*, but the VMWE is still *take seriously* here. Figure 1 shows an example use. Discontinuity varies from language to language [4]. This variability can be seen in Table 1.

Overlaps include different cases such as nesting, shared tokens, and so on [4]. Nesting can be defined as having at least one MWE inside an another MWE. We can give the sentence *I took her decision to move on seriously* in Fig. 1 as an example. Here, we have two VMWEs which are *took seriously* and *move on*. So, *move on* is referred as a nested VMWE. Two or more MWEs can also share one or more tokens. Additionally, there is a special case of overlaps which is called crossing. In this case, some or all tokens of different MWEs are positioned crosswise. The sentence *I made not only changes but also additions* is an example of both shared tokens and crossing. In this example, *not only but also* is an MWE because it is a complex function word. Both *made changes* and *made additions* are VMWEs belonging to LVC.full category. As a result, *made changes* and *made additions* together is an example to shared tokens case. The example to crossing case is *made changes* and *not only but also*. The example sentence also contains nesting due to *made additions* and *not only but also*. Moreover, we can infer that overlaps also contain the challenge of discontinuity.

Ambiguity stems from the fact that the group of tokens can be either an MWE or a non-MWE [4]. In other words, tokens can lose their original meanings or each token can contribute to the sentence with its original meaning. Also, the same MWE can belong to different types of MWE. An example can be given by these two sentences: *He teaches mathematics as well as physics* and *His performance in mathematics is as well as his performance in physics*. Here, *as well as* is ambiguous.

MWEs do not always appear in fixed forms. This flexibility is called variability [4]. Regarding the example in Fig. 1, *take seriously* changes its form and becomes *took seriously*.

4.2 Evaluation Metrics

Evaluation metrics are important to correctly determine the quality of a model in machine learning. However, evaluating a model only with general accuracy score may not explain how much the model resolves the domain-specific problems. Therefore, the PARSEME network [11] defines MWE-specific evaluation metrics which focus on the following challenges:

- Continuity metric is calculated for continuous (EN: *set up* a meeting) and discontinuous (EN: *set* me *up*) cases.
- Length metric is calculated separately for single-token VMWEs (DE: *aufmachen*) and multi-token VMWEs (DE: *macht* es *auf*).

– In terms of Novelty metric, if a VMWE is annotated at least once in the training corpus, it is accepted as "seen"; otherwise it is considered as "unseen".
– Variability metric is provided for seen VMWEs which are not identical to their original form in the training corpus.

Also, MWE-based score is calculated over fully predicted VMWE sequences, and token-based score is calculated over partial matches. In this paper, we evaluate our system results based on the aforementioned metrics.

5 Tagging Schemes

In this work, we approach the MWE identification task as a sequence labeling problem together with IOB encoding. However, the challenges of discontinuity and overlaps cannot be addressed using the classical IOB tagging schemes [13, 14]. Therefore, we make use of the gappy 1-level tagging scheme [17] but still it is not adequate for overlaps. For this reason, we developed a novel tagging scheme called *bigappy-unicrossy* to rise to the challenge of overlapping MWEs. Furthermore, we accept that MWEs can be both single-token and multi-token. Since there is a difference between the IOB1 [13] and the IOB2 [14] tagging schemes regarding the single-token ones, we applied the IOB2 tagging scheme. We also modified the gappy 1-level tagging scheme because it does not accept single-token MWEs in its original definition.

5.1 IOB1 Tagging Scheme

The IOB tagging scheme was first proposed by [13] which approach text chunking as a tagging problem. In this scheme, the tag set is $\{I, O, B\}$. I represents a token in the chunk, B stands for a token which is the beginning of the chunk spanning more than one token, and O denotes a token outside of any chunk. Therefore, B cannot be used alone without I. In other words, a single-token chunk gets the I tag. The IOB tagging scheme is also called the IOB1 tagging scheme in the literature [16] after the IOB2 tagging scheme is introduced.

5.2 IOB2 Tagging Scheme

The IOB2 tagging scheme is derived from the idea of giving B tag to every initial token of the chunk ignoring the chunk size [14,16]. It has same tag set with IOB1 which is $\{I, O, B\}$. B represents a token in the beginning of the chunk, I is used for a token in the chunk other than the initial token, and O is used for a token outside of any chunk. Hence, the only difference is that each chunk begins with B and B is followed by I if the chunk contains more than one word. In other words, a single-token chunk gets the B tag. We think that the IOB2 tagging scheme is more suitable for MWE identification by taking single-token MWEs into consideration. Hence, this tagging scheme rather than the IOB1 scheme is used in the experiments to create a baseline for comparing the performance of the other tagging schemes.

5.3 Gappy 1-Level Tagging Scheme

Both of the IOB tagging schemes are more suitable for tagging continuous chunks. However, MWEs include not only continuous chunks but also discontinuous chunks. The nature of MWEs can also pose nesting which cannot be represented by the IOB tagging schemes. Therefore, a new tagging scheme which is called the gappy 1-level tagging scheme is introduced by [17].

The tag set of the gappy 1-level tagging scheme is {*I, O, B, i, o, b*}. *I, O, B* tags are similar to the ones in the IOB tagging schemes. *I* symbolizes a token in the chunk, *B* represents a token which is at the beginning of the chunk, and *O* is used for a token outside of the chunk. Since [17] accepts MWEs as chunks containing more than one word, the difference between the IOB1 and IOB2 tagging schemes disappears. So, all *B* tags are followed by one or more *I* tags.

i, o, b tags have similar roles as *I, O, B* tags. The only difference is that the lowercase tags are used for nested chunks. *i* symbolizes a token in the nested chunk, *b* represents a token which is at the beginning of the nested chunk, and *o* is used for a token outside of the nested chunk which is also a gap for the chunk outside. Again, all *b* tags are followed by one or more *i* tags. Figure 1 shows an example use of this scheme.

The gappy 1-level tagging scheme accepts only multi-token MWEs. According to our definition of an MWE, we allow single-token MWEs too. Therefore, we modified the gappy 1-level tagging scheme such that it also allows single-token MWEs in the IOB2 tagging scheme fashion by using the *B* tag for the single-token MWEs that are not nested and the *b* tag for the nested single-token MWEs. The modified version is used in the experiments and it is referred as *gappy 1-level*.

The gappy 1-level tagging scheme solves the discontinuity and nesting problems partially. The discontinuity problem is solved by the *o* tag. Nesting problem which is a particular case of overlaps is solved partially because the gappy 1-level tagging scheme accepts only continuous nested MWEs.

5.4 Bigappy-Unicrossy Tagging Scheme

The variants of IOB tagging schemes propose partial solutions to the challenges of MWE identification. The IOB2 tagging scheme only identifies continuous chunks. The gappy 1-level tagging scheme partially solves the discontinuity problem because it does not allow discontinuous nested MWEs. Due to the elimination of discontinuous nested MWEs, nesting is also partially solved. Also, there is no attempt to solve other cases of overlaps such as crossing and shared tokens and so on. For this reason, there is a need of a new tagging scheme. We developed a novel tagging scheme called *bigappy-unicrossy* to represent overlaps in sequence labeling tasks and to solve the discontinuity problem accordingly.

The tag set of the bigappy-unicrossy tagging scheme is {*I, O, B, i, o, b*}. *B* stands for the beginning of the chunk. It is also used for single-token chunks. *I* represents a token in the chunk. It is used in the case of multi-token chunks where *B* is followed by one or more *I*. *O* is used for a token outside of the chunk.

The bigappy-unicrossy tagging scheme allows two levels of discontinuity, one level of nesting, and one level of crossing. The name *bigappy-unicrossy* is given accordingly. *i, o, b* tags are in charge of nested chunks, chunks with crosswise positioned tokens, and discontinuous chunks.

The *o* tag is used for a token outside of the nested chunk. It is also used for a token that does not belong to the nested chunk but between the tokens of the nested chunk. Tokens that are in between of the chunk but does not belong to the chunk can be called gaps [17]. In other words, *o* is used for all gappy chunks and each gap is tagged with *o*. We treated all the gaps the same. We do not differ the gaps between a chunk or a nested chunk because we assume that the tokens belonging to the gaps have the same role, which is being a token that can be inserted to the chunk without being a part of the chunk.

As an example, consider the sentence *I take her decision to make some changes seriously. take seriously* and *make changes* are VMWEs. The gaps for *take seriously* are *her*, *decision*, *to*, *make*, *some*, and *changes*. Since *make changes* is a nested VMWE here, the actual gaps are *her*, *decision*, *to*, and *some*. The gap for *make changes* is *some*. *some* has the same role for both of the VMWEs and tagged with *o*. Consequently, bigappy-unicrossy handles two levels of discontinuity: one level for gaps in the outer chunks and one level for gaps in the nested chunks.

Lowercase tags resemble uppercase tags in terms of their roles. *b* stands for the beginning of the nested or crossy chunk. It is also used for single-token ones. *i* represents a token in the nested or crossy chunk. It is used in the case of multi-token chunks where *b* is followed by one or more *i*. *o* is used for a token belonging to a gap of the gappy chunk.

In this tagging scheme, crossing cases and nesting cases are treated in a similar way, because the identification of crossing cases are like nesting cases. Here is our tagging procedure: Firstly, the first token of the chunk (we can call it chunk X) is identified if it is the first chunk appeared in the sentence and it is tagged with *B*. If the chunk X is multi-token, the remaining part is tagged with *I*. If there is a token belonging to beginning of another chunk (we can call it chunk Y) within the chunk X, the chunk Y can be nested and/or positioned crosswise with chunk X. Then, it is tagged with *b*. If the chunk Y is multi-token, the remaining part is tagged with *i*. Here, the index of the last token of chunk Y can be either smaller or bigger than the index of the last token of chunk X. The first case is called nesting and the latter one is called crossing. If the middle tokens of chunk X and chunk Y are positioned crosswise in the first case, there is also crossing.

The bigappy-unicrossy tagging scheme allows only one level of nesting or crossing because we have only two types of tag sets which are uppercase and lowercase. Here, the B and I tags belong to chunk X. So, they are held until the end of chunk X. After the last token of X, the B and I are released. The same rule applies for the b and i tags. The b and i tags belong to chunk Y. So, they are held until the end of chunk Y. After the last token of Y, the b and i are released.

On the other hand, it solves discontinuity problem in two levels. Incorporating more levels is not necessary since such a case is very scarce. Some examples are given in Table 2. Example 3 shows a crossing case. Other examples include nesting cases. Shared tokens are ignored in this tagging scheme. Table 2 shows different ways of eliminating shared tokens. In Example 3, *made additions* is eliminated. In example 4, *made changes* is eliminated. Therefore, it overcomes the challenge of overlaps partially.

Table 2. Examples to the bigappy-unicrossy tagging scheme.

Example 1	I	took	her	decision	to	move	on	seriously	
	O	**B**	**o**	**o**	**o**	**b**	**i**	**I**	
Example 2	I	take	her	decision	to	make	some	changes	seriously
	O	**B**	**o**	**o**	**o**	**b**	**o**	**i**	**I**
Example 3	I	made	not	only	changes	but	also	additions	
	O	**B**	**b**	**i**	**I**	**i**	**i**	**O**	
Example 4	I	made	not	only	changes	but	also	additions	
	O	**B**	**b**	**i**	**o**	**i**	**i**	**I**	

6 Model and Experiments

We design a language-independent system based on the bidirectional LSTM-CRF model provided by [7]. Similar to Deep-BGT system [2], we make use of the pre-trained word embeddings provided by fastText [6]. The word embeddings were trained on Common Crawl and Wikipedia. The dimension of word embedding vector is 300. In addition to the word embeddings, we choose the POS and dependency relation (DEPREL) tags that are available in the cupt files as inputs for the system.

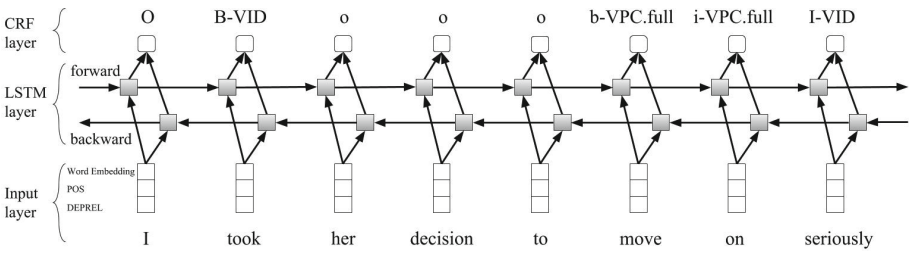

Fig. 1. The bidirectional LSTM-CRF model of Deep-BGT [2].

As shown in Fig. 1, the architecture of the bidirectional LSTM-CRF network composes of three layers. The inputs are fed into the BiLSTM layer. The bidirectional LSTM network processes both past and future features, respectively, in the

forward and backward units whose dimensions are set to 20. The outputs of the LSTM units pass to the CRF layer, which decodes the VMWE labels. We optimize the parameters of the model for each language with the Nadam optimizer without exceeding batch size 32 as suggested by [15]. All chosen hyperpameters of the model are displayed in Table 3. A fixed dropout rate of 0.1 is applied on all the bidirectional LSTM layers. Since the size of the training data plays a major role in deep learning models, we add the development set to the training set if a language has a development set and do not make use of the development set separately. As we use non-deterministic approach, we run our experiments five times in order to maintain reproducible and reliable results and take the average.

Table 3. Model parameters.

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

7 Results

Table 4 shows 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. Each language also has the F1 score of the system which is the best for that language in the open track of PARSEME shared task Edition 1.1 and it is referred as *shared task*. The last row in the table shows the cross-lingual macro-averages which is calculated by averaging the F1 scores for 19 languages.

According to the MWE-based results, both the bigappy-unicrossy and the gappy 1-level tagging schemes outperform the IOB2 tagging scheme. The reason behind is that bigappy-unicrossy and gappy 1-level capture discontinuous VMWEs whereas IOB2 cannot capture them.

On the other hand, there is a slight difference of 0.38 between gappy 1-level and bigappy-unicrossy. The gappy 1-level tagging scheme is the best in 11 languages while the bigappy-unicrossy tagging scheme is the best in 8 of them. The results are close in this experiment because the frequency of overlapping cases is low in the corpora of languages used for the experiment. The corpora only contains VMWEs. In the case of all other types of MWEs, the overlap frequency will be higher and the bigappy-unicrossy tagging scheme will show better performance. The bigappy-unicrossy tagging scheme is not only for MWEs. It can be

also used in other sequence labeling tasks. The bigappy-unicrossy tagging scheme can prove itself better in the other domains of NLP or in their combinations.

The F1 scores are close to each other for all the three tagging schemes in terms of the token-based results in Table 4. While the tagging scheme becomes more complex, identification also becomes more complex in the case of deep learning systems.

The experiments reveal that our application of the gappy 1-level and bigappy-unicrossy tagging schemes competes with the MWE-based best shared task results. When the tagging schemes and the best shared task results are compared, it is observed that gappy 1-level surpasses the best shared task results in 4 languages consisting of BG, DE, FR, HI and bigappy-unicrossy surpasses the best shared task results in 5 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 4. The 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.

Lang.	MWE-based				Token-based			
	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*

Figure 2 shows the relationship between the percentage of discontinuous VMWEs in the corpora for all languages and the relative success of the bigappy-unicrossy tagging scheme over the IOB2 tagging scheme. Discontinuity percentages are also available in Table 1 which provides the language-specific statistics. The relative success is found by subtracting the MWE-based F1 score of bigappy-unicrossy from that of IOB2. It is seen that there is a correlation to some extent between the discontinuity ratio and the success improvement with the bigappy-unicrossy scheme. This result denotes that the effect of the bigappy-unicrossy tagging scheme increases more on discontinuous MWEs.

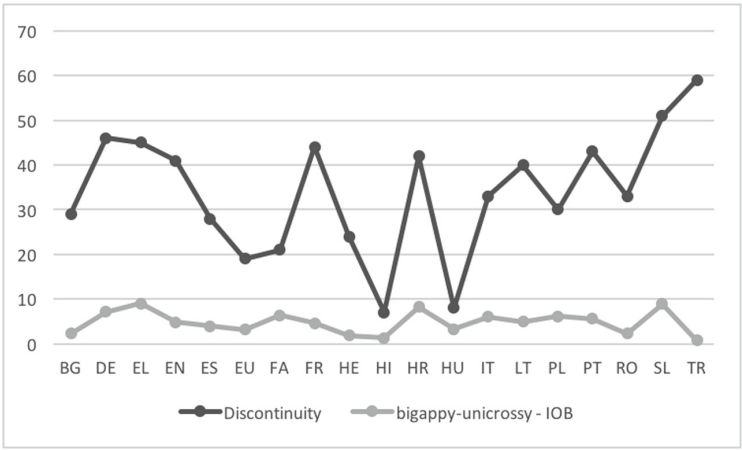

Fig. 2. The discontinuity percentages versus the MWE-based F1 score differences between the bigappy-unicrossy and the IOB tagging schemes.

8 Conclusion

In this study, we described a new tagging scheme called bigappy-unicrossy to address overlaps in sequence labeling tasks. It is attempted to solve the challenges of discontinuity and overlaps that include nesting and crossing.

Additionally, we presented an empirical study that explores the effect of a tagging scheme for VMWE identification on 19 languages by using bidirectional LSTM-CRF network. The code is publicly available[2]. The performance of the bigappy-unicrossy tagging scheme is close to the gappy 1-level tagging scheme and it gets ahead in 8 languages. To conclude, the bigappy-unicrossy tagging scheme is expected to be by far the best scheme on data sets with higher frequency of overlaps and we plan to apply our tagging scheme on such data sets as future work.

[2] https://github.com/deep-bgt/Deep-BGT.

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

References

1. Baldwin, T., Kim, S.N.: Multiword expressions. Handb. Nat. Lang. Process. **2**, 267–292 (2010)
2. Berk, G., Erden, B., Güngör, T.: Deep-BGT at PARSEME shared task 2018: bidirectional LSTM-CRF model for verbal multiword expression identification. In: Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018), pp. 248–253 (2018)
3. Boroş, T., Burtica, R.: GBD-NER at PARSEME shared task 2018: multi-word expression detection using bidirectional long-short-term memory networks and graph based decoding. In: Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018), pp. 254–260 (2018)
4. Constant, M., et al.: Multiword expression processing: a survey. Comput. Linguist. **43**(4), 837–892 (2017)
5. Ehren, R., Lichte, T., Samih, Y.: Mumpitz at PARSEME shared task 2018: a bidirectional LSTM for the identification of verbal multiword expressions. In: Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018), pp. 261–267 (2018)
6. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
7. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
8. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
9. Legrand, J., Collobert, R.: Phrase representations for multiword expressions. In: Proceedings of the 12th Workshop on Multiword Expressions, pp. 67–71. Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/W16-1810, http://aclweb.org/anthology/W16-1810
10. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354 (2016)
11. Ramisch, C., et al.: Edition 1.1 of the PARSEME shared task on automatic identification of verbal multiword expressions. In: Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG 2018). Association for Computational Linguistics, Santa Fe (2018)
12. 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
13. Ramshaw, L., Marcus, M.: Text chunking using transformation-based learning. In: Third Workshop on Very Large Corpora (1995). http://aclweb.org/anthology/W95-0107
14. Ratnaparkhi, A.: Maximum entropy models for natural language ambiguity resolution. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA (1998). aAI9840230

15. Reimers, N., Gurevych, I.: Reporting score distributions makes a difference: performance study of LSTM-networks for sequence tagging. arXiv preprint arXiv:1707.09861 (2017)
16. Sang, E.F.T.K., Veenstra, J.: Representing text chunks. In: Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics, EACL 1999, pp. 173–179. Association for Computational Linguistics, Stroudsburg (1999). https://doi.org/10.3115/977035.977059
17. Schneider, N., Danchik, E., Dyer, C., Smith, N.A.: Discriminative lexical semantic segmentation with gaps: running the MWE gamut. Trans. Assoc. Comput. Linguist. **2**, 193–206 (2014)
18. Taslimipoor, S., Rohanian, O.: SHOMA at parseme shared task on automatic identification of VMWEs: neural multiword expression tagging with high generalisation. arXiv preprint arXiv:1809.03056 (2018)
19. Zampieri, N., Scholivet, M., Ramisch, C., Favre, B.: Veyn at PARSEME shared task 2018: recurrent neural networks for VMWE identification. In: Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018), pp. 290–296 (2018)