Identification of Multiword Expressions Using Recurrent Neural Networks

Manon Scholivet

LIF

Encadrant

Carlos Ramisch

Résumé

Il existe de nombreux défis en traitement automatique des langues (TAL), comme par exemple la traduction automatique de textes, les résumés automatiques ou encore la détection des sentiments dans un texte. Mais pour parvenir à faire exécuter ces tâches correctement par un programme, il convient tout d'abord de retrouver la structure de notre texte. Pour segmenter et retrouver la structure syntaxique d'un texte de façon correcte, il faut se pencher sur le cas particulier des expressions polylexicales. Une expression polylexicale est un groupe de tokens qui doit être considéré comme une unique unité lexicale, et qui possède un certain niveau d'idiosyncrasie d'ordre lexicale, syntaxique, sémantique, pragmatique et/ou statistique. Ce mémoire présente le développement d'un étiqueteur permettant d'identifier efficacement des expressions polylexicales dans un texte à l'aide de techniques d'apprentissage profond. Cet étiqueteur est fondé sur un réseau de neurones récurrent et utilise, entre autres, une pré-initialisation de plongements de mots pré-entraînés (plus couramment appelés *word embeddings*). Nous testerons plusieurs architectures de système ainsi que deux représentations différentes pour les données, pour trouver le système répondant du mieux possible à notre tâche de détection des expressions polylexicales. Nos tests se feront sur différents corpus, en différentes langues et avec plusieurs types d'expressions polylexicales (verbales ou non, contiguës ou non). Nous montrerons que cette approche permet d'obtenir des résultats comparables voire même parfois supérieurs aux méthodes de l'état de l'art obtenus lors d'une campagne d'évaluation internationale récente. Nous ferons également une analyse d'erreurs qui nous permettra de mettre en évidence les avantages et inconvénients de la méthode proposée par rapport à un étiqueteur probabiliste fondé sur une représentation symbolique des données.

Mots-clés : Identification d'expressions polylexicales, étiqueteur, réseaux de neurones, couches récurrentes, plongement de mots (embeddings).

1 Introduction

In today's world in which every place is reachable in a few hours only, one major problem appears when you want to create links with people around the world: language. Translation is a real problem nowadays, and that is the reason why so many people are interested in using computers for automatic translation.

There are many sub-issues within automatic translation, and an especially difficult one is the translation of Multiword Expressions (MWEs). First of all, we can ask: what is a MWE? A MWE is a group of lexical units (tokens) which must be considered as an unique lexical unit, and displaying lexical, syntactic, semantic, pragmatic and/or statistical idiomaticity (Baldwin and Kim, 2010). Examples of MWEs include verbal idioms (*to make ends meet*, *to strike while the iron is hot*), particle verbs (*to give up*, *to throw away*), nominal compounds (*dead end*, *ivory tower*), light-verb constructions (*to make a presentation*, *to take a walk*), complex adverbials (*by the way*, *in fact*, *at stake*), and so on.

So one cannot simply translate an expression taking each part of the expression separately. Translated literally, an expression is often nonsense. You can always try to tell a French speaker "*Il pleut des chiens et des chats aujourd'hui!*" (*It's raining cats and dogs today* in English), but he/she will probably look at you sceptically. But if you tell this person "*Il pleut des cordes aujourd'hui!*" (*It's raining ropes today* literally in English), the conversation will run its course as if nothing had happened.

Unfortunately, MWEs are a really frequent phenomenon, so we cannot just ignore them (Baldwin and Kim, 2010). However, before wondering how to translate our MWEs, the first step will be to detect them. Identifying MWEs could appear as an easy task if one takes a lexicon and looks for each occurrence in the text. But many problems come into play, such as the inflection of each word or the variability of the MWE as a whole. Even worse, some MWEs can be ambiguous, like the French expression *Les carottes sont cuites*. In the case of a MWE, this sentence means *The jig is up*, but if someone is cooking and uses that sentence, the translation will probably literally be *The carrots are cooked*.

As part of an internship last year, we had the opportunity to work on this subject, and developed a tagger based on Conditional Random Fields (CRF) (Lafferty et al., 2001) to try to identify highly ambiguous MWEs without using syntactic trees to train our model (Scholivet and Ramisch, 2017). This was a follow-up of the paper by Nasr et al. (2015), in which a parser was used for the same task. We have showed that, while sequence models such as CRF taggers can adequately model some MWE phenomena (e.g. ADV+*que* conjunctions), other require syntactic trees to be disambiguated (e.g. partitive determiners).

This dissertation reports on the work developed during the Masters's internship which took place this year. The goal of the present work is to develop and evaluate a Recurrent Neural Network (RNN) model to perform the identification of MWEs. Indeed, our first hypothesis is that an RNN is able to generalise better than a CRF, which can generally detect only MWEs that it has already seen once. We also would like to investigate the impact of pre-trained word embeddings (see Section 3) on the performance of the RNN, as well as different architecture and hyper-parameter choices. Furthermore, we also believe that the RNN could learn cross-lingual MWE identification models, but this question will not be dealt with in this dissertation.

In this work, we will present an RNN tagger for MWE identification, and

its performance on different corpora. In Section 2 we will present the state of the art, in Section 3 we will describe our RNN system. We will present our results in Section 4, and finally, we will conclude and explain our future work in Section 5.

2 Related Work

Currently, Google Translate, to cite the most famous automatic translation service, uses statistical machine translation methods (Koehn, 2010). It means that Google Translate does not use grammar rules, but that the system learns a statistical model thanks to a large amount of parallel corpora (using, among others, corpora from the European Parliament and from the United Nations). Nonetheless, even Google does not have parallel corpora for each pair of languages. That is the reason why it often uses English as an interlingua, translating the source language into English, then translating from English into the target language.

However, in November 2016, Google announced that it began to switch to the Google Neural Machine Translation system, a deep learning system based on a recurrent neural networks, supposedly more fluent and accurate than previous models (Johnson et al., 2016). Since April 2017, the system has been able to support around twenty languages, and Google has been adding languages gradually.

This is an example of a situation in which deep learning revolutionises the domain of Natural Language Processing (NLP). Since we used to use statistical models for the MWE identification task, it seems to be a natural evolution to start using neural network models, as it was the case for for the machine translation task.

More specifically, currently the task of identifying MWEs is usually carried out using two main kinds of approaches : sequence taggers (as in Constant and Sigogne (2011)) and parsers (as in Nasr et al. (2015)).

The MWE identification can be seen as a tagging problem. Sequence taggers use different stochastic models such as Conditional Random Fields (CRFs) (Constant and Sigogne, 2011)), structured perceptron (Schneider et al., 2014; Riedl and Biemann, 2016) or structured support-vector machines. Data are represented with a BIO representation (see Section 3.1), and many features are used, including local context information or sometimes external resources such as MWE lexicons.

Parser-based methods build a parse tree for the sentence to represent its syntactic structure. During the construction of the parse tree, parsers detect MWEs thanks to special syntactic nodes and/or arcs.. These methods give quite good results, specially for discontinuous expressions such as *to take something into account* (Nasr et al., 2015; Constant and Nivre, 2016; Vincze, 2012). In sum, while sequence taggers are usually easier to train and appropriate to model contiguous expressions, parsers can take into account relations between words and features which are far from one another while building the parse tree.

In a previous paper, we have proposed an adaptation of a standard CRF to identify ambiguous and contiguous MWEs, supposing that sophisticated parsers are not essential to deal with these expressions (Scholivet and Ramisch, 2017). At that occasion, we decided to work with CRFs instead of parsers, since sequence models do not require treebanks to be trained.

Using machine learning for MWE identification is also nowadays possible thanks to the availability of corpora annotated with MWEs to train these models, released in recent shared tasks. We will focus on the DiMSUM shared task[1] and the PARSEME shared task[2]. Those two tasks provide us corpora annotated with MWEs. DiMSUM was a shared task whose aim was to predict the segmentation into minimal semantic units, and labelling of some of those units with semantic classes known as supersenses (Schneider et al., 2016). The supersense aspect was ignored in this work, as we are interested only in the segmentation into minimal semantic units, that is, words and MWEs. This corpus is in English and contains 3,812 sentences for 3,483 annotated MWEs.

The PARSEME shared task is also particularly interesting since, for that task, a European research network created a universal terminology and annotation guidelines for 18 languages (Savary et al., 2017). Thus, we have 18 corpora annotated with verbal MWEs (VMWEs) in different languages. Each VMWEs is additionally tagged with its VMWE category (5 categories, explained in Section 3.1). Two participants of this shared task submitted system quite similar to those we propose. The first one, ADAPT (Maldonado et al., 2017) is a CRF, and MUMULS (Klyueva et al., 2017) using a bidirectional recurrent neural network. The ADAPT system obtained good results in the shared task, since MUMULS had less good results, more especially in the languages we are interested in (French, Brazilian Portuguese and Romanian). That is the reason why we do not consider this system in Section 4.1, where we will compare our system to other ones.

3 RNN-Based MWE tagger

One of the goals of our work is to develop a tagger for MWE identification based on an RNN. Therefore, we first describe the data encoding used (Sec. 3.1), then we briefly survey the fundamentals of neural networks (Sec. 3.2) before presenting the architecture of our system (Sec. 3.3), and finally we describe the experimental setup in which the system is evaluated (Sec. 3.4).

3.1 Data representation

Parseme-tags representation To begin with, we will explain how our data are represented. First, we will be interested in the data from the PARSEME shared task. The provided corpora contain sentences, and on top of each sentence, human annotators have provided information about VMWEs present in that sentence. For each language, we have a train and a test corpus in the parseme-tsv format.

Each sentence starts with two comment lines, the first one containing the sentence ID, and the raw text on the next one. Then, we simply have a kind of CoNLL format, with one token per line, and an empty line to separate each sentence. Every line starts with the ID of the token followed by the wordform (all separated by tabulations), a column indicating whether the current token is adjacent to the next one (nsp), and the information of whether the token belongs to a VMWE. This information is represented as a code formed by the number

[1] `http://dimsum16.github.io`

[2] `http://multiword.sf.net/sharedtask2017`

```
# sentid: fr-ud-train_12947
# sentence-text: Les demi-finales et la finale se jouent à Copenhague.
1       Les     _       _
2       demi-finales    _       1:LVC
3       et      _       _
4       la      _       _
5       finale  _       2:LVC
6       se      _       _
7       jouent  _       1;2
8       à       _       _
9       Copenhague      nsp     _
10      .       _       _
```

Figure 1: The parseme-tsv format on a French sentence containing two overlapping LVC annotations: *jouent finale* and *jouent demi-finales*, sharing the token *jouent*.

of the VMWE in the sentence, and, if it is the initial token of the VMWE, we add the category of the latter. If the token is part of many VMWEs, multiple codes are separated with a semicolon. There are 5 different possible categories for the VMWEs, as described in Savary et al. (2017):

- idioms (*let the cat out of the bag*), abbreviated as ID,
- light-verb constructions (*make a decision*), abbreviated as LVC,
- verb-particle constructions (*give up*), abbreviated as VPC,
- inherently reflexive verbs (*se suicider 'to suicide'* in French), abbreviated as IReflV, and
- other expressions, abbreviated as OTH.

An example of this format is shown on Figure 1. Here, we can see that the second token of the sentence is the initial token of an LVC MWE, whose second token is number 7. It also illustrated partial overlap as this last token is also in a second LVC MWE (starting in token number 5).

Most of the time, we also have an aligned CoNLL-U file[3] for the train and test, which provides information such as the lemma or the part of speech (POS) of each token.[4]

BIO representation In some tests, we kept the representation of the VMWEs provided by the parseme-tsv file. But for all other tests, we used a BIO representation for the tags (Ramshaw and Marcus, 1995). In a BIO representation, every token has the information B, I or O. If the tag is B, it means the token is the Beginning of an MWE. In the I case, it means the token is Inside an MWE. Finally, if the token's tag is O, it means the token is Outside the expression, and does not belong to an MWE.

We had to slightly modify this representation which was thought for contiguous MWEs. We simply add the number of the MWE to the tag, so that we

[3] `http://universaldependencies.org/format.html`

[4] The *lemma* is the canonical form of a word, for instance, the lemma of *eating* is *eat*. The part of speech is the morphosyntactic category of a word (verb, noun, pronoun, etc.), for instance, the POS of *eating* is VERB.

i:	1	2	3	4	5	6	7	8	9
w_i:	*Les*	*demi-finales*	*et*	*la*	*finale*	*se*	*jouent*	*à*	*Copenhague*
First MWE:	O	B1	O	O	O	O	I1	O	O
Second MWE:	O	O	O	O	B1	O	I1	O	O

Figure 2: Example of BIO tagging, where the sentence had to be duplicated since there is an overlap. As in Figure 1, the MWEs are *jouent finale* and *jouent demi-finales*.

can identify to which MWE the token belongs. In this case, we consider that a token can only belong to a single MWE. For the training corpus, we duplicate each sentence containing overlaps as many times as the number of overlapping MWEs. An example of this BIO representation is shown on Figure 2.

3.2 Neural Networks

Since a large part of this work was done with neural network, it is important to understand well the idea behind this system.

To begin with, a neural network is named in this way because the base idea is to imitate the behaviour of our brain, more especially our neurons. We would like to have many neurons which each "learns" little things.

A neural network is composed of one or several layers of artificial neurons (Lecun et al., 1998; Goldberg, 2015). Each neuron has a set of weights, which are updated during the training process. Neurons are organised in layers, and each layer is connected to one another. Here, we will focus on multilayer perceptrons, in other words, neural networks with more than one layer, and whose weights can vary.

To train a neural network, we have to give it examples with the correct tags we want to predict. Thanks to the principle of backpropagation, if the system fails to predict the right answer, it will correct its neurons' weights in order to try to minimize the loss function. The loss function is the function which calculates the difference between the predicted tag and the correct one.

During training, the system can adjust the neurons' weights after each sample, or after each *batch of samples*. In this case, we have to define the size of batch : the bigger it is, the faster the training will be, but it might be less accurate. Moreover, learning is done while scanning through the training corpus several times. The number of times it learns on the training corpus is called the *number of epochs*.

In our problem of detecting MWEs, the history of the previous inputs and tags is important. That is the reason why we decided to use recurrent neural networks (RNN), using GRU and LSTM modules (Cho et al., 2014; Chung et al., 2014; Huang et al., 2015). These layers can take into account the previous events to make their decision. But sometimes, we even need the next inputs to decide the current tag. Therefore, we used a bidirectional layer wrapper (Graves and Schmidhuber, 2005), which enables the system to read the sentence in the both directions.

Finally, we have to talk about vectorial word representation, also called word embeddings (Mikolov et al., 2013b,a). As input for our neural network, we will give words, lemmas or POS. But a neural network takes as argument only real numbers. So we will transform our inputs into vectorial representations. These

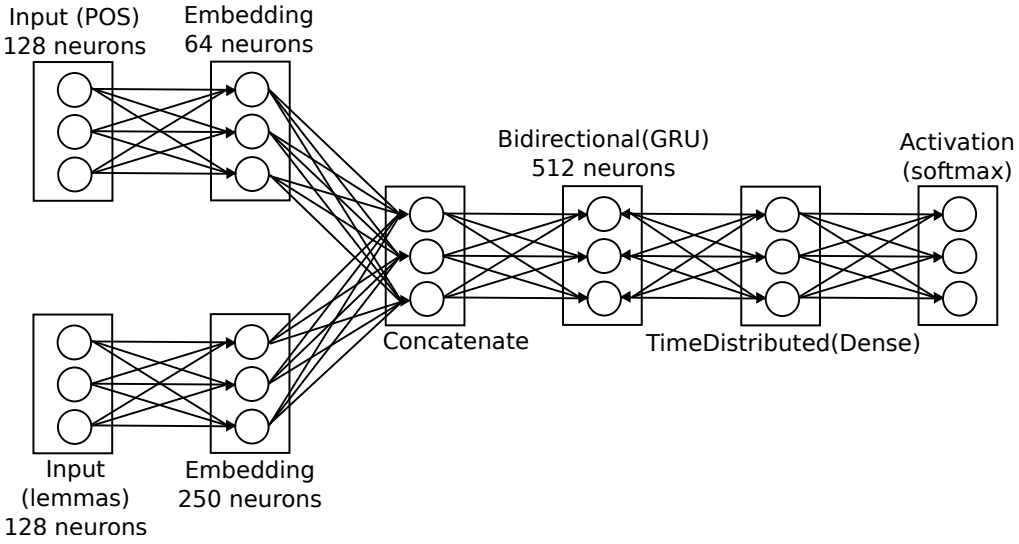

Figure 3: Architecture of our baseline RNN, with two inputs, lemmas and POS and initialising our lemma's input with pre-trained embedding of size 250

vectors can be pre-trained to represent the meaning of the words depending on their contexts, and so, the closer the words are, the closer their vectors will be too.

3.3 MWE tagger architecture

In this Section, we will explain the architecture of our baseline RNN system, and then, we will specify the extensions we will add.

3.3.1 Baseline

Now, we will examine the architecture of our RNN. Figure 3 shows the architecture of the system which was used as our baseline.

First, our network takes 2 different inputs: the first one corresponds to the lemma of each token, and the second to its POS. We considered that each sentence contains 128 tokens. If our sentence is smaller, we pad it with 0 until we reach 128 tokens in the sentence. However, if it is longer, we simply cut the sentence. By default, there are 64 neurons in embedding layers, if we do not use pre-trained embeddings as input to the system (more information in Sec. 3.3.2 about the pre-trained embeddings). The embeddings layers transform inputs, which are positive integers (codes of tokens' features) into dense vectors of fixed size. The embeddings layers are the first trainable layers in a model.

Once we have created an embedding layer for both inputs, we concatenate them, because the recurrent layer can take only one input. Next, we create the recurrent layer, which is bidirectional. This layer contains 512 neurons, and uses the option return_sequences, which enables the layer, with the help of the next layer (TimeDistributed), to predict one tag per token, and not one per sentence. The TimeDistributed wrapper allows us to apply a layer (here, a dense layer, which is a classic layer of neurons) to every temporal slice of an

input. Finally, the activation layer used a softmax function to transform results into probabilities for each possible token. We will choose the tags with the best probability.

For the loss function, we used the sparse_categorical_crossentropy, and we used Nadam as optimizer. We decided to carry out our tests with 10 epochs and batch with size 128. We used the Python library Keras[5] to implement our system, using tensorflow[6] as backend. We trained our model thanks to a cluster with GPU nodes.

3.3.2 Extensions

We performed some extra experiments which made us modify the structure of our base architecture. First of all, we tried to initialise the embedding layer with pre-trained embeddings. Consequently, we adjusted the number of neurons on those layers to the dimension of pre-trained vectors (in our case, 250 or 300 depending on the language). We also tried to replace the GRU layer with a LSTM layer, and then we tried to stack up recurrent layers: we test with 1, 2 and 3 layers. Each bidirectional recurrent layer is followed by a TimeDistributed layer, always in order to predict a tag for each token. More explanations about these extensions will be found in Section 4.

3.4 Experimental Setup

Corpora In order to evaluate our systems, we decided to test it on 5 different corpora: PARSEME-FR, PARSEME-PT, PARSEME-RO, the French treebank and the English corpus from DiMSUM shared task. Each of these corpora have their own interesting characteristics.

Corpora from the PARSEME shared task have the particularity to be annotated with verbal expressions only, which are not necessarily contiguous. Moreover, common annotation guideline have been written for all the languages of the shared task, and used for the manual annotations, which provide a quite good homogeneity in data. Most of those languages are accompanied by a CoNLL-U file, which provides information for each token, such as lemma or POS. We decide to focus on 3 corpora from the PARSEME shared task: French (FR), Brazilian Portuguese (PT) and Romanian (RO). We chose those languages because we could make an error analysis with the French and Brazilian and find some interesting or ambiguous cases, and we have some pre-trained embeddings. The Romanian corpus was the biggest one of the shared task (45,469 sentences for 4,040 VMWEs in training), but we did not have pre-trained embeddings. We used it to test the importance of the size of the training corpus.

We used a modified version of the French Treebank from SPMRL shared task (Seddah et al., 2013), which does not annotate only verbal MWEs. However, in this corpus, all the MWEs are contiguous. The corpus contains 14,759 sentences for 23,658 MWEs in training.

Finally, we used the DiMSUM corpus (Schneider et al., 2016) (3,812 sentences for 3,483 MWEs), to test our results on a task with little data, containing all types of MWEs, including non-contiguous ones. Furthermore, we have pre-trained embeddings for English. The task associated with DiMSUM is quite

[5] `https://keras.io/`

[6] `https://www.tensorflow.org/`

hard, the best system on this task obtaining an F-measure of 57.24% on MWEs identification.

Evaluation In order to have results comparable with those from the PARSEME shared task, we decide to use their evaluation script, which provides us with two different evaluations: the one based on MWEs, and the one based on tokens.

Both of them correspond to an F-measure. The MWE-based measure verifies that all the tokens on the MWE have been found, whereas the TOKEN-based measure will be calculated relying on each token.

It is important to notice that there is a notion of randomness in neural networks. Variabilities of our results led us to launch each of our test around 20 times. The results we present are the average on these tests for each measure (precision, recall and F-measure). That is the reason why we decided to highlight the standard deviation for each test, because some of them are more stable than others, and we believe that this variation must be taken into account.

4 Evaluation

We will now describe the results our systems obtained. First we will detail results while focusing on 4 main questions, then, we will carry out an error analysis to better understand our results.

4.1 Results

To evaluate our system, we will focus on the 4 main hypotheses that we would like to verify:

- What is the influence on performance of word embedding pre-initialisation?
- What is the influence on performance of the architecture (e.g. number and type of layers) and of hyper-parameter values (e.g. size of layers, number of epochs) for different configurations?
- What is the influence on performance of the data representation?
- How does the RNN model compare to other models?

We will present our results on graphics summarizing different experiments, answering the questions above. In each graphic we show two measures for every experiment: the MWE-based measure, and the TOKEN-based measure. In Appendix A, we show a table giving the precision and recall for each of these measures for every experiment. For each measure, you will find the standard deviation as error bars, since our results are an average, as explained in Section 3.4.

Each experiment code shown in the X axis of a graph corresponds to a corpus and an architecture. For example, when we treat the French Treebank with the basic configuration, the corresponding experiment code is *FTB_B*. Thus, FTB corresponds to the French Treebank, Dim to DiMSUM, PBIO to PARSEME-FR with BIO tags, P to PARSEME-FR with the original tags, PT to PARSEME-PT and finally, RO corresponds to PARSEME-RO. For the second part of the code, which corresponds to the model configuration, B simply refers to the basic

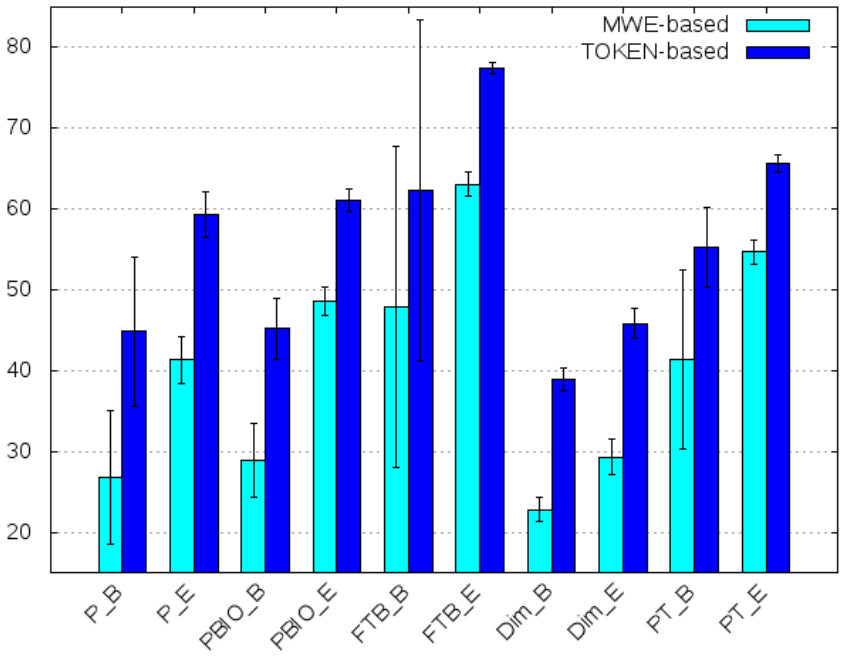

Figure 4: Influence of pre-initialising lemma's embeddings

configuration described in Section 3.3, E to the tests when we add pre-trained embeddings, 2GRU when we decide to stack up two layers of GRU, 3GRU for three layers of GRU, LSTM is when we test with a LSTM layer instead of a GRU layer, WF is when we decide to add in inputs the wordforms (not having pre-trained embeddings for them), and diffParam is when we test a completely different configuration (number of neurons on hidden layers, number of epochs, etc...) and eventually, 20epochs means we test 20 epochs instead of 10. Details about these configurations will be given in the corresponding experiments.

Pre-initialisation of embeddings To begin with, we think it is important to remind that embeddings are used to represent the meaning of each word thanks to a simple vector which represents its co-occurrence profile. We have two ways to estimate the values of this vector: either we initialise them randomly and then refine them directly during the training of our recurrent neural network, or we can pre-train our vectors on a large quantity of plain text, and then, we continue to train them while we train our RNN.

With the second method, a problem may arise because some words of our training and test corpus have never been seen in the corpus used to pre-initialise the embeddings. This is quite rare, since we most words do occur in a huge corpus of plain texts. However, some words will never have been seen, so we initialise them randomly.

In order to pre-initialise the embedding layer of our network, we retrieve existing lemma embeddings, in French, English and Portuguese. In French and Portuguese, our vectors are in 250 dimensions, trained using word2vec on web

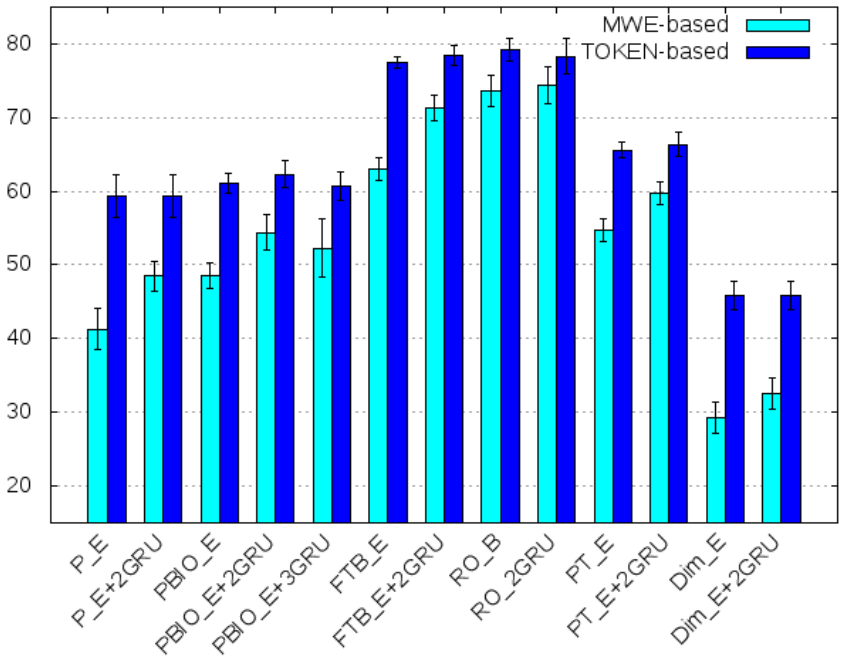

Figure 5: Influence of the number of GRU layers

corpora of around 2 billion words for each language, as described in Cordeiro et al. (2016). In English, we use the embeddings available on GloVe website[7] (Pennington et al., 2014) trained on Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, in 300 dimensions).

Results of experiments that evaluate the influence of the pre-initialisation of our lemma embeddings can be found on Figure 4. We clearly observe that in all experiments, the pre-initialisation of the embeddings considerably improves the results. Even better, the standard deviation sharply decreased, particularly in the case of the French Treebank. This reduction of the standard deviation reveals much about the importance of initialising our lemma's input with pre-trained embedding to make our results converge much faster than with no pre-initialisation.

For the following experiments, we take as a reference the results with embedding pre-initialisation, in order to be more coherent with the importance of these inputs.

Architecture Here, we focused on the importance of the architecture of our system. We will start to linger over the influence of the number of GRU layers (Figure 5). The first interesting thing to notice is that, when we put a second GRU layer, the results increase overall, in all the different experiments. More specifically, we notice that it is the MWE-based measure which increases the most, whereas the TOKEN-based measure remains quite stable, although it tends to increase a little. The explanation would probably be that the TOKEN-

[7] `https://nlp.stanford.edu/projects/glove/`

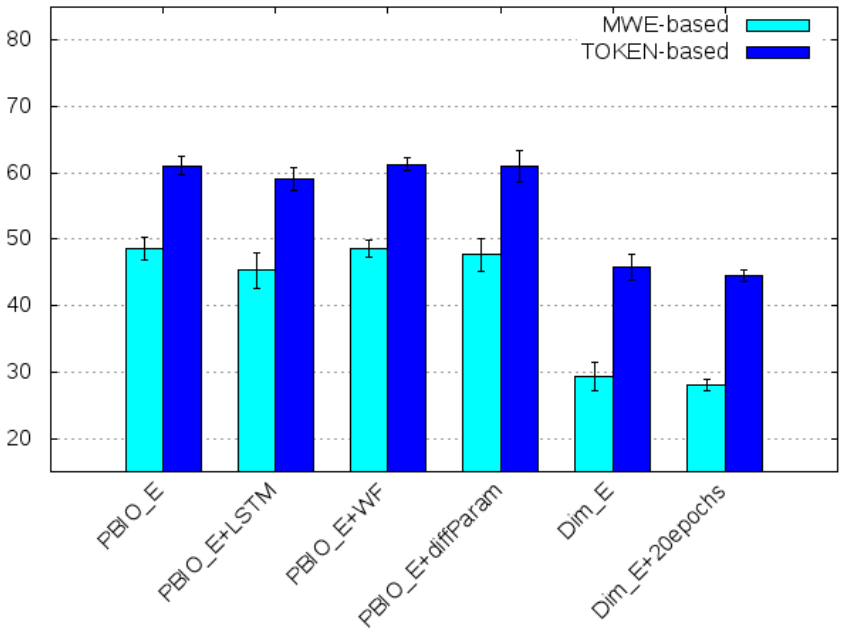

Figure 6: Influence of the architecture

based measure increases even when we only find a part of a MWE. With a single GRU layer, we identified some important characteristics of the structure of the MWE. With the second GRU layer, we identified the entire MWE in a better way. For example, in PARSEME-FR with BIO, in the sentence : *"... accomplir de multiples tâches et démarches..."*, the word *accomplir* is a part of two MWEs : *accomplir tâche* and *accomplir démarche*. With a single GRU layer, our system manages to find that *accomplir* is part of an MWE, but does not find anything else. With two GRU layers, our system finds the complete MWE *accomplir tâche*! However, the BIO tag system is not expected to predict several MWE tags on only one token. That is the reason why the VMWE *accomplir démarche* cannot be found at the same time as the first VMWE *accomplir tâche*. Though, with the parseme-tag system, our RNN could have predicted two MWEs on a single token, but since this case is rare enough, even with this representation, the system did not predict both VMWEs.

As a complementary experiment, we have also tested the use of three GRU layers for the PARSEME-FR corpus with BIO tags (configuration PBIO_E+3GRU). Nevertheless, we can notice that the test with three GRU layers is negative, since it reduces performances compared to two GRU layers. Further experiments would be required to confirm this tendency for other languages and configurations.

Now, we will discuss the results present in Figure 6. This time, we present results showing the influence of some hyper-parameters, or their values in different configurations, in order to make some sanity checks. First, we compare our reference, PARSEME-FR with BIO tags and pre-trained embeddings, with

the same RNN, but with a LSTM layer instead of a GRU one. Besides being slower to train, (108 seconds for one epoch with the LSTM against 88 seconds with GRU) the GRU architecture obtains better results. For the second test, we verified if adding wordforms as an additional input could be useful. We noticed that the results are not significantly different, but standard deviation slightly decreased, so we can suppose that wordforms help the training to converge faster, but they do not help increase the performances.

Then, we carried out tests with different hyper-parameter values, automatically inferred on a development set (diffParam). We tested values for batch size of 16, 32 or 64, number of neurons in the hidden layers from 128 or 250, number of epochs between 1 and 20, and embedding dimensions from 6, 8 or 10. The dimension of embeddings is small because we consider we used pre-trained embeddings, so the default value is applied only to POS, which have only 17 possible values. The chosen values are: number of neurons in the hidden layer = 250, batch size = 16, number of neurons in the recurrent layer = 128,embedding dimensions = 8, and number of training epochs = 4. As expected, the standard deviation increased since we only had 4 epochs, so it converged slower. However, we can also notice the results are quite comparable : around 1% difference only between the two TOKEN-based measures. Our system does not need many epochs to learn a basic model, since with only 4, we obtained those results.

Our last test was realised on DiMSUM corpus, and tested again the influence of the number of epochs. Once again, we observed that the results did not change significantly, but the standard deviation clearly decreased. It is quite certain that the more epochs we have, the more stable our results are.

Data representation This time, we will concentrate on the difference of results between two identical models, but different representation of the tags, show in Figure 7. The first representation is the BIO system, and the second system keeps the tags' system from the PARSEME task (for more information, see section 3.1). Notice that, in the configuration where the system is learned on the original data representation, the category of the MWE is also predicted (ID, LVC, IReflV, etc), which provides a richer output with respect to the BIO tags.

The first thing these tests allow us to say is that the TOKEN-based measure does not vary much between the two representations, even if the BIO system has slightly better results. However, when we concentrate on MWE-based measures, the BIO representation systematically obtains better results. Furthermore, we observe that the standard deviation is lower in the case of BIO system, and so, the RNN converges faster with BIO tags, probably because there are less different tags to predict in the case of BIO.

Surprisingly, the use of the original data representation provides reasonably good results, only slightly inferior to BIO. We discovered this because we were looking for a new representation to replace the BIO scheme, and we tried the original tags by chance. We never thought we could have results so close to the BIO system, and if we had had more time, we would have liked to pursue these experiments to find a representation that would be even more adequate to this problem.

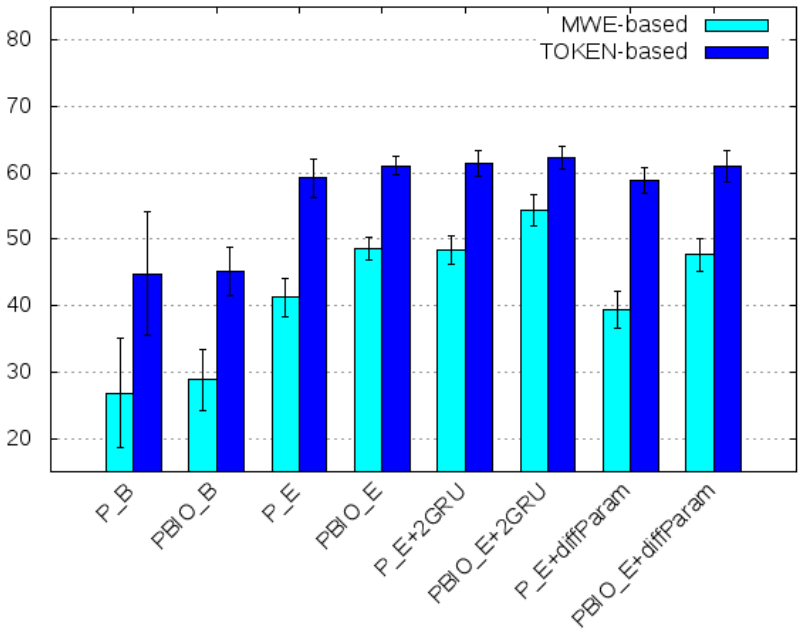

Figure 7: Influence of the data representation

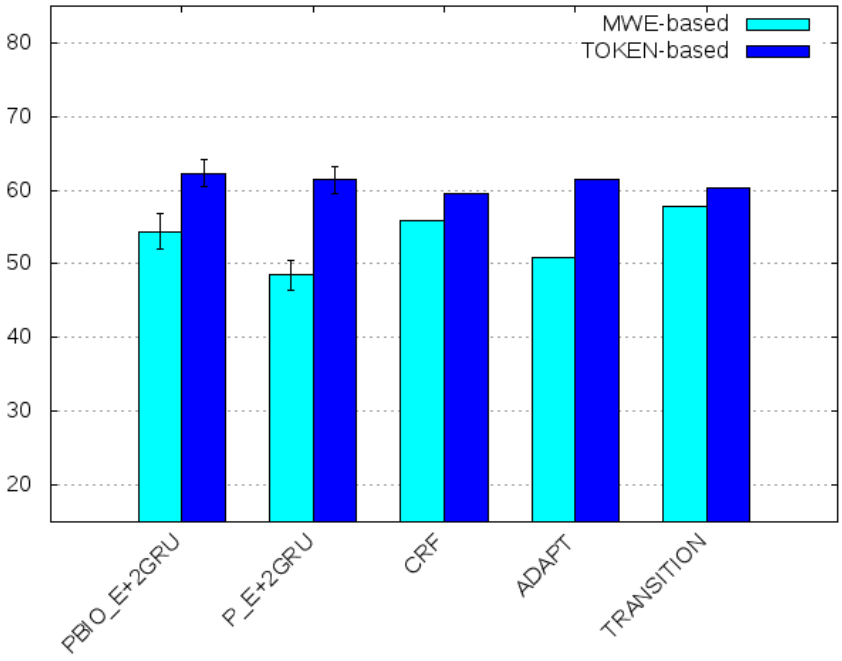

Figure 8: Comparison between models

Comparison with other models Finally, we will compare our systems to other ones (Figure 8). We will focus on our best system which is the one with pre-trained embeddings initialisation and two GRU layers. We compare our best system with a sequence tagging model based on conditional random fields (CRF) developed earlier (Scholivet and Ramisch, 2017).

There is no randomness in the CRF system, so there is no error bar for the standard deviation. We also report the results of the two best systems from the PARSEME-FR shared-task, ADAPT (Maldonado et al., 2017), a system based on a CRF too, and the TRANSITION system (Al Saied et al., 2017), based on a transition-based parsing system. We did not re-implement or re-run the ADAPT and TRANSITION systems, but instead we use the results published on the PARSEME shared task website.[8]

We can notice that, for the TOKEN-based measure, our BIO system obtains the best results: 62.33%, for 61.43% for the system of PARSEME tags, 59.61% for our CRF, ADAPT obtains 61.52% and TRANSITION 60.28%. However, in the case of MWE-based measure, we can note the TRANSITION system obtained the best results. Despite that, our system has rather good results, even better than the ADAPT system. It is interesting to see that the MWE-based and the TOKEN-based measures are really close in the TRANSITION system and in the CRF system. Nevertheless, the difference between the TOKEN-based and the MWE-based measure in our best system is still large, even with the second GRU layer. It would be very interesting to try to reduce this difference in results, as future work.

4.2 Error Analysis

Now, we will focus on concrete examples our RNN succeeds in predicting, or not. The goal of this error analysis is to understand what kinds of errors are made, so that we can focus on them for future work suggestions.

Never Previously Seen MWEs Before starting, it is interesting to notice that in Maldonado et al. (2017), Table 1 gives indication about *Previously Seen (PS) MWEs* in the training set. We notice that in the French PARSEME corpus, the percentage of MWEs that were in the training corpus is only 28%. Just after Slovene, with only 5.51%, French is the language with the least PS MWEs. Despite this, we obtain quite good results in French, indicating that the model is able to generalise well and learn a model which is able to identify MWEs which were never seen in the training data.

First, we wondered if our CRF or our RNN could predict some MWEs they never encountered in the training corpus. We will focus on the case of Never Previously Seen (NPS) MWEs in the training corpus.

In the test corpus, we find many NPS VMWEs. For example, we find in the sentence : *"La musique n'adoucit pas toujours les moeurs."* the VMWE *La musique adoucit les moeurs*, which is an idiomatic expression. None of our systems (CRF, PBIO_E+2GRU and P_E+2GRU) was able to predict this NPS VMWE.

[8] http://multiword.sourceforge.net/PHITE.php?sitesig=CONF&page=CONF_05_MWE_2017___lb__EACL__rb__&subpage=CONF_50_Shared_Task_Results

However, in the sentence *"... M. Soyer a fait l'historique de l'école maternelle ..."*, we find the NPS VMWE *faire historique*. In this case, the CRF was not able to predict it, but the two RNN system did. And the P_E+2GRU system even managed to predict it was a VMWE of the type LVC, which in this case is the correct category to predict.

Another interesting example is this sentence in the test corpus : *"... chacun ne rêvait que de remettre la main à la pâte avec Sylvie et Daniel."* with the VMWE *remettre la main à la pâte*. This VMWE as it is had never been seen in training corpora. And yet, the system PBIO_E+2GRU manage to identify the part of the expression *remettre la main à*, while de P_E+2GRU managed to predict *remettre la main* with the right category ID. Meanwhile, our CRF system did not identify anything, once again. Nonetheless, it is really interesting to observe what RNN systems manages to predict. In the training test, we could find the ID VMWEs *prendre en main* and *mettre (la) dernière main*. We can suppose the RNN systems generalise enough from these examples to predict part of the NPS VMWE *remettre la main à la pâte*.

CRF vs RNN We previously saw that the CRF often has difficulties in predicting NPS VMWEs. We will now focus on some hard cases that were present in training corpus, but are hard to identify.

Once again, let us focus on the following example : in the sentence *"Une réflexion commune est menée avec les enseignants ..."*, we have the the VMWE *mener réflexion*. In the training corpus, we only see this VMWE in the sentence : *"Nous n'avons mené aucune réflexion sur le sujet"*. In this case, the CRF was not able to identify the VMWE, probably because it does not appear in the same way (there is an inversion in the order of the expression due to passive voice) as in the training corpus. The system P_E+2GRU also had some difficulties identifying it. Sometimes, it found only the verb (*mener*), sometimes absolutely nothing, and sometimes it found the entire expression, and its category (LVC). Nonetheless, the PBIO_E+2GRU did not have any problem to predict this VMWE each time.

On the other hand, in the sentence *"... elle descendait seule faire ses courses en centre ville, et prenait alors plaisir à faire un brin de causette avec ses copines ..."*, we have 3 VMWEs : *faire courses*, *prendre plaisir* and *faire causette*. The first two were in the training set, but not the last one. All our systems managed to predict *prendre plaisir*, none predicted correctly *faire causette*, and only the CRF correctly identified *faire courses*. The P_E+2GRU predicted only the verb *faire* (with the correct LVC category), and the PBIO_E+2GRU system identified the whole *faire ses courses*, wrongly including an extra word *ses*. In short, we observe that sometimes, the CRF system outperformed the RNN, because the latter sometimes generalised too much.

5 Conclusions and Future Work

We presented a recurrent neural network, with some derivatives, able to identify MWEs. We evaluated these systems on different corpora in different languages, with different types of MWEs (VMWEs, non-contiguous MWEs, etc) and also different representations of the tags. We compared our systems to those from the PARSEME shared task and to a CRF we previously developed for the same

task. We concluded that our best systems did well in this tasks compared to all other best systems.

Even if we are pretty satisfied with our results, we still have many improvements we would like to test. First, we would like to make a parameter optimisation, since we did not had enough time to explore the whole hyper-parameter space, specially trying to reduce overfitting.

As we previously said, we are not fully satisfied with the BIO representation for the tags, and we would like to spend some time on the parseme-tags system to improve it, or to think about a completely different system, such as prediction the POS tags and MWEs simultaneously.

Then, we think it could be interesting to try to shed some light on the importance of the tags containing MWE information. An option in Keras allows us to give a weight matrix for each tag in the training, in this way explaining to the system that this information is more important. At a moment we tried to do this, but we empirically tested different weights, and one would have had to be more systematic in these choices.

Furthermore, we are currently working on a thesis project where the goals would be to develop deeply multilingual models for analysis tasks in natural language processing, even when we do not have training corpora (*zero-shot learning*, see (Johnson et al., 2016)) and to find a representation that is independent from the language for the lexical units (words and MWEs) to use in these models. One of the tasks in which we intend to evaluate such model is MWE identification. Therefore, we could train a multilingual MWE identification system able to exploit the similarities between languages in order to obtain better results.

References

Al Saied, H., Candito, M., and Constant, M. (2017). The atilf-llf system for parseme shared task: a transition-based verbal multiword expression tagger. *MWE 2017*, page 127.

Baldwin, T. and Kim, S. N. (2010). Multiword expressions. In Indurkhya, N. and Damerau, F. J., editors, *Handbook of Natural Language Processing*, pages 267–292. CRC Press, Taylor and Francis Group, Boca Raton, FL, USA, 2 edition.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Constant, M. and Nivre, J. (2016). A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 161–171, Berlin, Germany. Association for Computational Linguistics.

Constant, M. and Sigogne, A. (2011). MWU-aware part-of-speech tagging with a CRF model and lexical resources. In Kordoni, V., Ramisch, C., and Villav-

icencio, A., editors, *Proc. of the ALC Workshop on MWEs: from Parsing and Generation to the Real World (MWE 2011)*, pages 49–56, Portland, OR, USA. ACL.

Cordeiro, S., Ramisch, C., Idiart, M., and Villavicencio, A. (2016). Predicting the compositionality of nominal compounds: Giving word embeddings a hard time. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1986–1997, Berlin, Germany. Association for Computational Linguistics.

Goldberg, Y. (2015). A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F. B., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2016). Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558.

Klyueva, N., Doucet, A., and Straka, M. (2017). Neural networks for multi-word expression detection. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Workshop on Multiword Expressions*, page 6, Valencia, Spain.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

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

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Maldonado, A., Han, L., Moreau, E., Alsulaimani, A., Chowdhury, K. D., Vogel, C., and Liu, Q. (2017). Detection of Verbal Multi-Word Expressions via Conditional Random Fields with Syntactic Dependency Features and Semantic Re-Ranking. In Stella Markantonatou, Carlos Ramisch, A. S. and Vincze, V., editors, *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 114–120, Valencia, Spain. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Nasr, A., Ramisch, C., Deulofeu, J., and Valli, A. (2015). Joint dependency parsing and multiword expression tokenization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1116–1126, Beijing, China. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ramshaw, L. and Marcus, M. (1995). *Third Workshop on Very Large Corpora*, chapter Text Chunking using Transformation-Based Learning.

Riedl, M. and Biemann, C. (2016). Impact of mwe resources on multiword recognition. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 107–111, Berlin, Germany. Association for Computational Linguistics.

Savary, A., Ramisch, C., Cordeiro, S. R., Sangati, F., Vincze, V., Qasemizadeh, B., Candito, M., Cap, F., Giouli, V., Stoyanova, I., and Doucet, A. (2017). The PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions. In *MWE 2017 - Proceedings of the 13th Workshop on Multiword Expressions* , pages 31 – 47, Valencia, Spain.

Schneider, N., Danchik, E., Dyer, C., and Smith, A. N. (2014). Discriminative lexical semantic segmentation with gaps: Running the mwe gamut. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*, pages 193–206.

Schneider, N., Hovy, D., Johannsen, A., and Carpuat, M. (2016). SemEval-2016 Task 10: Detecting Minimal Semantic Units and their Meanings (DiMSUM). In *Proc. of SemEval*, San Diego, California, USA.

Scholivet, M. and Ramisch, C. (2017). Identification of ambiguous multiword expressions using sequence models and lexical resources. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 167–175, Valencia, Spain. ACL.

Seddah, D., Tsarfaty, R., Kübler, S., Candito, M., Choi, J., Farkas, R., Foster, J., Goenaga, I., Gojenola, K., Goldberg, Y., et al. (2013). Overview of the spmrl 2013 shared task: cross-framework evaluation of parsing morphologically rich languages. Association for Computational Linguistics.

Vincze, V. (2012). Light verb constructions in the szegedparalellfx english-hungarian parallel corpus. In *LREC*, pages 2381–2388.

A Appendix: detailed results

System	Average per-MWE results			Average per-token results		
	Precision	Recall	F-measure	Precision	Recall	F-mesure
PBIO_B	22.92	40.20	28.85	37.09	**58.98**	45.17
PBIO_E	57.83	41.99	48.61	**78.35**	50.09	61.05
PBIO_E+diffParam	55.68	42.12	47.71	77.50	50.79	61.04
PBIO_E+2GRU	**65.38**	**46.73**	**54.43**	76.91	52.52	**62.33**
PBIO_E+3GRU	64.86	44.19	52.26	76.36	50.94	60.68
PBIO_E+WF	56.65	42.64	48.58	77.05	51.02	61.29
PBIO_E+LSTM	53.33	39.44	45.30	75.65	48.52	59.03
P_B	34.66	22.16	26.84	67.65	34.64	44.83
P_E	48.82	36.03	41.34	**75.88**	48.96	59.29
P_E+2GRU	**55.56**	**43.18**	**48.46**	73.49	**52.85**	**61.43**
P_E+diffParam	45.50	34.94	39.42	74.90	48.79	58.88
FTB_B	45.38	51.64	47.84	65.53	61.88	62.31
FTB_E	59.16	67.79	63.09	78.26	76.91	77.44
FTB_E+2GRU	**70.57**	**72.56**	**71.34**	79.17	78.28	78.44
FTB_E+diffParam	61.28	69.82	65.20	**80.20**	**78.32**	**79.14**
Dim_B	26.09	2.58	22.82	47.06	33.75	38.88
Dim_E	32.82	27.03	29.30	54.45	**40.83**	29.30
Dim_E+2GRU	**38.69**	**28.69**	**32.50**	**56.78**	38.94	**45.44**
Dim_E+20epochs	29.93	26.37	28.02	50.57	39.92	44.57
PT_B	44.39	38.99	41.36	64.45	48.77	55.27
PT_E	58.32	51.59	54.68	**74.91**	58.50	65.59
PT_E+2GRU	**63.78**	**56.52**	**59.71**	73.07	**61.24**	**66.39**
RO_B	73.43	74.11	73.65	**81.36**	**77.55**	**79.28**
RO_2GRU	**74.61**	**74.53**	**74.39**	79.88	77.11	78.27

Table 1: Results on all configuration

System	SD per-MWE results			SD per-token results		
	Precision	Recall	F-measure	Precision	Recall	F-mesure
PBIO_B	4.96	2.59	4.61	5.31	3.05	3.72
PBIO_E	**2.30**	2.24	1.77	**1.86**	**2.37**	1.37
PBIO_E+diffParam	4.16	4.12	2.47	4.66	4.77	2.43
PBIO_E+2GRU	3.27	2.87	2.39	3.38	2.59	1.78
PBIO_E+3GRU	8.45	2.35	3.97	6.98	4.11	1.90
PBIO_E+WF	2.63	**2.12**	**1.36**	2.93	2.39	**0.98**
PBIO_E+LSTM	2.79	3.02	2.74	2.19	2.71	1.71
P_B	9.90	7.36	8.29	6.06	8.68	9.28
P_E	3.93	3.94	2.87	**3.26**	4.57	2.87
P_E+2GRU	**3.81**	2.87	**2.11**	3.82	3.27	**1.87**
P_E+diffParam	4.95	**2.44**	2.86	5.16	**3.20**	1.89
P_B	9.90	7.36	8.29	6.06	8.68	9.28
P_E	3.93	3.94	2.87	**3.26**	4.57	2.87
P_E+2GRU	**3.81**	2.87	**2.11**	3.82	3.27	**1.87**
P_E+diffParam	4.95	**2.44**	2.86	5.16	**3.20**	1.89
Dim_B	3.00	2.36	1.50	4.54	4.13	1.45
Dim_E	4.75	2.92	2.15	7.46	5.63	1.87
Dim_E+2GRU	4.53	4.00	2.11	6.74	5.91	3.02
Dim_E+20epochs	**1.29**	**1.01**	**0.85**	**1.45**	**1.65**	**0.82**
PT_B	12.30	10.65	11.07	3.24	6.77	4.87
PT_E	**1.97**	**2.82**	**1.53**	**3.02**	**2.80**	**1.07**
PT_E+2GRU	4.11	3.68	1.60	4.21	4.20	1.58
RO_B	3.92	**3.30**	**2.10**	**3.29**	**3.81**	**1.54**
RO_2GRU	**3.28**	5.11	2.59	3.59	5.12	2.41

Table 2: Standard deviation on all configuration