Semantic reranking of CRF label sequences for verbal multiword expression identification

Erwan Moreau, Ashjan Alsulaimani, Alfredo Maldonado, Lifeng Han, Carl Vogel, Koel Dutta Chowdhury

Chapter 6

Semantic reranking of CRF label sequences for verbal multiword expression identification

Erwan Moreau
ADAPT Centre, Trinity College Dublin

Ashjan Alsulaimani
Trinity College Dublin

Alfredo Maldonado
ADAPT Centre, Trinity College Dublin

Lifeng Han
ADAPT Centre, Dublin City University

Carl Vogel
Trinity Centre for Computing and Language Studies, Trinity College Dublin

Koel Dutta Chowdhury
ADAPT Centre, Dublin City University

Verbal multiword Expressions (VMWE) identification can be addressed successfully as a sequence labelling problem via conditional random fields (CRFs) by returning the one label sequence with maximal probability. This work describes a system that reranks the top 10 most likely CRF candidate VMWE sequences using a decision tree regression model. The reranker aims to operationalise the intuition that a non-compositional MWE can have a different distributional behaviour than

Erwan Moreau, Ashjan Alsulaimani, Alfredo Maldonado, Lifeng Han, Carl Vogel & Koel Dutta Chowdhury. 2018. Semantic reranking of CRF label sequences for verbal multiword expression identification. In Stella Markantonatou, Carlos Ramisch, Agata Savary & Veronika Vincze (eds.), *Multiword expressions at length and in depth: Extended papers from the MWE 2017 workshop*, 177–207. Berlin: Language Science Press.
DOI:10.5281/zenodo.1469559

that of its constituent words. This is why it uses semantic features based on comparing the context vector of a candidate expression against those of its constituent words. However, not all VMWE are non-compostional, and analysis shows that non-semantic features also play an important role in the behaviour of the reranker. In fact, the analysis shows that the combination of the sequential approach of the CRF component with the context-based approach of the reranker is the main factor of improvement: our reranker achieves a 12% macro-average F1-score improvement on the basic CRF method, as measured using data from PARSEME shared task on VMWE identification.

1 Introduction

The automatic identification of multiword expressions (MWEs) is an important but challenging task in natural language processing (NLP) (Sinclair 1991; Sag et al. 2002). An effort in response to this challenge is the shared task on detecting multiword verbal constructions (Savary et al. 2017) organised by the PARSing and Multiword Expressions (PARSEME) European COST Action.[1] The shared task consisted of two tracks: a closed one, restricted to the data provided by the organisers, and an open track that permitted participants to employ additional external data.

The ADAPT team participated in the closed track with a system that exploited syntactic dependency features in a Conditional Random Fields (CRF) sequence model (Lafferty et al. 2001) and ranked 2nd in the detection of full MWEs in most languages (Maldonado et al. 2017).[2] In addition to extending the description of our CRF-based solution in §3, this chapter focuses on a second component aimed at reranking the top 10 sequences predicted by the CRF decoder, using a regression model. This component, called a *semantic reranker* and described in §4, increases the performance of the system by an average 12% in F1-score over the datasets at the MWE level. Because the reranker requires a third-party corpus, the system using both components (the CRF-based and the reranker) would compete in the open track task.

The design of the semantic reranker was originally oriented towards detecting non-compositional expressions. In such expressions, the meaning of the expression cannot be obtained by combining the meanings of its individual words, i.e. the actual meaning is unrelated to the literal meaning (e.g. *to kick the bucket*). This is a distinctive feature which can be recognised by comparing their context

[1] http://www.parseme.eu.

[2] Official results: http://multiword.sourceforge.net/sharedtask2017/; system details, feature templates, code and experiment instructions: https://github.com/alfredomg/ADAPT-MWE17.

vectors (these vectors can be built from any large corpus). This idea has been used for bigram expressions (Schütze 1998; Maldonado & Emms 2011), and we adapted it to multiword expressions. Nevertheless, most verbal MWEs are actually compositional, at least to some extent (e.g. *to give somebody a break*). In light of the performance improvement obtained when adding the reranker to our system, it is clear that the reranker gives a boost in detecting MWEs across the board, and not only for a few non-compositional expressions. In order to understand how the reranker contributes to the performance, we carried out a thorough study and provide a detailed analysis of the results in §5.

2 Related work

MWEs have long been discussed in NLP research and a myriad of processing techniques have been developed, such as combining statistical and symbolic methods (Sag et al. 2002), single and multi-prototype word embeddings (Salehi et al. 2015), and integrating MWE identification within larger NLP tasks, such as parsing (Green et al. 2011; 2013; Constant et al. 2012) and machine translation (Tsvetkov & Wintner 2010; Salehi et al. 2014a,b).

More directly related to our closed-track approach are works such as that of Venkatapathy & Joshi (2006), who showed that information about the degree of compositionality of MWEs helps the word alignment of verbs, and of Boukobza & Rappoport (2009) who used sentence surface features based on the canonical form of VMWEs. In addition, Sun et al. (2013) applied a hidden semi-CRF model to capture latent semantics from Chinese microblogging posts; Hosseini et al. (2016) used double-chained CRF for minimal semantic units detection in a SemEval task. Bar et al. (2014) discussed that syntactic construction classes are helpful for verb-noun and verb-particle MWE identification. Schneider et al. (2014) also used a sequence tagger to annotate MWEs, including VMWEs, while Blunsom & Baldwin (2006) and Vincze et al. (2011) used CRF taggers for identifying continuous MWEs.

In relation to our open-track approach, Attia et al. (2010) demonstrated that large corpora can be exploited to identify MWEs, whilst Legrand & Collobert (2016) showed that fixed-size continuous vector representations for phrases of various lengths can have a performance comparable to CRF-based methods in the same task. Finally, Constant et al. (2012) used a reranker for MWEs in an *n*-best parser. We combine these ideas by reranking the *n* best CRF VMWE predictions for each sentence using regression scores computed from vectors that represent different combinations of VMWE candidates. The vectors are computed from a large corpus, namely EUROPARL's individual language subcorpora.

3 VMWE identification via CRF

We decided to model the problem of VMWE identification as a sequence labelling and classification problem. We operationalise our solution through CRFs (Lafferty et al. 2001), implemented using the CRF++ system.[3] CRFs have been successfully applied to such sequence-sensitive NLP tasks such as segmentation, named-entity recognition (Han et al. 2013; 2015) and part-of-speech (POS) tagging. Our team attempted 15 out of the 18 languages involved in the shared task. It should be noted that of these 15 languages, four (Czech, Farsi, Maltese and Romanian) were provided without syntactic dependency information, although morphological information (i.e. tokens' lemmas and POS) was indeed supplied. The data for the languages we did not attempt (Bulgarian, Hebrew and Lithuanian) lacked even morpho-syntactic information, leaving the CRF with only tokens as features; so we felt that we were unlikely to obtain good results with them, and chose to focus on the richer datasets.

3.1 Features

We assume that features based on the relationships between the different types of morpho-syntactic information provided by the organisers will help identify VMWEs. Ideally, one feature set (or *feature template* in the terminology of CRF++) per language should be developed. Due to time constraints, we developed a feature set for three languages (German, French and Polish), then for every language the feature template that performed best in cross-validation among these three was selected.

For each token in the corpus, the direct linguistic features available are its word surface (W), word lemma (L) and POS (P). In the languages where syntactic dependency information is provided, each token also has its head's word surface (HW), its head's word lemma (HL), its head's POS (HP) and the dependency relation between the token and its head (DR). It is possible to create CRF++ feature templates that combine these features. In addition, it is also possible to use the predicted output label of the previous token (B).

The three final feature templates, which we call FT3, FT4 and FT5,[4] are shown in Table 1. Whilst the feature templates in this table are expressed in the CRF++ format, a comment (starting with #) at each feature (line) expresses the type of feature and its relative position to the current token. For instance, `L-2` refers to

[3] https://taku910.github.io/crfpp/. Release 0.58, Last verified 2017-12-29.

[4] The feature template numbering starts at 3 for consistency with their original description in Maldonado et al. (2017).

Table 1: CRF++ Feature Templates developed. Example: template `U32:%x[0,3]` indicates current token (row 0, i.e. current row) and lemma (column 3) while template `U41:%x[-1,2]` refers to the previous token (row -1 from current row) and POS tag (column 2), etc.

FT3	FT4	FT5
`# L-2`	`# L-2`	`# L-2`
`U30:%x[-2,3]`	`U30:%x[-2,3]`	`U30:%x[-2,3]`
`# L-1`	`# L-1`	`# L-1`
`U31:%x[-1,3]`	`U31:%x[-1,3]`	`U31:%x[-1,3]`
`# L`	`# L`	`# L`
`U32:%x[0,3]`	`U32:%x[0,3]`	`U32:%x[0,3]`
`# L+1`	`# L+1`	`# L+1`
`U33:%x[1,3]`	`U33:%x[1,3]`	`U33:%x[1,3]`
`# L+2`	`# L+2`	`# L+2`
`U34:%x[2,3]`	`U34:%x[2,3]`	`U34:%x[2,3]`
`# L-2/L-1`	`# L-2/L-1`	`# L-2/L-1`
`U35:%x[-2,3]/%x[-1,3]`	`U35:%x[-2,3]/%x[-1,3]`	`U35:%x[-2,3]/%x[-1,3]`
`# L-1/L`	`# L-1/L`	`# L-1/L`
`U36:%x[-1,3]/%x[0,3]`	`U36:%x[-1,3]/%x[0,3]`	`U36:%x[-1,3]/%x[0,3]`
`# L/L+1`	`# L/L+1`	`# L/L+1`
`U37:%x[0,3]/%x[1,3]`	`U37:%x[0,3]/%x[1,3]`	`U37:%x[0,3]/%x[1,3]`
`# L+1/L+2`	`# L+1/L+2`	`# L+1/L+2`
`U38:%x[1,3]/%x[2,3]`	`U38:%x[1,3]/%x[2,3]`	`U38:%x[1,3]/%x[2,3]`
`# P`	`# HL/DR`	`# HL/DR`
`U00:%x[0,2]`	`U01:%x[0,4]/%x[0,6]`	`U01:%x[0,4]/%x[0,6]`
`# HL/DR`	`# P/DR`	`# P/DR`
`U01:%x[0,4]/%x[0,6]`	`U02:%x[0,2]/%x[0,6]`	`U02:%x[0,2]/%x[0,6]`
`# P/DR`	`# HP/DR`	`# HP/DR`
`U02:%x[0,2]/%x[0,6]`	`U03:%x[0,5]/%x[0,6]`	`U03:%x[0,5]/%x[0,6]`
`# HP/DR`		
`U03:%x[0,5]/%x[0,6]`	`# Previous token's label`	`# Previous token's label`
	`B`	`B`
`# Previous token's label`		
`B`	`# P-2`	`# P-2`
	`U40:%x[-2,2]`	`U40:%x[-2,2]`
	`# P-1`	`# P-1`
	`U41:%x[-1,2]`	`U41:%x[-1,2]`
	`# P`	`# P`
	`U42:%x[0,2]`	`U42:%x[0,2]`
	`# P+1`	`# P+1`
	`U43:%x[1,2]`	`U43:%x[1,2]`
	`# P+2`	`# P+2`
	`U44:%x[2,2]`	`U44:%x[2,2]`
	`# P-1/P`	`# P-1/P`
	`U45:%x[-1,2]/%x[0,2]`	`U45:%x[-1,2]/%x[0,2]`
	`# P/P+1`	`# P/P+1`
	`U46:%x[0,2]/%x[1,2]`	`U46:%x[0,2]/%x[1,2]`
		`# L/HP`
		`U52:%x[0,3]/%x[0,5]`

the lemma of the token at position $i-2$ relative to the current token at position i, `P+1` refers to the part of speech of the token at position $i+1$, and `HL/DR` refers to the combination of head's lemma of the current token and the dependency relation between the current token and its head.[5]

FT5 is based on FT4, which in turn, is based on FT3. FT3 is itself based on the CRF++ example feature template, commonly used in NER experiments. The difference between FT3 and this example feature template is that FT3 exploits syntactic dependency information.

We conducted preliminary 5-fold cross validation experiments on German, French and Polish training data using the FT3 template. We then started tweaking the template independently for each of these three languages based on successive 5-fold cross validation results. This exercise resulted in the three final templates: FT3 for French and FT5 for German and Polish. Given that FT4's performance was very similar to that of FT5, we decided not to discard it.

During this preliminary experimentation, we also observed that templates exploiting token word surface features (W) performed unsurprisingly worse than those based on token lemmas (L) and POS (P). Templates using head features (HL, HP, DR) in addition to token features (L, P) fared better than those relying on token features only.

We also attempted to test the assumption that these feature templates would perform similarly in other languages of the same language family. That is, that FT3 would also perform better than FT4 and FT5 in other Romance languages and that FT5 would score higher than FT3 and FT4 in the rest of the languages. So we conducted a final set of 5-fold cross validation experiments on all 15 languages, this time trying each feature template (FT3, FT4 and FT5) independently on each language. The results are shown in Table 2. The F1 scores in bold italic are the maximum scores per language. For each given language, the results of the three templates are very similar. Therefore we are not able to comfortably confirm or refute our language family assumption. Nonetheless, we decided to choose for the final challenge the template that maximised the MWE-based F1 score for each language.

In order to use these templates with the provided data, we combined the supplied PARSEMETSV (VMWE annotations) and CONLLU files (linguistic features) into a single file. The training and blind test files were combined separately. The resulting file is also columnar in format, with column 0 representing the token ID as per the original PARSEMETSV file, column 1 the token's surface form, col-

[5] CRF++ feature template format described in https://taku910.github.io/crfpp/#templ Last verified 2017-12-19.

Table 2: F1 scores from cross validation experiments on 15 languages using feature templates FT3, FT4 and FT5.

Lang	CS		DE		EL		ES		FA	
Eval. Type	MWE	Token	MWE	Token	MWE	Token	MWE	Token	MWE	Token
FT3	54.23	70.79	23.84	39.02	***50.41***	***62.03***	***56.04***	60.74	76.09	83.52
FT4	55.91	71.81	24.41	39.62	50.16	61.76	55.72	60.77	77.88	84.75
FT5	***57.12***	***72.57***	***25.23***	***40.53***	49.72	62.02	55.70	***61.00***	***78.61***	***85.24***

Lang	FR		HU		IT		MT		PL	
Eval. Type	MWE	Token	MWE	Token	MWE	Token	MWE	Token	MWE	Token
FT3	3.99	6.71	66.03	70.24	***65.85***	75.81	81.44	80.96	28.31	31.12
FT4	4.20	6.92	65.70	70.56	65.62	76.07	81.30	81.00	28.19	30.80
FT5	***5.35***	***7.96***	***66.28***	***71.21***	65.6	***76.08***	***81.86***	***81.76***	***28.68***	***31.51***

Lang	PT		RO		SL		SV		TR	
Eval. Type	MWE	Token	MWE	Token	MWE	Token	MWE	Token	MWE	Token
FT3	56.56	64.81	75.87	***83.76***	***37.06***	***48.90***	19.68	20.09	***49.71***	59.42
FT4	***56.73***	65.32	75.91	83.47	34.04	46.70	***24.47***	***24.56***	49.49	***59.43***
FT5	56.64	***65.52***	***76.00***	83.69	34.65	47.57	22.09	22.33	49.46	59.38

umn 2 the token's POS tag (P), column 3 the lemma (L), column 4 the head's lemma (HL), column 5 the head's POS tag (HP), column 6 the dependency relationship between the token and its head (DR), and column 7 the VMWE label for the token.

The VMWE label was changed from the numerical values in the PARSEMETSV file to "B" for the tokens that start a VMWE and "I" for subsequent tokens that belong to a VMWE. This labelling scheme (usually called BIO, for *Begin, Inside, Outside*) is common in CRF-based implementations of NER systems. The BIO scheme can represent several consecutive VMWEs but cannot represent embedded or overlapping VMWEs, so these were ignored and a single B or I label was used for overlapping tokens.[6] The proportion of overlapping VMWEs is between 2 and 6% for Czech, German, Spanish, French, Italian, Polish, Portuguese and

[6]Remark: Schneider et al. (2014) proposes several tagging schemes, some using special "o" labels for discontinuous expressions; since we use the most simple scheme (BIO), the words which appear between the lexicalized components of the expressions are labeled with a regular "O".

Swedish, and it is even less for the rest of the languages we studied (see Maldonado & QasemiZadeh 2018 [this volume] for further details). Because of these low proportions, we consider embedded/overlapping VMWEs to have only a small negative impact on our system's performance. Therefore, we decided to ignore them.

Our system does not distinguish among the different categories of VMWEs, treating them all equally. The templates in Table 1 make reference to each feature based on the position of the current token and the column in which they appear in the input file.

3.2 CRF results

Table 3 shows, under the "CRF only" category, the Precision (P), Recall (R) and F1 (F) scores on the test set based on the shared task two evaluation modalities, MWE-based and token-based.[7] On token-based evaluation, our system was ranked in first place in French, Polish and Swedish, second place in eight languages (Czech, Greek, Farsi, Maltese, Portuguese, Romanian, Spanish and Turkish) and third in three (German, Italian and Slovenian). For MWE-based scores, our system ranked second place in nine languages (Farsi, French, Italian, Maltese, Polish, Portuguese, Slovenian, Spanish and Swedish) and third in four languages (Czech, German, Hungarian and Turkish). If all languages' scores are averaged per system, our system ranks in third place on both token-based and MWE-based evaluation (see Maldonado & QasemiZadeh 2018 [this volume] for average scores per system).

4 Semantic reranker

4.1 Motivations

The semantic reranker is the second component of the VMWE detection system. While the first component (CRF) offers a decent level of performance in its predictions, the reranker is intended to fix as many mistaken predictions as possible, by exploiting features that CRF are poorly equipped to deal with. These features are based on a distributional semantics approach (Schütze 1998; Maldonado & Emms 2011): they rely on comparing context vectors which are extracted from a *reference corpus* (usually a large third-party corpus). As it is often the case with

[7] We did not include the languages for which there is no Europarl data in this table.

Table 3: Performance by language according to official evaluation measures for the CRF component alone and the two components together (CRF and semantic reranker) P/R/F stands for precision/recall/f-score. All the values are expressed as percentages. The last two rows show the macro-average performance over the 12 languages.

Lang.	Eval. Type	CRF only			CRF + Reranker			Improvement (%)		
		P	R	F	P	R	F	P	R	F
CS	MWE	59.3	56.2	57.7	79.8	63.4	70.6	+34.5	+12.8	+22.4
	Token	81.9	65.6	72.9	86.4	65.7	74.6	+5.4	+0.2	+2.4
DE	MWE	33.1	17.4	22.8	53.5	19.6	28.7	+61.9	+12.6	+25.9
	Token	70.6	28.4	40.5	72.2	25.6	37.8	+2.3	-9.7	-6.6
EL	MWE	34.4	28.8	31.3	45.1	32.2	37.6	+31.2	+11.8	+19.9
	Token	53.8	36.0	43.1	54.8	36.3	43.6	+1.8	+0.7	+1.1
ES	MWE	61.1	34.8	44.3	66.7	38.8	49.0	+9.2	+11.5	+10.6
	Token	74.5	36.7	49.2	74.2	38.9	51.1	-0.3	+6.0	+3.9
FR	MWE	61.5	43.4	50.9	75.6	47.6	58.4	+22.9	+9.7	+14.8
	Token	80.9	49.6	61.5	82.6	50.0	62.3	+2.1	+0.7	+1.2
HU	MWE	75.7	59.9	66.9	74.5	63.3	68.5	-1.5	+5.7	+2.4
	Token	78.5	57.1	66.1	77.3	61.5	68.5	-1.5	+7.8	+3.7
IT	MWE	61.7	14.2	23.1	70.8	9.2	16.3	+14.6	-35.2	-29.5
	Token	69.6	15.3	25.1	76.1	9.7	17.3	+9.2	-36.4	-31.2
PL	MWE	78.0	60.2	68.0	83.7	63.8	72.4	+7.4	+6.0	+6.6
	Token	87.4	62.3	72.7	87.0	63.9	73.7	-0.5	+2.6	+1.3
PT	MWE	64.1	53.2	58.1	75.9	57.2	65.2	+18.3	+7.5	+12.2
	Token	83.5	60.5	70.2	82.2	60.1	69.4	-1.5	-0.8	-1.1
RO	MWE	75.5	71.4	73.4	90.4	77.6	83.5	+19.8	+8.7	+13.8
	Token	88.3	76.4	81.9	91.8	77.9	84.3	+4.0	+2.1	+3.0
SL	MWE	51.4	29.0	37.1	68.6	32.4	44.0	+33.5	+11.7	+18.7
	Token	72.9	32.6	45.1	75.4	32.4	45.3	+3.5	-0.8	+0.5
SV	MWE	48.6	22.0	30.3	48.7	23.7	31.9	+0.2	+7.7	+5.2
	Token	52.5	22.5	31.5	52.1	24.1	32.9	-0.7	+7.0	+4.6
macro-average	MWE	58.7	40.9	48.2	69.4	44.1	53.9	+18.3	+7.8	+11.9
	Token	74.5	45.3	56.3	76.0	45.5	56.9	+2.0	+0.6	+1.1

complex machine learning problems, the orthogonality of the information, obtained on the one hand from the sequential CRF model and computed on the other hand from an independent semantic vector space, proves fruitful; this will be demonstrated in §5.

4.2 Design

Intuitively, the goal is to estimate whether a candidate expression should be considered a MWE or not. Thus, the core part of the reranker is to generate features which are relevant to assess the likeliness of a given candidate MWE being an actual MWE. These expression-level features are then combined to produce a set of sentence-level features, which in turn are used to train a decision tree regression model. Later, this model is applied to the candidate sequences provided by the CRF component, so that their predicted scores can be compared; finally, the sequence with the highest score (among a set of N candidate sequences) is selected as the final answer. This is why the reranker receives the output produced by CRF++ in the form of the 10 most likely predictions for every sentence.

4.3 A distributional semantics approach

Distibutional semantics is a well known method to represent the meaning of a single word as a context vector (Schütze 1998). However, our algorithm must calculate a context vector for the multiple words in an expression whether they are continuous or discontinuous (see §5.7). This raises new questions about the optimal way to take the co-occurences into account in the algorithm. To the authors' knowledge, such questions have not been previously studied.

In order to calculate the context vector, we count the words co-occurring with the MWE in a *reference corpus* (see §4.4). Given a candidate MWE identified by the CRF, the reference corpus is searched for every occurrence of this MWE. This consists in matching the words which compose the expression; because MWEs are not continuous in general, the matching only requires that the words appear in the same order in a sentence, i.e. allows other words to appear between the words of the expression.[8] As a consequence, false positive matches may happen, in particular if the words of the MWE appear in a sentence by chance, without any direct relation between them (neither syntactic or semantic).

[8] This implies that ambiguities can arise if the same word is used several times in a sentence. In such cases, the matching always selects the shortest possible distance between the first and the last word if the MWE.

Once an occurrence of the MWE expression is identified, the words which appear within a fixed-size window around its lexicalized components are counted as its co-occurrences. The number of times a given word co-occurs with the MWE across all the sentences in the reference corpus is recorded; by doing this for every word which co-occurs with a component of an MWE, we obtain a vector which represents the meaning of the MWE. However this method is traditionally used with single words, and its adaptation to sequences of words raises new questions. This is why we propose several options, as detailed below, that determine how the co-occurences are extracted and counted. The combinations of these options offer multiple possibilities that we analyse in §5.8.

- *IncludeWordsInExpression (WIE):* This option determines whether to add the actual expression words to the context vector as contexts of other components or to exclude them, when they fall within the scope of the window.
- *MultipleOccurencesPosition (MO):* This option determines whether or not to count multiple occurrences of a word within the window scope. Such a word can be part of the actual expression or not, depending on the first option.
- *ContextNormalization (CN):* This option determines whether the frequencies of the co-occurrences are normalized for every occurrence of the expression, i.e. divided by the number of co-ccurrences found.[9] This is meant to account for the differences in the number of context words across different sentences or expressions (since a longer expression generally has more words in its context window).

Table 4 illustrates the impact of these options when applied to the following example, in which the words of the expressions are in bold:

(1) French (Indo-European; FR training data)
Les gens ne ***se rendent*** *pas bien* ***compte*** *du coût énorme de*
The people NEG self give not well account of.the cost enormous of
l' opération.
the operation.
'People do not fully **realize** the enormous cost of the operation.'

[9] Otherwise absolute frequencies are used at the level of a single expression. In both cases a different stage of normalization is carried out once all the occurrences of the expression have been collected, where the values are divided the number of occurrences.

Table 4: Example of how context words are counted (for one occurence of the expression). A context window of size 2 is assumed on both sides. IWIE and MO represent the options *IncludeWordsInExpression, MultipleOccurrences*, respectively. The values are represented in the case where *ContextNormalization* is false, i.e. they are not normalized; in the case where *ContextNormalization* is true, every value is divided by the sum of the values in the row.

IWIE	MO	gens	ne	**se**	**rendent**	pas	bien	**compte**	du	coût
False	False	1	1	0	0	1	1	0	1	1
True	True	1	2	1	1	3	2	0	1	1
False	True	1	2	0	0	3	2	0	1	1
True	False	1	1	1	1	1	1	0	1	1

4.4 Third-party reference corpus: Europarl

We use Europarl (Koehn 2005) as reference corpus, because it is large and conveniently contains most of the languages addressed in the shared task data. However there is no Europarl data for Farsi, Maltese and Turkish. This is why these languages are excluded from this part of the process. For each of the 12 remaining languages, we use only the monolingual Europarl corpus, and we tokenise it using the generic tokeniser provided by the shared task organisers. However this tokeniser was not necessarily used for the shared task data, because each language team was free to use their own tools or to use existing pre-tokenised data.[10] Therefore, discrepancies are to be expected between the tokenisation of the shared task corpus and the one performed on Europarl. Additionally, Europarl consists of raw text, so the reranker cannot use the morpho-syntactic information (POS tags and lemmas) provided with the input data.

4.5 Features

The core component of the reranker computes a set of feature values for every sequence proposed by the CRF; these features are later used for training or applying the decision tree model. First, these features include a few simple values which are either available directly from the CRF output or easily computed from the reference corpus:

- The confidence score assigned by the CRF component to the sequence (i.e. the probability of the sequence according to the CRF);

[10]For instance, the French dataset originates from several existing corpora; their tokenisation follows language-specific patterns which cannot be obtained with a generic tokeniser.

- The number of expressions labeled by the CRF in the sequence;
- The minimum/mean/maximum number of words in the expression, over the expressions of the sequence;
- The minimum/mean/maximum frequency of the expression in the reference corpus, over the expressions of the sequence.

The original intuition behind the semantic reranker is to calculate features which give indications about the level of compositionality between the words in the expression. The underlying assumption is that such features might help detect at least non-compositional expressions. In the past this idea has been used successfully to detect collocations among two-words sequences: in Maldonado & Emms (2011), for every two-words collocation *xy* word vectors are computed for the word *x*, the word *y* and the full sequence *xy*; the cosine similarity is used to compare (1) the vector for *x* with the vector for *xy* and (2) the vector for *y* with the vector for *xy* . In a compositional expression both scores are expected to be high, because the semantic overlap between the full expression and each of its words is high, as opposed to a non-compositional expression. Hence the average similarity score between (1) and (2) can be used as a measure of the compositionality of the pair.

This idea is generalized to the case of VMWEs of any length by comparing different parts of the expression against the full candidate expression; we call *pseudo-expressions* these "parts of the expression". Every pseudo-expression extracted from the candidate expression is analyzed as if it was a regular candidate expression: the reference corpus is searched to identify all its occurrences, then a context vector is built, as described in §4.3. Examples of pseudo-expressions based on the expression *avoir sa place* ('have their place') are provided below.

Four different similiarity measures have been implemented for comparing pairs of context vectors: Jaccard index, min/max similarity, and cosine similarity with or without inverse document frequency (IDF) weighting. Additionally to the semantic context vectors comparison, the frequencies of the pseudo-expressions are compared with respect to the frequency of the full MWE: this feature is the ratio of the full expression frequency divided by the pseudo-expression frequency. Finally every candidate MWE is compared against every other candidate MWE in the 10 predicted sequences. Thus, for each of the three groups of features below, the frequency ratio and the similarity score obtained between the context vectors of the pseudo-MWEs and the full MWE are added as features.

- Features comparing each pseudo-MWE consisting of a single word of the MWE against the full MWE. Example: for the candidate expression *avoir sa*

place ('have their place'), three comparisons are performed: *avoir* vs. *avoir sa place*, *sa* vs. *avoir sa place*, *place* vs. *avoir sa place*.

- Features comparing each pseudo-MWE consisting of the MWE minus one word against the full MWE. Example: for the candidate expression *avoir sa place*, three comparisons are performed: *sa place* vs. *avoir sa place*, *avoir place* vs. *avoir sa place*, *avoir sa* vs. *avoir sa place*.
- Features comparing one of the other MWEs found in the 10 predicted sequences[11] against the current MWE.[12]

The main difficulty in representing a predicted sequence as a fixed set of features is that each sentence can contain any number of MWEs, and each MWE can contain any number of words. We opted for a simple method which consists in "summarising" any non-fixed number of features with three statistics: minimum, mean and maximum. For instance, the similarity scores between each individual word and the corresponding MWE (n scores, where n is the number of words) are represented with these three statistics, each computed across these n scores. Similarly, if the sequence contains m expressions, a feature f has m values $f_1, \ldots, f_m$, with each value f_i corresponding to one expression; here again the minimum, mean and maximum are calculated across these m values, i.e. every expression-level feature f is converted into three features f_{min}, f_{mean} and f_{max} in the final set of sequence-level features.

4.6 Supervised regression and cross-validation process

As explained above, the reranker has to assign a score to each of the 10 sequences provided by the CRF component, in order to select the highest one. We use regression, rather than classification, because a categorical answer would cause some sentences to have either no positive answer or multiple positive answers in its set of predicted sequences, thus making the decision impossible.

[11] This includes the 9 other sequences as well as the other candidate expressions in the current one, if any.

[12] The last group is not meant to measure compositionality of the expression. The rationale is that such features might help eliminate candidate expressions which are very different from the other candidates, under the hypothesis that likely candidates tend to share words together (such features are unlikely to help with sentences which contain several expressions). As explained in §5.5, this group of features turned out to be the least useful.

In training mode, an instance (i.e. sequence) is assigned the score 1 if it corresponds exactly to the sequence in the gold standard, or 0 otherwise.[13] Since the 10 candidate sequences are all different, there should always be only one correct sequence. This way, the regression model assigns scores in $[0, 1]$ to every instance, with the highest values expected to be more likely correct answers. As regression model, we use the Weka (Hall et al. 2009) implementation of decision trees regression (Quinlan 1992): the final score is determined by the decision tree rules, then by a linear combination of the features values. This choice has the advantage of simplicity, but also of the interpretability of decision trees models. Of course, other regression models could be considered as well.

Because of the two-components approach (CRF and reranker), the training process requires special care. In order to train the reranker with the kind of data that it will receive in testing mode, predictions from the CRF are needed. The testing data cannot be used for this, which is why we use 5 fold cross-validation on the training data: on each of the five 20% subsets of the data, a model trained from the 80% data left is applied. This way the reranker can be fed with real predictions for the full training data, including the classification errors of the CRF. If necessary, the cross-validation process can be repeated, for instance to tune the parameters of the reranker. Otherwise, the reranker can simply be trained with the full training set of predictions, then applied to the test set (after the CRF predictions have been predicted on this test set).

5 Results and analyses

In this section we present detailed results of our system and analyze how the reranker helps improving performance with respect to various factors. All the experiments presented in this section have been carried out using the official PARSEME shared task 2017 training and test data. Unless otherwise stated, we use the same "standard" configuration of options for the reranker throughout the experiments (e.g. context window size, minimum frequency, etc.).

5.1 Results

Table 3 shows the performance of both the CRF component and the semantic reranker by language, as well as the improvement brought by the reranker. Despite differences by language, all but one language (Italian) show a significant

[13] It might happen that none of the 10 sequences corresponds to the gold sequence; in such cases all the instances are left as negative cases.

increase in MWE-level F-score, with a macro-average F-score improvement of +11.9%. The increase in token-level F-score is much smaller, with even a decrease in a few languages; the macro-average token-level F-score improvement is only +1.1%. This means that the reranker does not drastically change the expressions predicted by the CRF (hence the little improvement at the token level), but instead tends to fix the proposed expressions by finding their exact boundaries, thus bringing the MWE F-score closer to the token F-score. This is because the top 10 CRF predictions tend to be variants of one another, rather than drastically different labelings; they frequently focus on the same part(s) of the sentence, varying only by labeling one or two words differently; thus the reranker seldom introduces a new expression that the top prediction would have missed, but can select a more likely variant among the remaining 9 predictions. This hypothesis is also backed by the observation that the increase in precision is larger in general than the increase in recall: the reranker mostly follows the top CRF prediction and possibly fixes it, hence turning a false positive instance into a true positive. These observations are consistent with the design of the system and validate the reranking approach in general.

5.2 Error analysis methodology

The performance of the reranker can be evaluated straightforwardly using the official evaluation measures, as presented above; these measures are useful to compare against other systems or between datasets. However, in order to get a clear understanding of how the system works, we also look at the different combinations of error status between the CRF component and the semantic reranker: the CRF is said to be right if and only if it ranks the actual (gold standard) answer as its top prediction; similarly, the reranker is right if and only if it assigns the top score to the actual answer. Thus the four following categories are defined:

- **RR** stands for right-right, which means that the CRF component ranked the right answer as first sequence (first **R**) and the reranker kept it as the final answer (second **R**).
- **WR** stands for wrong-right: the CRF answer was wrong, but the reranker successfully selected an alternative answer.
- **RW** stands for right-wrong: in this case the reranker mistakenly changed the CRF answer, which was correct in the first place.
- **WW** stands for wrong-wrong: the CRF answer was wrong, and the reranker either kept it or changed it to another wrong answer.

These four categories cover all the cases, except when the correct answer is not present in the 10 most likely sequences that the CRF component provides. For this case we use the special label **GoldNotIn10**.

It is worth noticing that the reranker works at the sentence level (as opposed to the expression level or the token level). This is why these categories apply to complete sentences, in accordance with the design of the two-components system. In particular, the number of expressions in a sentence is not taken into account in this categorization. As a consequence, sentences which contain no expression at all are considered as important as sentences which contain one or multiple expressions.

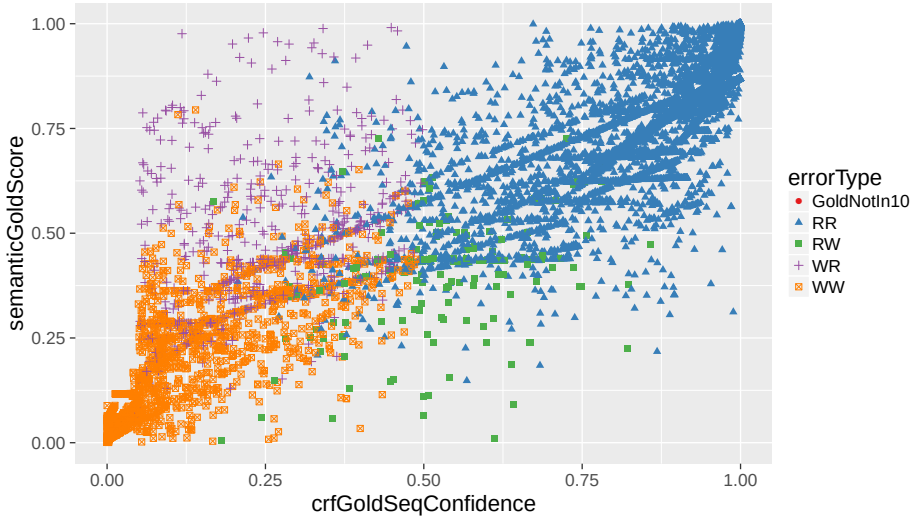

Figure 1: Reranker score w.r.t CRF confidence for the gold sequence in every sentence, by error type (all languages together). A sentence is represented in the right half (resp. left half) if the CRF assigned a high confidence (resp. low) to its gold sequence, i.e. the CRF answer is correct (resp. incorrect). Similarly, a sentence appears in the top half (resp. bottom half) if the reranker assigned a high score (resp. low) to its gold sequence, i.e. the reranker answer is correct (resp. incorrect).[14]

Figure 1 gives an overview of how the reranker improves performance over the CRF predictions. Every point in this graph represents a sentence, positioned according to the CRF confidence (X axis) and reranker final score (Y axis) for its gold sequence. This way, if the CRF finds the right answer for a sentence,

[14]Remark: Category *GoldNotIn10* is not visible on this graph, since in such cases the gold sequence cannot be assigned a CRF confidence nor a reranker score.

i.e. the gold sequence obtains the highest confidence among the 10 sequences, it is represented in the right half of the graph, and conversely for wrong answers. If the reranker finds the right answer, it assigns a high score to the gold sequence, so the sentence appears in the top half, and conversely. This explains why the four error types appear mostly clustered each in its own quadrant: the top right quadrant contains sentences for which the correct sequence is recognized by both the CRF and the reranker (hence in the **RR** error category), and the bottom left contains sentences for which neither finds the right answer (**WW**). The last two quadrants are the interesting ones, since this is where the reranker changes the CRF prediction to another sequence: in the top left quadrant, the **WR** points correspond to successful changes, whereas the **RW** cases, in the bottom right quadrant, correspond to mistakes introduced by the reranker.[15] It can be observed that the former category outnumbers the latter, thus confirming the positive contribution of the reranker.

5.3 Insight: what the reranker actually does

The vast majority of the sentences (85.3% of a dataset in average) fall into the *RR* category, i.e. the reranker simply confirms the correct CRF answer. The *WW* cases account for 7.6% of the sentences, and the *GoldNotIn10* cases for 4.4%. The reranker actually changes only 2.7% of the answers, and when it does, it does it correctly 81.5% of the time (2.2% *WR*, 0.5% *RW*).

Figure 2 shows how the positions of the sequences selected by the reranker are distributed. The reranker strongly favors top positions for its selected sequence; more precisely, as the position of the sequence decreases, the number of sentences for which this position is selected decreases exponentially (this is why a logarithmic scale is used in Figure 2). This trend is regular from the top position, which is selected 92.5% of the time, down to the 9th position, selected in only two cases.[16] This shows that increasing the number of candidate sequences supplied by the CRF (10 in all our experiments) would not improve the performance of the reranker, since it seldom selects a sequence associated with a low CRF confidence (the importance of the CRF confidence as a feature is shown more clearly

[15] This graph can give the impression that one could easily prevent **RW** mistakes (bottom right quadrant) by accepting any CRF answer with high confidence, but this is due to the fact that only the gold sequence is represented here. Thus, for cases where the gold sequence has low confidence, some other (wrong) sequence has high confidence. Therefore selecting the highest confidence sequence would simply prevent any reranking to happen, in particular for the CRF mistakes which can be fixed.

[16] The small rebound in position 10 is not significant, as it represents only 3 cases.

in §5.5 below). Finally the fact that the reranker makes more correct changes than mistakes is confirmed in this graph again, by observing that the number of *WR* cases is higher or equal than the number of *RW* cases at every position.

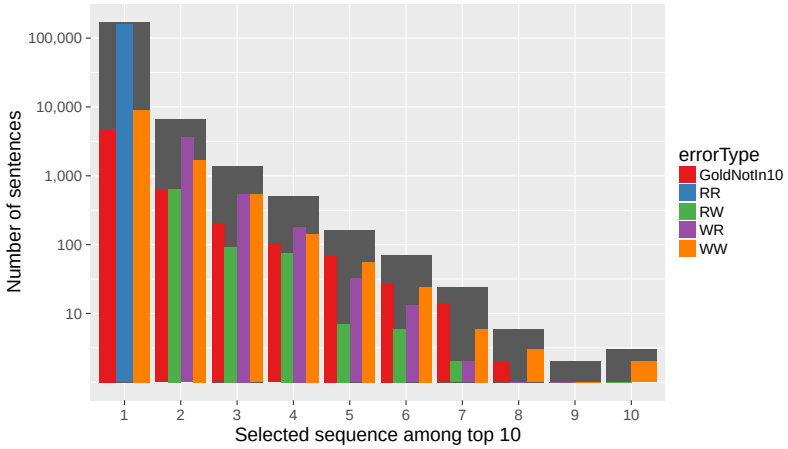

Figure 2: Distribution of the position selected by the reranker (logarithmic scale, all languages together). For every position, the large dark grey bar shows the total number of sentences, while the coloured bars show the number of sentences for every possible error type.[17]

The distribution of errors is not uniform over the data, and the number of expressions in a sentence is one of the most obvious factors: For example, Figure 3 shows that 96% of the sentences with no expression in the gold sequence are correctly identified by both the CRF and the reranker (*RR*), whereas only 9% of the sentences with three expressions are. The difference is mostly due to the proportion of sentences for which the CRF does not propose the right answer in the top 10 candidate sequences (*GoldNotIn10*), which is naturally higher in the more complex cases with multiple expressions in the sentence. Figure 3 also shows that the reranker is more useful with the sentences which contain one or two expressions (with 7.4% and 7.5% of changes, respectively), because these contain more mistakes to correct compared to sentences with no expressions, and contain more possibilities to correct the mistakes compared to sentences with three (or more) expressions (since the reranker cannot correct anything in the *GoldNotIn10* cases).

[17] This means that the dark grey bar represents the sum of the coloured bars, although the logarithmic scale makes this difficult to observe. Since the sequence selected by the CRF is the top one, position 1 is the only way for both the CRF and the reranker answers to be correct, thus it contains all the *RR* cases. Similarly, it cannot contain any *WR* or *RW* case, by definition.

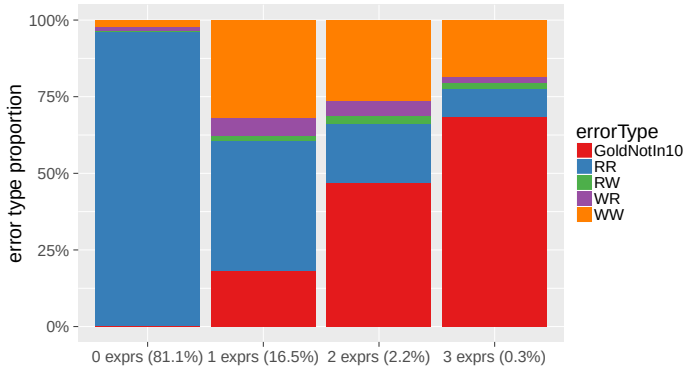

Figure 3: Proportion of error type by number of expressions in the gold sequence (all languages). Sentences with more than 3 expressions were discarded (30 cases, 0.1% of the data). Example: 16.5% of the sentences contain exactly one expression; among these, 18% belong to the *GoldNotIn10* category, and 41%, 1%, 6%, 32% belong respectively to categories *RR, RW, WR, WW*.

5.4 Reranker-specific evaluation

Based on these error types, a new reranker-centered evaluation method can be defined. Indeed, using this categorization, the reranker can be seen as a binary classifier: from this perspective, the job of the reranker is to detect the sentences for which the CRF answer was wrong, and leave the right ones as they are. Thus, for every sentence, either the answer is changed (positive instances) or not (negative instances). With this idea in mind, the four main categories can be translated to the standard true/false positive/negative categories in a straightforward way: if the reranker changes the answer correctly (**WR**), the instance is a true positive; if it changes the answer incorrectly (**RW**), the instance is a false positive, and similarly for the last two cases: **RR** and **WW** correspond respectively to true negative and false negative instances (the former was rightly not changed, and the latter should have been changed). Thanks to this interpretation, performance measures like precision, recall and F-score can be calculated for the semantic reranker, independently from the performance of the CRF component.[18] An example of using such performance scores is given in Table 5.

[18]The **GoldNotIn10** category is ignored when calculating these performance measures for the reranker, consistently with the idea of evaluating the reranker on its own: since these cases are impossible to solve, they should not be taken into account.

Table 5: Reranker-specific performance by number of expressions in the gold sequence (all languages). P/R/F stands for precision/recall/f-score; the macro-average performance is the average over languages (datasets with NaN F-scores are ignored).

Nb exprs	Macro-average			Micro-average		
	P	R	F	P	R	F
0	82.1	40.6	53.8	83.9	38.3	52.6
1	79.5	14.5	23.5	81.1	16.1	26.8
2	64.3	20.7	30.4	66.0	15.9	25.6
3	75.0	20.8	31.0	50.0	10.0	16.7
all	74.9	18.9	29.1	81.4	22.4	35.2

In the rest of this section we do not detail results by dataset, since the large number of languages and the dataset particularities would make it harder to recognize the general patterns related to the reranker. Nevertheless, it is important to keep in mind that such dataset-specific factors exist even though they are not shown. Additionally, the inequal size of the datasets clearly favors large datasets over small ones when grouping all the sentences together. This is why we present both the micro-average and macro-average performance whenever relevant, like in Table 5.

5.5 Feature analysis

As explained in §4, the reranker uses various kinds of features to determine the likelihood of the expressions in a sequence. Table 6 gives a brief overview of the impact of the different groups of features on performance, expressed as the F-score, computed from the micro-average precision and recall of the reranker alone (column 1, see §5.4), macro-average of the same over languages (column 2) and expression-level F-score (column 3, official evaluation on the full system).

First, it should be observed that the reranker relies heavily on the CRF confidence to make its decisions: without this feature, the performance drops to a ridiculously low level. Nevertheless, the reranker needs additional features in order to improve over the CRF alone (since otherwise the best it can do is to always agree with the CRF top prediction). A few simple features allow a large gain in performance (*SF* in Table 6: number of candidate expressions in the sequence, min./mean/max. number of words by expression and frequency in the

reference corpus). Adding more complex features based on frequency and semantic similarity of the candidate expression allows the reranker to make even better decisions: the micro F-score reaches 35.6 with the best combination, compared to 32.0 with only *SF*. Among these features, frequency and semantic similarity features seem to be equally useful, and combining both of them achieves the best performance; the only group of features which performs poorly (and is apparently even counter-productive) is the one where the candidate expression is compared against all other candidates in the sentence (group III in Table 6).

Table 6: Performance of the reranker using various subsets of features (percentages). Simple features *(SF)* represents the number of expressions and words as well as the frequency in the reference corpus; Groups *I*, *II* and *III* represent respectively *single word*, *expression minus one word* and *alternative expressions features*; Groups *a* and *b* represent respectively frequency features and semantic similarity features (see §4); NaN values correspond to cases where the precision and/or recall is zero.

Features	micro F-score reranker	macro F-score reranker	macro F-score MWE-level
baseline: CRF answer	NaN	NaN	48.2
all but confidence	00.6	NaN	09.6
confidence + SF (*)	32.0	NaN	53.1
(*) + Ia, Ib	34.9	29.7	53.4
(*) + IIa, IIb	34.3	29.0	53.4
(*) + IIIa, IIIb	33.2	27.8	53.6
(*) + IIa, IIb, IIIa, IIIb	34.4	29.1	53.7
(*) + Ia, Ib, IIIa, IIIb	34.2	29.0	53.4
(*) + Ia, Ib, IIa, IIb	**35.6**	**30.2**	53.7
freq. only: (*) + Ia, IIa, IIIa	33.9	28.7	53.7
sem. sim. only: (*) + Ib, IIb, IIIb	34.0	28.4	53.7
all features, with mean only	34.8	29.3	53.6
all features, with min/mean/max	35.2	30.1	**53.9**

5.6 Analysis: impact of the coverage in the reference corpus

Some candidate expressions might not be found in the reference corpus, either because they are simply rare or because of tokenization/lemmatization issues (see §4.4). In fact, the coverage rate of the expressions in Europarl is quite low: for 36.2% of the sentences containing at least one expression, the expression(s)

they contain are not found at all in the reference corpus. Figure 4 shows that the error type depends greatly on whether the expression appears in the reference corpus or not. First, the CRF finds the right answer much more often when the expression is covered ($RR + RW = 73\%$) than when it is not ($RR + RW = 30\%$). This can be explained by the fact that the least frequent expressions are hard to identify by the CRF, and they also tend not to appear in the reference corpus. While this implies that the reranker has potentially more mistakes to fix in the zero-coverage cases, it actually changes fewer sentences (4.4% against 12.5% for covered expressions), resulting in a very low recall (3.5% against 37.7%); the precision is also lower, with 67% against 81%.

As explained in §5.5, the reranker can work with only a small set of "simple features", which is why its performance in the zero-coverage case is lower but positive. Clearly, the more advanced features which rely on the reference corpus increase performance. This means that the coverage in the reference corpus is critical for the reranker to give its best results, but our current implementation of the system is probably not optimized from this point of view; in particular, the tokenization process might not be identical between the input data (where tokenization is provided) and the reference data (for which we apply a generic tokenizer), and the reference corpus is not lemmatized (see §4.4). This might explain why the recall is low with the current implementation. Ideally, a larger corpus would also help by covering a broader range of expressions; but there are very few such large datasets available for multiple languages.

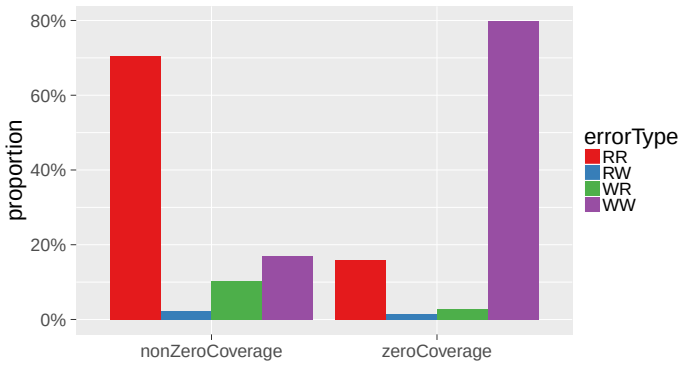

Figure 4: Sentences error types by coverage/non-coverage of their expressions in the reference corpus (all languages). Example: 70% of the sentences containing expressions which appear in the reference corpus belong to the *RR* category, whereas only 16% of the sentences with expressions not covered in the reference corpus belong to this category.

5.7 Analysis: continuous vs. discontinuous expressions

Verbal multiword expressions can be classified as either continuous or discontinuous: in the former case, the expression appears as a sequence of contiguous words, as in the following idiomatic expression:

(2) French (Indo-European; FR training data)
Celles-ci peuvent à tout moment jeter l' éponge.
they.3.FEM.PL can at any time throw the sponge
'They.3.FEM.PL can give up at any time.'

In the latter case, the expression appears with words inserted between its lexicalized components, e.g.:

(3) French (Indo-European; FR training data)
J'ai ***obtenu*** *de Jean-Marie Molitor […]* ***la permission*** *de publier.*
I have obtained from Jean-Marie Molitor […] the permission to publish
'Jean-Marie Molitor gave me the permission to publish.'

It is worth noticing the same lexicalized components might appear sometimes as a continuous expression and other times as a discontinuous expression.

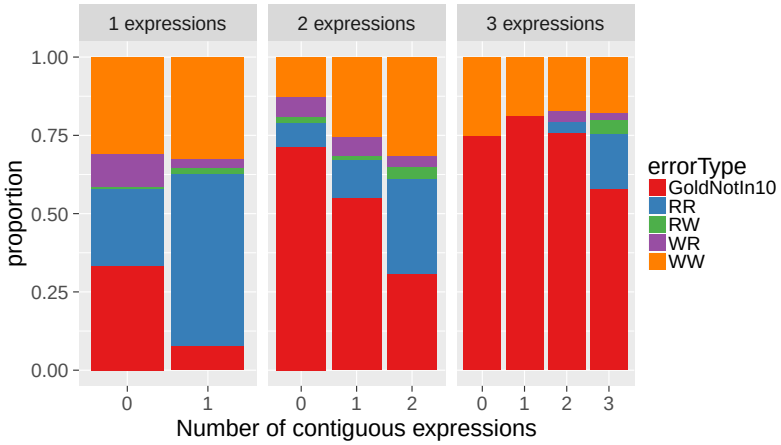

Figure 5: Proportion of error type by number of continuous expressions in the sentence, for sentences containing 1, 2 or 3 expressions (all languages). Example: among sentences which contain two expressions, the proportion of *RR* cases is 7% (respectively 12%, 30%), when there are no (resp. 1, 2) continuous expressions among the two.

Figure 5 shows the impact of continuity in expressions: *RR+RW* cases increase with the number of continuous expressions for any number of expressions in the sentence, which means that the more there are continuous expressions, the better the performance of the CRF (there are also much less *GoldNotIn10* cases). Interestingly, however, the semantic reranker follows an opposite trend: the less there are continuous expressions (i.e. the more there are discontinuous expressions), the better its performance: not only it fixes more mistakes from the CRF (better recall), it also fixes them better (better precision).[19] The most likely explanation for these observations is that the CRF suffers from a "sequential bias" which makes it less good with discontinuous expressions, whereas such cases are not any harder for the semantic reranker, which is "sequence-agnostic". In our opinion, this point clearly illustrates the complementarity of the two components.

5.8 Analysis: context vectors options

In §4.3, we presented several options which modify the way words which cooccur with the expression are taken into account in the MWE context vector. Table 7 shows the impact on performance of these options. Although there is no decisive pattern in these results, the absence of context-level normalization (CN)

[19]Except in the case of three expressions with zero or one continuous; this is probably due to the low number of cases.

Table 7: Performance of the reranker depending on context vector options. Left: Overall micro-average performance; right: F-score for continuous/discontinuous cases, for sentences with one expression exactly. CN, IWIE and MO represent the options presented in §4.3, respectively: *ContextNormalization, IncludeWordsInExpression, MultipleOccurrences*. P/R/F stands for precision/recall/f-score.

Options			Micro-average		
CN	IWIE	MO	P	R	F
0	0	0	82.9	21.9	34.6
0	0	1	81.7	**22.4**	**35.2**
0	1	0	82.5	22.1	34.8
0	1	1	81.6	21.7	34.3
1	0	0	**83.3**	21.8	34.5
1	0	1	82.8	21.7	34.4
1	1	0	80.9	21.8	34.4
1	1	1	81.4	**22.4**	**35.2**

Options			F-score	F-score
CN	IWIE	MO	Continuous	Discontinuous
0	0	0	14.7	**41.4**
0	0	1	**16.8**	40.3
0	1	0	14.7	40.9
0	1	1	15.9	39.6
1	0	0	15.2	39.8
1	0	1	15.2	39.6
1	1	0	14.0	41.0
1	1	1	14.8	40.8

as well as allowing multiple occurrences of the same word to be counted multiple times (MO) obtain slightly higher performance in general. Looking at the effect of these options on continuous/discontinuous expressions, including expressions words in the context window (IWIE) has a negative effect on all the cases, except if CN is selected but only in the discontinuous case. In fact, an interesting pattern can be observed in the continuous/discontinuous table: the combinations of options which make the F-score increase for continuous expressions tend to make the F-score in the discontinuous case decrease, and conversely. This is confirmed by a moderate negative Pearson's correlation coefficient of -0.56 and a high negative Spearman's rank correlation coefficient of -0.79. Here again the differences in performance are too moderate to conclude decisively; however this point suggests that there is a trade-off between the continuous and discontinuous cases, and this trade-off might be controlled through these options to some extent. This means that the system could potentially be tuned to favor one or the other case.

6 Conclusion and future work

In this chapter we described a two stages approach for identifying VMWEs, based on sequence labeling with CRF followed by reranking of the CRF candidates. We showed experimentally that the reranker significantly improves the performance of the system, with in average a 12% F1-score improvement over using the CRF component alone. Then we proceeded to analyze how the reranker works.

We found that the reranker follows the CRF quite closely, rarely selecting a candidate with a low CRF confidence, and selecting the CRF top prediction in 92.5% of the cases. Consistently with this observation, when the reranker diverges from the top CRF prediction, it does so correctly with high confidence (81.5% of correct answers among the changed predictions).

The contribution of the reranker is more important with the sentences which contain one or two expressions: sentences with no expressions are almost always correctly detected by the CRF alone, whereas the cases with 3 or more expressions are so complex that the CRF does not usually provide the right candidate among its top 10 predictions, leaving the reranker unable to fix these errors. The coverage of the MWE in the reference corpus is another major factor of performance for the reranker, with the recall dropping 10 times for expressions which do not appear in the reference corpus. Finally the last important finding of our study is that the reranker seems to compensate the CRF sequential bias: while the latter performs better with continuous MWEs, the reranker performs comparatively better with discontinuous cases.

The semantic reranker presented in this chapter is a proof-of-concept version, and new perspectives emerge from the fact that the combination of a CRF component with this reranker proves fruitful for detecting MWEs. There are a few obvious areas in which the reranker could be improved, especially in the tokenization/lemmatization part, and it is likely that choosing a more adequate reference corpus would help. But there are also deeper questions which are worth studying:

- Computing a context vector for a MWE is not as trivial as for a single word (especially if the words in the expression are not continuous), and the authors are not aware of any standard approach for this. While several options were tested, this question deserves to be studied on its own.
- In the same line of thought, the current state of the art in distributional semantics is based on word embeddings (Legrand & Collobert 2016). Here again, the authors are not aware of any software able to retrieve word embeddings for multiple (possibly discontinuous) words.
- Would it be possible for the reranker to work at the expression level instead of the sentence level? Indeed, the current method used to "merge" multiple expressions in a sentence is likely to lose some information in the process. One could also ask whether some of the information currently computed by the reranker could be fed directly into the CRF. An iterative process could even be considered, perhaps allowing to refine the quality of the predicted expressions over iterations.

Acknowledgements

We are grateful to the editors and anonymous reviewers for their valuable comments. Lifeng Han also thanks Qun Liu for his continued support.

The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

The graphics in this chapter were created with `R` (R Core Team 2012), using the `ggplot2` library (Wickham 2009).

Abbreviations

CRF	conditional random fields	RW	right-wrong
IDF	inverse document frequency	VMWE	verbal mutiword expression
MWE	multiword expression	WR	wrong-right
NLP	Natural Language Processing	WW	wrong-wrong
RR	right-right		

References

Attia, Mohammed, Lamia Tounsi, Pavel Pecina, Josef van Genabith & Antonio Toral. 2010. Automatic extraction of Arabic multiword expressions. In *Proceedings of the COLING Workshop on Multiword Expressions: From Theory to Applications* (MWE '10), 19–27. Association for Computational Linguistics.

Bar, Kfir, Mona Diab & Abdelati Hawwari. 2014. Arabic multiword expressions. In Nachum Dershowitz & Ephraim Nissan (eds.), *Language, culture, computation. Computational Linguistics and Linguistics: Essays dedicated to Yaacov Choueka on the occasion of his 75th birthday, Part III*, 64–81. Berlin: Springer. DOI:10.1007/978-3-642-45327-4_5

Blunsom, Phil & Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 164–171. Sydney, Australia: Association for Computational Linguistics. DOI:10.3115/1610075.1610101

Boukobza, Ram & Ari Rappoport. 2009. Multi-word expression identification using sentence surface features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 468–477. August 6-7, 2009.

Constant, Matthieu, Anthony Sigogne & Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 204–212. Association for Computational Linguistics.

Green, Spence, Marie-Catherine de Marneffe, John Bauer & Christopher D. Manning. 2011. Multiword expression identification with tree substitution grammars: A parsing tour de force with French. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 725–735. Association for Computational Linguistics. http://www.aclweb.org/anthology/D11-1067.

Green, Spence, Marie-Catherine de Marneffe & Christopher D. Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics* 39(1). 195–227. DOI:10.1162/COLI_a_00139

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann & Ian H. Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter* 11(1). 10–18.

Han, Aaron Li-Feng, Derek F. Wong & Lidia S. Chao. 2013. Chinese named entity recognition with conditional random fields in the light of Chinese characteristics. In Mieczysław A. Kłopotek, Jacek Koronacki, Małgorzata Marciniak, Agnieszka Mykowiecka & Sławomir T. Wierzchoń (eds.), *Language processing and intelligent information systems. iis 2013*, vol. 7912 (Language Processing and Intelligent Information Systems), 57–68. Berlin Heidelberg: Springer.

Han, Aaron Li-Feng, Xiaodong Zeng, Derek F. Wong & Lidia S. Chao. 2015. Chinese named entity recognition with Graph-based semi-supervised learning model. In *Proceedings of the 8th SIGHAN Workshop on Chinese Language Processing* (SIGHAN-8), 15–20. Association for Computational Linguistics & Asian Federation of Natural Language Processing. July 30-31, 2015.

Hosseini, Mohammad Javad, Noah A. Smith & Su-In Lee. 2016. UW-CSE at SemEval-2016 task 10: Detecting multiword expressions and supersenses using double-chained conditional random fields. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, 931–936. http://aclweb.org/anthology/S/S16/S16-1143.pdf. June 16-17, 2016.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, 79–86. Phuket, Thailand.

Lafferty, John D., Andrew McCallum & Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (ICML '01), 282–289. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=645530.655813.

Legrand, Joël & Ronan Collobert. 2016. Phrase representations for multiword expressions. In *Proceedings of the 12th Workshop on Multiword Expressions* (MWE '16), 67–71. Association for Computational Linguistics. http://anthology.aclweb.org/W16-1810.

Maldonado, Alfredo & Martin Emms. 2011. Measuring the compositionality of collocations via word co-occurrence vectors: Shared task system description. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, 48–53. http://dl.acm.org/citation.cfm?id=2043121.2043130.

Maldonado, Alfredo, Lifeng Han, Erwan Moreau, Ashjan Alsulaimani, Koel Dutta Chowdhury, Carl Vogel & Qun Liu. 2017. Detection of verbal multi-word expressions via conditional random fields with syntactic dependency features

and semantic re-ranking. In *Proceedings of the 13th Workshop on Multiword Expressions* (MWE '17), 114–120. Association for Computational Linguistics. DOI:10.18653/v1/W17-1715

Maldonado, Alfredo & Behrang QasemiZadeh. 2018. Analysis and Insights from the PARSEME Shared Task dataset. In Stella Markantonatou, Carlos Ramisch, Agata Savary & Veronika Vincze (eds.), *Multiword expressions at length and in depth: Extended papers from the MWE 2017 workshop*, 149–175. Berlin: Language Science Press. DOI:10.5281/zenodo.1469557

Quinlan, J. Ross. 1992. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, 343–348.

R Core Team. 2012. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/. ISBN 3-900051-07-0.

Sag, Ivan A., Timothy Baldwin, Francis Bond, Ann A. Copestake & Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Computational Linguistics and Intelligent Text Processing*, vol. 2276/2010 (CICLing '02), 1–15. Springer-Verlag.

Salehi, Bahar, Paul Cook & Timothy Baldwin. 2014a. Detecting non-compositional MWE components using Wiktionary. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (EMNLP 2014), 1792–1797. Doha, Qatar. http://aclweb.org/anthology/D/D14/D14-1189.pdf.

Salehi, Bahar, Paul Cook & Timothy Baldwin. 2014b. Using distributional similarity of multi-way translations to predict multiword expression compositionality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics* (EACL 2014), 472–481. Gothenburg. http://aclweb.org/anthology/E/E14/E14-1050.pdf.

Salehi, Bahar, Paul Cook & Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies* (NAACL '15), 977–983. http://dblp.uni-trier.de/db/conf/naacl/naacl2015.html#SalehiCB15.

Savary, Agata, Carlos Ramisch, Silvio Cordeiro, Federico Sangati, Veronika Vincze, Behrang QasemiZadeh, Marie Candito, Fabienne Cap, Voula Giouli, Ivelina Stoyanova & Antoine Doucet. 2017. The PARSEME Shared Task on automatic identification of verbal multiword expressions. In *Proceedings of the 13th Workshop on Multiword Expressions* (MWE '17), 31–47. Association for Computational Linguistics. DOI:10.18653/v1/W17-1704

Schneider, Nathan, Emily Danchik, Chris Dyer & Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics* 2(1). 193–206. https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/281.

Schütze, Hinrich. 1998. Automatic word sense discrimination. *Computational Linguistics* 24(1). 97–123. http://dl.acm.org/citation.cfm?id=972719.972724.

Sinclair, John. 1991. *Corpus, concordance, collocation*. Oxford: Oxford University Press.

Sun, Xiao, Chengcheng Li, Chenyi Tang & Fuji Ren. 2013. Mining semantic orientation of multiword expression from Chinese microblogging with Discriminative Latent Model. In *Proceedings of the 2013 International Conference on Asian Language Processing*, 117–120. http://dblp.uni-trier.de/db/conf/ialp/ialp2013.html#SunLTR13.

Tsvetkov, Yulia & Shuly Wintner. 2010. Extraction of multi-word expressions from small parallel corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics* (COLING-10), 1256–1264. http://dl.acm.org/citation.cfm?id=1944566.1944710.

Venkatapathy, Sriram & Aravind K. Joshi. 2006. Using information about multiword expressions for the word-alignment task. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties* (MWE '06), 20–27. http://dl.acm.org/citation.cfm?id=1613692.1613697.

Vincze, Veronika, István Nagy T. & Gábor Berend. 2011. Multiword expressions and named entities in the Wiki50 corpus. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, 289–295. RANLP 2011 Organising Committee. http://aclweb.org/anthology/R11-1040.

Wickham, Hadley. 2009. *Ggplot2. Elegant graphics for data analysis*. New York: Springer-Verlag.