VectorWeavers at SemEval-2016 Task 10: From Incremental Meaning to Semantic Unit (phrase by phrase)

Andreas Scherbakov **Ekaterina Vylomova** **Fei Liu** **Timothy Baldwin**
andreas@softwareengineer.pro, evylomova@gmail.com,
fliu3@student.unimelb.edu.au, tb@ldwin.net
The University of Melbourne
VIC 3010, Australia

Abstract

This paper describes an experimental approach to Detection of Minimal Semantic Units and their Meaning (DiMSUM), explored within the framework of SemEval'16 Task 10. The approach is primarily based on a combination of word embeddings and parser-based features, and employs unidirectional incremental computation of compositional embeddings for multiword expressions.

1 Motivation

This paper proposes an approach to the segmentation of POS-tagged English sentences into minimal semantic units along with labelling of units with semantic classes ("supersenses"). Supersenses are "lightweight semantic annotation[s] of text originating in WordNet" (Schneider et al., 2012). Here, we investigate two major ideas, as follows.

First, inspired by Salehi et al. (2015) we hypothesise that word embeddings (WEMBs), e.g. from word2vec (Mikolov et al., 2013a), could be an extremely valuable resource of knowledge for guessing the sense of the word. WEMBs have been shown to represent distinguishable sense components learnt from large training corpora. Many papers have described experiments with word meaning extraction from word embeddings, and demonstrated that it's possible to detect semantic relations between words based on them. Additionally, they may be used to simulate various types of relations between words with simple linear operations over word vectors, and for capturing morphological properties of words (Mikolov et al., 2013b; Vylomova et al., 2015). However, more generally, they lack any coherent sense-to-value correspondence. According to this, it's natural to attempt to use WEMBs as a source of word meaning information in predicting the senses of semantic units, as part of the larger task of simultaneously detecting minimal semantic units and assigning them meaning. It should be noted that the general term "multiword expression" (MWE) embraces expressions of different types (Baldwin and Kim, 2010). Some of them are meaning-based and the meaning may be preserved under substitution with synonyms, e.g.; other kinds, on the other hand, are semantically idiomatic word combinations, where we would expect WEMBs to have less utility. This first approach is thus an examination of the use of WEMBs in analysing semantic units of mixed semantic compositionality.

Our second hypothesis is that parser-based features will positively impact on the identification of MWEs. Our preliminary explorations showed that the syntactic structure of a text is closely related to the probability of an MWE occurring. For instance, in almost all cases an MWE is fully subsumed within a single clause; in the DiMSUM training set, e.g., there is just one sentence (out of 4800) where this condition is violated. Additionally, all of the components of an MWE tend to be directly connected within a dependency graph. As we observe strong correlation between the distance between two words in a parse tree and their likelihood of forming an MWE, we decided to employ parse trees as a source of features. It may be noted that we initially attempted to create a system where an MWE is treated as a special kind of clause: we modified

Proceedings of SemEval-2016, pages 946–952,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

the syntactic tree using knowledge of MWE boundaries in the training corpus and then trained a special version of the parser aware of such modified trees. That special parser version was used to directly produce MWE-labeled clauses over an arbitrary text. Although such a direct approach didn't yield good results, we believe that directly incorporating MWE identification as part of the parsing process is a promising and fruitful direction for future work.

In our submission, we intentionally avoided the use of any pre-existing lexical resources, including multiword lexicons, to better focus our attention on WEmbs.

Our source code is available at: `https://github.com/andreas-softwareengineer-pro/dimsum-semeval2016`.

2 Overview

An overall architecture of our system is shown in Figure 1. The neural network consists of three blocks: (1) an *incremental vector (recurrency)* calculator for a candidate MWE; (2) a two-layer perceptron for sense classification; and (3) a (somewhat wider) two-layer perceptron for MWE classification. An incremental vector produced by the first block is used as a source of feature vectors by each of the latter blocks.

The incremental vector calculator produces a vector $v_n \in \mathbb{R}^{D_v}$ for a n-word semantic unit expression:

$$v_n = \tanh \left(\begin{array}{l} W_v(P_n) \times wordvec(w_n) + \\ W_h(P_n) \times hash(w_n) + \\ W_f \times wordfeat(w_n) + \\ W_c(P_n) \times \left\{ \begin{array}{ll} v_{n-1}, & n > 0 \\ seed, & n = 0 \end{array} \right\} \end{array} \right)$$

where $W_v \in \mathbb{R}^{D_v \times M}$, $W_h \in \mathbb{R}^{D_h \times M}$, $W_f \in \mathbb{R}^{D_f \times M}$ and $W_c \in \mathbb{R}^{D_c \times M}$ are parameter matrices, D_v, D_h, D_f and D_c are their respective feature vector sizes, M is the incremental vector size, w_n and P_n are the nth word and its part of speech, respectively, $wordvec$ is a word embedding lookup, $hash$ is a hash function over characters of the word (used as a means of generating embeddings for unknown words and preserve the ability to distinguish concrete words), and $wordfeat$ produces a vector of various word-wise features (see Table 2), in the spirit of approaches like Drahomíra johanka Spoustová et al. (2009). When calculating v_1 (for a single word), an initial $seed$ vector of parameters learnt through backpropagation is used. The motivation behind using the $n = 1$ stage for every MWE (rather than simply starting with $n = 2$ and two word vectors) is based on an intention to avoid switching between word embeddings (that play the role of features) and internally calculated v_i that we expect to better capture the structure of the MWE based on the WEmbs (including following their dimension counts). Also, this technique makes the vector composition learning cycle more frequent w.r.t. the amount of training data, improving the learning rate and the final penalty. This evaluation schema is inspired by the work of Socher et al. (2011; 2010; 2014). It's actually a recursive neural network (RNN) but, in contrast to previously used techniques, the recursion is based on candidate MWEs rather than the whole content of the sentence.

The incremental vector is used as an input to two two-layer perceptrons: one for MWE classification and one for one-hot sense vector learning. Back propagation processing of these two also drives the training of the incremental vector calculator, as described above.

We calculate distance-based features based on positions of two adjacent words in an MWE. A *position* here may mean a position in a sentence, in a parse tree, or, say, inside or outside a quoted phrase. The distance feature vector is supplied as input to the MWE classifier. As an alternative, it may be supplied to the incremental vector calculator input (shown by the dotted line at Figure 1, although no significant difference in performance was observed when we did this).

3 Learning

As briefly mentioned above, our procedure considers n-word expressions incrementally, starting with single words.

3.1 MWE boundary

The system learns a single scalar output value of +1 for every extension from $(n-1)$-word prefix to a n-

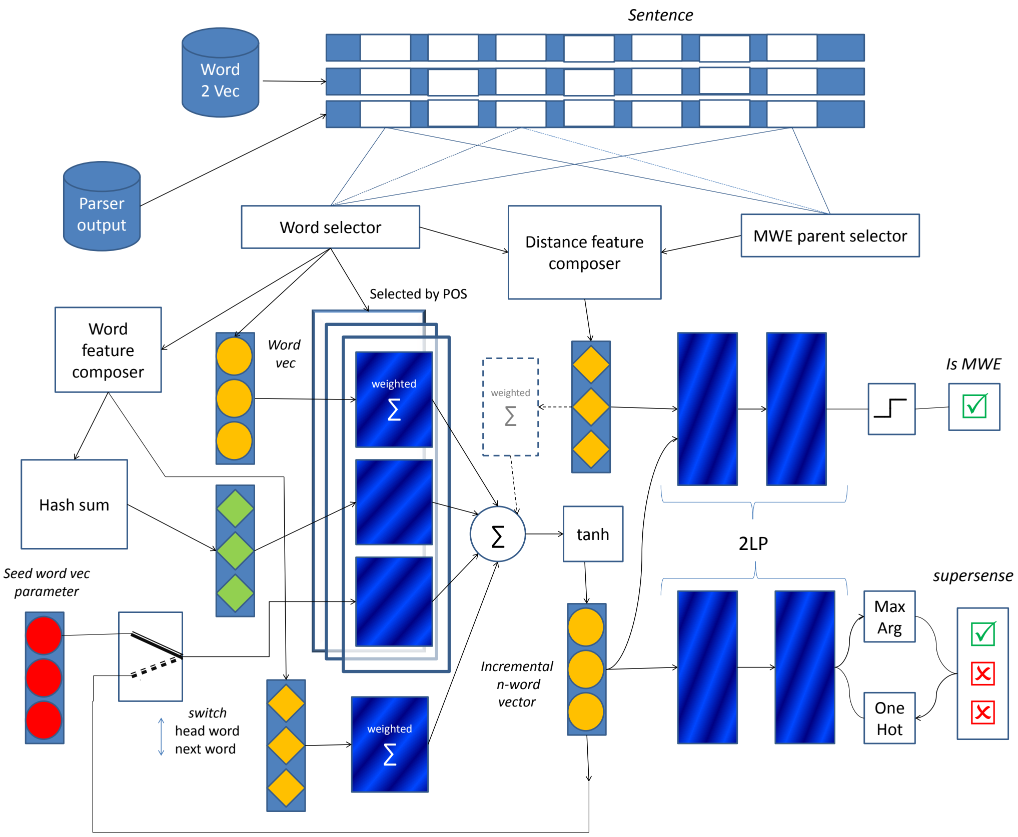

Figure 1: An outline of the system architecture

word MWE (either complete or incomplete), where $n \geq 2$. In such a way, every n-word expression finally yields $n-1$ incremental positive samples.

Also, for every word, regardless of whether it is a standalone word or a member of MWE, it learns a -1 value (i.e. a negative sample) for its (unobserved) extension to an MWE with a random word. Such a random word is chosen of the L words ahead in the same sentence, not involved in the same MWE with the current word.[1]

During the learning process (and, correspondingly, during the prediction), recursive computations of the incremental vector are solely fed by members of the same MWE, and are restarted for each semantic unit.

3.2 Senses

We learn a sense once per complete n-word expression, including $n = 1$. We have a distinct output per sense, including "unknown" (42 in total in the DiMSUM data set). We factorise sense classification error by the number of senses in order to balance the backpropagation of two perceptrons to the incremental vector calculator.

4 Prediction

We use a greedy procedure for MWE prediction in a text. An *outer* loop iterates through all words of a sentence. It selects each word (not yet consumed by some previously predicted MWE) as the head

[1] We limit L to 9 words, and also limit the probability of every word selection to < 0.25 (by randomly down-weighting the sample processing for $L < 4$) in order to prevent possible bias to the distribution of inter-word offsets occurring while processing the last few words of a sentence. We solely pick the same sentence words as negative samples in order to make learning more focused on prediction environment patterns rather than noise, as we use a small training corpus, and at the same time to keep an approximate balance in count between negative and positive samples.

word of a possible new MWE and restarts the incremental vector computation. An *inner* loop iterates through (up to L) remaining words in the sentence following the current head word. Each time, a probability for such a second word to be a continuation of the MWE is evaluated. Once a new MWE or an MWE extension is predicted, it consumes the right hand word and designates it to be the next word of the MWE; the incremental vector is updated respectively with the new MWE member. Such a procedure is able to generate a deep stack of nested MWEs with gaps, but we restrict the depth to be compliant to the DiMSUM data format.

5 Features

5.1 Word Embeddings

We utilised a publicly available pre-trained CBOW word2vec model, with 300-dimensional word vectors based on the Google News corpus. Out-of-vocabulary words are represented with zero-filled vectors.

We use the following search strategy when looking up a word in the word2vec dictionary (until the first successful lookup returns the final embedding):[2].

1. strip off leading "#" and "@"
2. if the word is a number then replace by "NUM"
3. lowercase the word
4. (optionally) lemmatize
5. remove all non-alpha characters
6. return the word embedding associated with the word OR if no match, return a zero vector

5.2 Word Hash Sum

We produce a 64-dimensional $+1/-1$ hash sum vector for words where the embedding vector is unknown; a zero vector was supplied for words (somehow) found in the Google News WEmb database. Only alphabetic characters were counted in the hash sum.

Feature	Range	Description
`Gap`	$\frac{n}{8}$,$n \in Z_{\geq 0}$	gap between word positions (divided by 8)
`Par`$_{\texttt{Dist}}$	$\{2; 0; -\frac{3}{2}\}$	hierarchical distance between words in a parse tree; three gradations for distances of 1,2,$\geq$ 3
`Par`$_{\texttt{Parent}}$	$\{-1, 1\}$	two words are head word and child word of the same clause, according to the parser output
`InterQt`	$\{-1, 1\}$	double-quotes anywhere between given words?

Table 1: Distance features

5.3 Word Distance features

Table 1 displays inter-word distance-based features.

We used a parser prediction file created for the source text in order to evaluate a hierarchical distance between two candidate words. The hierarchical distance here means the maximum of two counts of edges that connect two given words to the nearest clause they share. For instance, two sibling nodes have the hierarchical distance value of one. The same is also true of a phrase head word and its immediate dependent word (that case is indicated by a distinct feature). We employed TurboParser v.2.0.0 (Martins and Almeida, 2014) and trained it on a Penn TreeBank data collection (some conversion was needed to match with the part-of-speech tagset used in the DiMSUM task).

5.4 Heuristic word features

The full list of miscellaneous word features (concerning capitalization, punctuation characters, lookup success etc.) is presented in Table 2.

6 Results

The system configuration which was used as our official run for the SemEval 2016 task was parameterized as follows: wide layer 1 in MWE prediction perceptron = 1024 nodes; hash sum size = 16bit, calculated both for known and unknown word, using all characters in a word (alpha + non-alpha); a

[2] We tried using light spelling correction to reduce the rate of unknown words (a simple substitution of one character and then selecting the most probable word according to its frequency ranking). However it didn't seem to be effective, as the number of mispredicted words was greater then the number of correctly predicted ones, in particular due to the absence of a great number of frequent words ("stop-words") in the Google News vectors DB. Thus, a more sophisticated system would be needed if one wants to correct typos.

Feature	Value range	Description
$\texttt{Cap}_{\texttt{First}}$	$\{-1, 1\}$	Word starts with a capital letter
$\texttt{Cap}_{\texttt{Norm}}$	$[0, 1]$	Word starts with a capital letter, a value normalized by dividing by the sentence word count
$\texttt{Cap}_{\texttt{Ratio}}$	$[0, 1]$	Ratio of uppercase letters in the word
$\texttt{Has}_{\texttt{NAlpha}}$	$\{-1, 1\}$	Word has non-alpha characters
$\texttt{Has}_{\texttt{Num}}$	$\{-1, 1\}$	Word has any digit inside
$\texttt{Has}_{\texttt{Prime}}$	$\{-1, 1\}$	Word has an apostrophe
$\texttt{Is}_{\texttt{At}}$	$\{-1, 1\}$	Word starts with "@"
$\texttt{Is}_{\texttt{Hash}}$	$\{-1, 1\}$	Word starts with "#"
$\texttt{Is}_{\texttt{Num}}$	$\{-1, 1\}$	Word is num (integer, float or "NUM" keyword)
$\texttt{Is}_{\texttt{Punct}}$	$\{-1, 1\}$	Word contains punctuation character(s) (one or more of "!?.,;:[]()/")
$\texttt{Is}_{\texttt{Unk}}$	$\{-1, 1\}$	word2vec out-of-vocabulary word, including stop words
$\texttt{Is}_{\texttt{Url}}$	$\{-1, 1\}$	Word is "URL"
$\texttt{Log}_{\texttt{Range}}$	$[0, 12]$	Smoothed logarithm of the word's frequency range found in the word2vec dictionary, $\log(0.1 \times \text{range} + 1)$; the more frequent the word, the lower the value
$\texttt{Quot}_{\texttt{Pre}}$	$\{-1, 1\}$	A double-quote is located at the previous word position
$\texttt{Quot}_{\texttt{Post}}$	$\{-1, 1\}$	A double-quote is found at the next word position

Table 2: Word Features

Type	Prec	Recall	F1
Experiments			
MWEs	0.4697	0.4655	46.76%
Supersenses	0.5320	0.5138	52.27%
Combined	0.5194	0.5046	51.19%
Official			
MWE	0.6122	0.2807	38.49%
Supersenses	0.5009	0.5326	51.62%
Combined	0.5114	0.4846	49.77%
Macro			49.94%

Table 3: Results

mean vector of word embeddings over the sentence was included as extra feature; distance features are supplied to the composed vector evaluator (as shown with the dotted line in Figure 1); a confidence level of -0.15 was applied to the MWE perceptron when predicting a two-word MWE prefix (but not when expanding it to the third and next words, encouraging an MWE to start). The whole DiMSUM training set was used to train the system.

Table 3 displays the results, measured with the DiMSUM evaluation script.

Table 4 represents a brief ablation study with feature vector disabled. Word embeddings seem to be the critical source of supersense information, and they are also one of the important contributors to MWE recall. The incremental n-word vector is critical for MWE precision, and Distance features are of great importance both for precision and recall in MWE detection, but especially for the recall. Word hash and, surprisingly, heuristic word features are of much less significance than other feature vectors.

7 Findings & Conclusions

The system captures supersenses rather well (when we consider the large number of senses, small size of the training data, and ambiguity in the sense assignments). However, it frequently admits harsh mispredictions that are probably caused by the lack of global sense coherence in WEMBS (Qu et al., 2015). Also, it suffers from the lack of (other than MWE-related) context information in cases of ambiguity. Improving context awareness may be the most obvious next step. Unsurprisingly, the most frequent supersenses have the best recall, up to 80% for V.STATIVE. The mean recall value for senses is around 50–60%, and there are senses (like N.WEATHER) that are never correctly predicted. Exceptions are N.ATTRIBUTE, N.LOCATION, and V.CHANGE, where recall is below 30% despite them being reasonably frequent. Some of the most frequent mispredictions are the following: N.PERSON → N.GROUP; N.COMMUNICATION →

Ablated features	MWEs			Supersenses			Combined		
	Prec	Recall	F1	Prec	Recall	F1	Prec	Recall	F1
−Recurrency	0.2125	0.4960	29.75%	0.4638	0.3581	40.41%	0.3591	0.3843	37.13%
−Heuristic	0.4230	0.5821	48.99%	0.5033	0.4797	49.12%	0.4822	0.4991	49.05%
−Distance	0.3246	0.2771	29.90%	0.4494	0.4575	45.34%	0.4281	0.4232	42.56%
−Word hash	0.3640	0.6350	46.27%	0.5133	0.4811	49.67%	0.4674	0.5104	48.80%
−word2vec	0.3204	0.5193	39.63%	0.1898	0.1766	18.29%	0.2293	0.2418	23.54%

Table 4: Feature ablation results

N.ARTEFACT; N.ATTRIBUT → N.COGNITION; N.ACT → N.EVENT; and V.EMOTION → V.COGNITION.

MWE identification looks generally reasonable for all principal types of MWEs (even, surprisingly, ones of an idiomatic nature), but the overall accuracy is low.

Hysteresis bias pattern. The MWE classifier tends to be biased toward *not* joining two words into a MWE[3]; at the same time, once an MWE is predicted, it tends to be extended excessively to include a third and subsequent words, often selecting a non-relevant word with some gap.[4] This probably means that the proposed sampling schema is not balanced enough to work for small amounts of training data.

Lookahead. The method obviously lacks the ability to model context when predicting MWEs. In cases of expressions like *John , Mary & Company*, for example, our system will predict *John Mary Company* with all the punctuation missing. Some bidirectional, attentional (Bahdanau et al., 2014; Cohn et al., 2016) or lookahead-based approach is needed, as we may not have a reasonable isolated rule for whether *John* should be joined to the comma. A similar situation may also occur in punctuation-less contexts.

Parsing. The use of parsing results in markedly better precision and recall. Taking into account the strong correlation between parser-based features and MWE boundaries, further investigation of the parser-based approach is warranted.

Multiword-to-Sense Collision. A shared n-word incremental vector computation both for MWE and sense learning imposes a collision. It may be observed that for better MWE training, the distance features should be supplied to the computation input (as shown with the dotted line in Figure 1), but this will decreases the sense prediction score.

Needs a deeper network. The training dynamics observed in the experiments show that the neural networks (especially the one used for MWE prediction) need to be deeper (use more than two layers, as suggested in Sutskever et al. (2014)).

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Baldwin and Kim2010] Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*. CRC Press, Boca Raton, USA, 2nd edition.

[Cohn et al.2016] Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085*.

[Drahomíra johanka Spoustová et al.2009] Jan Drahomíra johanka Spoustová, Hajič, Jan Raab, Miroslav Spousta, et al. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771. Association for Computational Linguistics.

[Martins and Almeida2014] André FT Martins and Mariana SC Almeida. 2014. Priberam: A turbo semantic parser with second order features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 471–476.

[3] An example from the DiMSUM test data: *English speaking advisor* produces an MWE predicted to be *English advisor* (not including *speaking*

[4] As a very rough compensation of such an effect, one may use two confidence levels, one (negative) at $n = 2$, another (positive) at $n \geq 3$.

[Mikolov et al.2013a] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mikolov et al.2013b] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 746–751, Atlanta, USA.

[Qu et al.2015] Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. Big data small data, in domain out-of domain, known word unknown word: The impact of word representation on sequence labelling tasks. *arXiv preprint arXiv:1504.05319*.

[Salehi et al.2015] Bahar Salehi, Paul Cook, and 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*, pages 977–983.

[Schneider et al.2012] Nathan Schneider, Behrang Mohit, Kemal Oflazer, and Noah A Smith. 2012. Coarse lexical semantic annotation with supersenses: an arabic case study. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 253–258. Association for Computational Linguistics.

[Socher et al.2010] Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

[Socher et al.2011] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

[Socher2014] Richard Socher. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Ph.D. thesis, Citeseer.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Vylomova et al.2015] Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2015. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. *CoRR*, abs/1509.01692.