Discovery Team at SemEval-2020 Task 1: Context-sensitive Embeddings not Always Better Than Static for Semantic Change Detection

Matej Martinc *
Jozef Stefan Institute, Slovenia
matej.martinc@ijs.si

Syrielle Montariol
Univ. Paris-Saclay, Societé Générale
LIMSI - CNRS, Univ. Paris-Sud, France
syrielle.montariol@limsi.fr

Lidia Pivovarova
University of Helsinki, Finland
lidia.pivovarova@helsinki.fi

Elaine Zosa
University of Helsinki, Finland
elaine.zosa@helsinki.fi

Abstract

This paper describes the approaches used by the Discovery Team to solve SemEval-2020 Task 1 - Unsupervised Lexical Semantic Change Detection. The proposed method is based on clustering of BERT contextual embeddings, followed by a comparison of cluster distributions across time. The best results were obtained by an ensemble of this method and static Word2Vec embeddings. According to the official results, our approach proved the best for Latin in Subtask 2.

1 Introduction

Each word has a variety of senses and connotations, constantly evolving through usage in social interactions and changes in cultural and social practices. Identifying and understanding these changes is important for linguistic research and social analysis, since it allows the detection of cultural and linguistic trends and possibly predict future changes. Detecting these changes can also be used to improve many NLP tasks, such as text classification and information retrieval.

The SemEval-2020 Task 1 — Unsupervised Lexical Semantic Change Detection (Schlechtweg et al., 2020) deals with detection of semantic change in temporal corpora containing texts from two distinct time periods in four languages: English, German, Latin and Swedish. The challenge defines two subtasks: Subtask 1 is binary classification, i.e., to determine whether a word has changed or not; SubTask 2 aims at ranking a set of target words according to their rate of semantic change.

In this paper, we present the approaches used by the Discovery Team to tackle these two subtasks. The Discovery Team qualified as 11th and 5th on Subtasks 1 and 2, respectively, and also proved the best for Latin language in Subtask 2. Our systems leverage the transformer-based BERT model to generate contextualised embeddings for each word usage. Then these embeddings are aggregated into meaningful time-specific word representations. We explore different aggregation techniques, such as clustering (k-means and affinity propagation) and averaging. We also combine BERT-based representations with static Word2Vec embeddings[1].

2 System Overview

2.1 Word Representation

In order to derive meaningful temporal representations for each target word, we adapted the methodology proposed in Martinc et al. (2020a) to the multilingual setting of the SemEval-2020 Task 1. The core component of our approach is the use of BERT (Bidirectional Encoder Representations from Transformers), a pretrained masked language model based on the transformer architecture (Devlin et al., 2019). We use specific models for each language—for English: bert-base-uncased model, for Swedish: bert-base-swedish-uncased (https://github.com/af-ai-center/SweBERT), for German: bert-base-german-cased (https://deepset.ai/german-bert), for Latin: bert-base-multilingual-uncased

*All authors contributed equally to this research.

This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: http://creativecommons.org/licenses/by/4.0/.

[1]Code for the experiments is available under the MIT license at https://github.com/smontariol/Semeval2020-Task1.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 67–73
Barcelona, Spain (Online), December 12, 2020.

model—all with 12 attention layers and a hidden layer of size 768. German is the only language for which we use a cased model since most target words are nouns, which are capitalized in German. The only model available for Latin is a multilingual BERT model trained on 104 languages, including Latin.

For each language, the model is fine-tuned for five epochs on the task's corpus, as advised by Martinc et al. (2020a). This fine-tuning is unsupervised, i.e., a masked language model objective is used in the fine-tuning step (Devlin et al., 2019) in order to adapt each model to a specific corpus.

The fine-tuned models are used to generate token embeddings. The corpus for each language is split into two periods and the fine-tuned models are fed with sentences containing one or more target words from the sub-corpus. The sentences are split into tokens, and an embedding of dimension 768 is generated for each token by summing the last four encoder output layers of BERT, as advised by recent studies which confirm the fact that semantic features are captured in higher layers of BERT (Jawahar et al., 2019).

Note that byte-pair tokenization (Kudo and Richardson, 2018) in some cases generates tokens that correspond to sub-parts of words. To generate embedding representations for the target words split into sub-parts, we concatenate the embeddings of each byte-pair token constituting a word. After this procedure, we obtain a contextual embedding representation for each target-word usage, together with the time period each word usage representation belongs to.

In addition to context-dependent embeddings, we generate static word representations by training a 300-dimensional Word2Vec model using the skip-gram architecture with negative sampling (SGNS) (Mikolov et al., 2013) for each time slice. We align the embeddings from the different time slices using the Orthogonal Procrustes (OP) method as in Hamilton et al. (2016). We also applied pre- and post-processing steps such as mean-centering and vector normalisation, as recommended in Schlechtweg et al. (2019) [2].

2.2 Measures of Semantic Change

We use two methods to aggregate contextual embeddings from BERT: averaging and clustering. The methods were introduced and compared in our previous work (Martinc et al., 2020a; Martinc et al., 2020b).

Averaging is a simple aggregation approach where all target-word usage representations from a given time period are averaged. A quantitative estimate of semantic change for each target word is measured by computing the cosine distance between two averaged time-specific representations of the word.

Clustering of word usage representations results in sets of word usages, where each set is expected to correspond to a single word sense or a specific context. We create two time-specific cluster distributions by counting the number of cluster members for each time period and creating a vector of cluster counts for each cluster within a time period. Then the Jensen-Shannon divergence (JSD) between two period-specific distributions is used to measure the semantic change, as in Martinc et al. (2020a).

We use two clustering techniques, *affinity propagation* (Frey and Dueck, 2007) and *k-means*[3]. Affinity propagation has been extensively used in the literature for semantic tasks such as word sense induction (Alagić et al., 2018). It works by exchanging messages between data points until a high-quality set of *exemplars*, i.e. members of the input set that are representative of clusters, is obtained. A big advantage of this method is that it considers all the data points as potential cluster centers and therefore does not require the number of clusters to be defined in advance. K-means is a very popular clustering method and has been shown to perform well for the semantic change detection task (Giulianelli et al., 2020). Contrarily to affinity propagation, it requires to define the number of clusters in advance. We try several values of k; the highest accuracy for this task is obtained with $k = 5$.

To obtain a measure of semantic change using static embeddings, we measure the cosine distance between the aligned embedding representations of the same word from two time slices.

2.3 Subtask 1: Binary Classification

In order to determine whether a target word has changed or not, we experiment with two distinct methods, *thresholding using stopwords* and *identification of period-specific clusters*.

[2]We trained Word2Vec using the code from `https://github.com/Garrafao/LSCDetection`

[3]We use the Scikit-learn implementations (`https://scikit-learn.org/stable/modules/clustering.html`) with default parameters, except for the number of clusters for k-means, for which we tried several options.

		English	Latin	Swedish	German
Number of stopwords		109	334	78	142
Mean JSD	stopwords	0.181	0.210	0.355	0.328
	targets	0.239	0.264	0.460	0.384

Table 1: Number of stopwords used and average semantic change score (JSD) for target words and stopwords.

2.3.1 Thresholding Using Stopwords

We want to find the best threshold in the ranked list of target words by relying on the assumption that stopwords—words that are very frequent in a language and play primarily auxiliary roles—undergo a low semantic change.

Though stopwords are more stable than most words of the dictionary, they can still change their meaning due to the grammaticalisation processes, i.e. when a previously meaningful word looses most of it functions except for auxiliary ones. For example, the English stopword *hence* used to have a concrete deictic meaning "from here" (e.g. "hence we go") but nowadays it is used only to connect two propositions. Since not all stopwords are stable, finding an appropriate threshold is not straightforward.

It should be noted that stopwords have extremely context-specific representations (Ethayarajh, 2019). However, high polysemy and highly variable context do not necessarily induce more semantic change (Martinc et al., 2020a). We check the difference of average semantic change between a set of stopwords and the list of target words for all languages.

First, to compute semantic change scores for a list of stopwords, we use the same procedure that was used for the target words. For all languages except Latin, we create a list of stopwords by taking the words at the intersection of the `nltk` and `Spacy` stopword lists. For Latin, we use an external resource[4]. We keep only stopwords with more than 30 occurrences in each period; the number of stopwords per language is shown in Table 1. When the number of occurrences of a word is too high, we sample 5000 sentences per period for this word. As can be seen in Table 1, the mean JSD for stopwords is sensibly lower than the one for target words.

Then, we compare stopword and target word score distributions in order to define a threshold below which a target word should be classified as unchanged.

We first divide the stopwords' semantic change score distribution into 10 bins to derive a frequency distribution in a shape of a histogram with 10 columns, as exemplified for English in Figure 1. We take the threshold as the local maximum score of the bin in the histogram containing a number of words lower than an epsilon ϵ. We exclude the first bin, which is composed of very stable words and can sometimes have a size smaller than ϵ. The frequency limit ϵ used to select the threshold depends on the number of stopwords for each language: $\epsilon = 1/10 * \textit{number-of-stopwords}$. We compute two sets of thresholds: the leftmost and the rightmost points of the border bin, as shown in the Figure 1. The higher threshold is more conservative, meaning that fewer words are classified as changed.

2.3.2 Identification of Period-Specific Clusters

The second method looks for concrete indications of semantic change, such as the appearance or disappearance of a specific word sense. Target word clusters should to some extent resemble different word senses, allowing identification of target words that obtained or lost a meaning. If one of the clusters for a target word contains word occurrences from one time period and contains less or equal than k (where k=2) word occurrences from another time period, we assume that this word has lost or gained a specific meaning.

Since clustering methods sometimes produce small-sized clusters, we consider only the clusters bigger than a threshold, in order to focus on the "main" usages of a word. Thus, for k-means we enforce a constraint that a cluster should contain at least 10 word occurrences to be considered in the analysis. For affinity propagation, we implement a dynamic threshold strategy: the threshold beyond which we consider

[4]List of Latin stopwords: https://github.com/aurelberra/stopwords

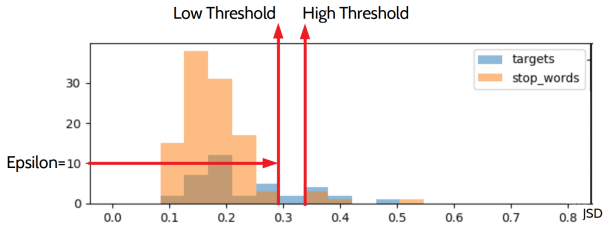

Figure 1: Distribution of semantic change scores in the English corpus: target words VS stopwords

	aff-prop	avg	kmeans_5	W2V	GS
aff-prop	1				
averaging	0.789	1			
kmeans_5	0.815	0.811	1		
word2vec	0.501	0.558	0.481	1	
Gold Standard	0.298	0.397	0.305	0.394	1

Table 2: Spearman correlation between the semantic change scores of various methods and the gold standard, averaged for all languages.

a cluster is computed for each target word as twice its average cluster size.

2.4 Subtask 2: Ranking

For Subtask2, target words were ranked according to the semantic change scores described in Section 2.2, namely divergence between cluster distributions (JSD) or cosine distance. Additional steps were performed in some of our submissions to improve this basic approach: cluster filtering and ensembling.

2.4.1 Cluster Filtering

Affinity propagation tends to produce a large number of clusters, and cluster size distribution is highly skewed. We try several heuristics to filter out the clusters that potentially contain noise and can distort the comparison between time periods. The first idea is to remove the smallest clusters (containing only one or two instances), whose appearance in a given time period is not significant. The second idea is to filter out sentences in which a target word is used as a proper noun, as in the following example: *her daddy warn everyone that rose* ***lane_nn*** *be bring home a musician with long hair*.

Finally, we noticed that some clusters contain sentences that refer to specific events. For example, one of the clusters for *attack* contains sentences about terrorist attack in Israel and consists only of sentences from the latter time period, for the obvious reasons. The sentences in this cluster contain many named entities (NEs), e.g.: *hezbollah leader hassan fadlallah defend* ***attack_nn*** *on israeli civilian target civilian be a war crime*. We filter out clusters that contain too many NEs in some of our submissions, though this "radical" NE filtering may have drawbacks: one may argue that a "terrorist attack" is a new meaning of a word *attack* that was correctly distinguished by the clustering algorithm but then discarded by filtering.

In a real-world application, NE recognition should be done on documents with preserved capitalization, preferably using a model trained specifically on historical documents. For the shared task we rely on out-of-the-box NLP pipelines.[5] Most of the tools are unable to recognize names in lowercased lemmatized text but POS-taggers are more reliable: e.g., the SpaCy NE recognition model was unable to recognize lower-cased names even if the SpaCy POS-tagger labeled the corresponding tokens as proper nouns.

We performed the NE filtering as a post-processing step, to compensate for errors in the NE recognition: we filter out a cluster if at least 80% of the target word mentions are NE. For the radical filtering, a cluster is filtered out if the number of proper nouns is 5 times larger than the number of sentences.

2.4.2 Ensembling

We ensemble different approaches for semantic change detection by multiplying the semantic change scores produced by different methods for each target word. We choose multiplication rather than the arithmetic average since the underlying distributions of the semantic shift measures are unknown, even though they produce numbers within the same range. If, for example, the numerical values of a particular measure are generally larger than values of another measure, the former measure would contribute more to the average and thus dominate the ensemble. Multiplication does not have this side effect.

We experiment with different combinations of averaging, clustering and Word2Vec based methods in order to test the hypothesis that the synergy between contextualised and static embeddings improves the

[5]We used SpaCy for English and German (`https://spacy.io/`), Polyglot for Swedish (`https://pypi.org/project/polyglot/`) and CLTK for Latin (`http://cltk.org/`).

Model	Binary method	AVG	English	German	Latin	Swedish
k-means 5	time-period specific clusters	0.600	0.649	0.542	0.500	**0.710**
aff-prop	time-period specific clusters, dynamic threshold	0.496	0.568	0.458	**0.700**	0.258
aff-prop, merging cluster	time-period specific clusters, dynamic threshold	0.545	0.514	0.542	0.575	0.548
aff-prop	stopwords, high threshold	0.573	0.622	0.604	0.550	0.516
aff-prop	stopwords, low threshold	0.552	**0.703**	0.667	0.450	0.387
ensemble: averaging + aff-prop	stopwords, low threshold	**0.621**	0.568	**0.688**	0.550	0.677

Table 3: SubTask 1 results: accuracy.

overall performance. Combinations of models that are too strongly correlated (above 0.8) are discarded. Some correlations averaged for all languages can be found in Table 2, though these values hide important disparities among languages.

3 Results

3.1 Subtask1

The results for the binary classification are shown in Table 3. We use BERT fine-tuned on the Semeval corpora for all submissions. The best official result was achieved by applying the stopword thresholding method to rankings obtained by measuring the JSD between affinity propagation cluster distributions. The stopwords thresholding method seems to work best with higher thresholds, which classify fewer words as changed.

The method of identifying period-specific clusters worked competitively when conducted on k-means clusters but performed worse with affinity propagation, since the latter method usually produces a large number of clusters. Reducing the number of clusters by merging the closest clusters together increased the performance of the method.

Looking at the average accuracy, the stopwords method seems to work better than the period-specific clusters method. However, we face high discrepancies between languages. Comparing the results for the same model, i.e. BERT with affinity propagation clustering, the latter method worked best for Latin and worse than the stopwords method for all the other languages.

3.2 Subtask2

Results for SubTask 2 are presented in Table 4. The best official result was obtained by an ensemble of Word2Vec static embeddings and fine-tuned BERT contextual embeddings, further improved with radical NE filtering as a postprocessing step—see row $\#11$ in the table. The good performance of the method can be explained by the fact that the semantic change scores outputted using static embeddings and contextualised embeddings are not highly correlated, as shown in Table 2 and we speculate that these two types of embedding capture different aspects of the semantic change.

Ensembling of four different methods—affinity propagation, K-means (k=5), averaging and Word2Vec with OP alignment—allows the merging of all the information that they gather $(\#12)$. However, it still does not out-perform the ensemble of only affinity propagation and Word2Vec $(\#10)$.

The cosine distance between averaged contextual embeddings performs much better than between Word2Vec representations for Latin but worse for other languages (rows $\#8$ and $\#9$). The affinity propagation clustering, which was the best in our previous study (Martinc et al., 2020a), did not perform well (rows $\#1$ to $\#6$), especially for Swedish, where it performed close to random. One explanation for this discrepancy could be the shuffling of sentences in the shared task corpora. BERT models cannot leverage the usual sequence of 512 tokens as a context in this setting but are limited to the number of tokens in the sentence. The correlation between larger context and better performance of the transformer-based models has been shown on some NLP tasks before (Dai et al., 2019). Therefore, the lack of context could have a detrimental effect on the quality of BERT contextual embeddings. The results however do suggest that by averaging these embeddings, a static embedding of good quality for each target token can be obtained.

The radical NE filtering has a significant impact on English and German results (compare rows $\#2$ to $\#5$), though in the opposite directions: it improves the performance on the English corpus from 0.313

	Input	Method	Post-Processing	AVG	English	German	Latin	Swedish
Clustering								
1	pretrained BERT	aff-prop, JSD	-	0.278	0.216	0.488	**0.481**	-0.072
2	fine-tuned BERT	aff-prop, JSD	-	0.298	0.313	0.436	0.467	-0.026
3	fine-tuned BERT	aff-prop, JSD	small clusters	0.302	0.327	0.440	0.472	-0.030
4	fine-tuned BERT	aff-prop, JSD	target NE	0.300	0.328	0.426	0.467	-0.023
5	fine-tuned BERT	aff-prop, JSD	NE	0.295	**0.436**	0.302	0.467	-0.025
6	fine-tuned BERT	aff-prop, JSD	NE, small clusters	0.291	0.413	0.310	0.472	-0.029
7	fine-tune BERT	kmeans k=5, JSD	-	**0.320**	0.189	**0.528**	0.324	**0.238**
Methods not using clustering								
8	fine-tune BERT	averaging, cosine dist	-	**0.397**	0.315	0.565	**0.496**	0.212
9	word2vec OP	cosine dist	(Schlechtweg et al., 2019)	0.394	**0.341**	**0.691**	0.131	**0.413**
Ensembling								
10	aff-prop (#2) + w2v (#9)	distance multiplication	-	0.417	0.357	**0.642**	0.366	0.303
11	aff-prop (#2) + w2v (#9)	distance multiplication	NE, small clusters	**0.442**	**0.361**	0.603	**0.460**	**0.343**
12	aff-prop(#2), k-means (#7), averaging (#8), w2v (#9)	multiplication, equal weights	-	0.403	0.279	0.607	0.451	0.276

Table 4: SubTask 2 results: Spearman correlation with the ground truth. Submissions made during the official evaluation phase are marked with yellow. Numbers preceded with # refer to the rows in this table, i.e. models used for the ensembling.

to 0.436 but reduces it on the German corpus from 0.436 to 0.302. Filtering as such slightly reduces the average performance (compare #2 to #6), but by removing small clusters (row #3) we gain slight improvements for all four corpora. The best performing method also uses filtering, which improves the ensemble performance for all corpora except for German (compare rows #10 and #11).

Many of the techniques that we try improved the overall method performance only for English: BERT fine-tuning, affinity propagation clustering, NE filtering. This might be related to the fact that the corpora are lemmatized, and lemmatization has a smaller effect on English, with its reduced morphology. The poor results on the Swedish corpus might be related to OCR-errors, leading to a large number of out-of-vocabulary tokens. BERT models deal with the out-of-vocabulary words by using a vocabulary of sub-word units (Kudo and Richardson, 2018). However, the vocabulary size is fixed and consists of 30,522 sub-word units, which might not be enough for a noisy corpus. This is supported by the findings of the another participant of the SemEval-2020 Task 1, which showed that character-based embeddings (ELMo) yield a significant improvement over BERT on the Swedish corpus (Kutuzov and Giulianelli, 2020).

4 Conclusion

We present the approaches employed by the Discovery team to tackle SemEval-2020 Task 1: Unsupervised Lexical Semantic Change Detection (Schlechtweg et al., 2020). While our main method was based on clustering BERT contextualised embeddings, the best official result was obtained by combining this technique with a method for semantic change detection based on static Word2Vec embeddings.

The methods based on contextualised embeddings with clustering are outperformed by averaging of contextualised embeddings and static embeddings methods. Other task participants, in particular the winning team of Subtask 2, used similar static and contextual methods and reached the same conclusion on the adequacy of static embeddings for these specific tasks and corpora (Pömsl and Lyapin, 2020). However, the discrepancy among languages is significant and the results averaged on all four corpora can be misleading. A more thorough analysis on how different embeddings perform in different settings (short or long term semantic change, type of corpus preprocessing, etc...) and different languages will be performed in the future work.

Acknowledgements

This work has been partly supported by the European Union's Horizon 2020 research and innovation programme under grant 770299 (NewsEye) and 825153 (EMBEDDIA), and Project Development of Slovene in the Digital Environment (RSDO), co-financed by the Republic of Slovenia and the European Union from the European Regional Development Fund.

References

Domagoj Alagić, Jan Šnajder, and Sebastian Padó. 2018. Leveraging lexical substitutes for unsupervised word sense induction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *EMNLP/IJCNLP*.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315(5814):972–976.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online, July. Association for Computational Linguistics.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Andrey Kutuzov and Mario Giulianelli. 2020. Uio-uva at semeval-2020 task 1: Contextualised embeddings for lexical semantic change detection. *To appear in SemEval@COLING2020*.

Matej Martinc, Syrielle Montariol, Elaine Zosa, and Lidia Pivovarova. 2020a. Capturing evolution in word usage: Just add more clusters? In *Companion Proceedings of the Web Conference 2020*, pages 343–349.

Matej Martinc, Petra Kralj Novak, and Senja Pollak. 2020b. Leveraging contextual embeddings for detecting diachronic semantic shift. In *LREC*.

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

Martin Pömsl and Roman Lyapin. 2020. Circe at semeval-2020 task 1: Ensembling context-free and context-dependent word representations. *To appear in SemEval@COLING2020*.

Dominik Schlechtweg, Anna Hätty, Marco Del Tredici, and Sabine Schulte im Walde. 2019. A wind of change: Detecting and evaluating lexical semantic change across times and domains. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 732–746.

Dominik Schlechtweg, Barbara McGillivray, Simon Hengchen, Haim Dubossarsky, and Nina Tahmasebi. 2020. Semeval-2020 task 1: Unsupervised lexical semantic change detection. *To appear in SemEval@COLING2020*.