DEF2VEC: You Shall Know a Word by Its Definition

Irene Morazzoni[1], Vincenzo Scotti[1*], Roberto Tedesco[1]

[1]DEIB, Politecnico di Milano, Via Golgi 42, Milano, 20133, MI, Italy.

*Corresponding author(s). E-mail(s): vincenzo.scotti@polimi.it;
Contributing
authors: irene.morazzoni@mail.polimi.it; roberto.tedesco@polimi.it;

Abstract

DEF2VEC introduces a new perspective on building words embeddings by using dictionary definitions. By leveraging term-document matrices derived from dictionary definitions and employing *Latent Semantic Analysis* (LSA), our method, DEF2VEC, yields embeddings characterized by robust performance and adaptability. Through comprehensive evaluations encompassing token classification, sequence classification and semantic similarity, we show empirically how DEF2VEC consistently demonstrates competitiveness with established models like WORD2VEC, GLOVE, and FASTTEXT. Notably, our model's utilization of all the matrices resulting from LSA factorisation facilitates efficient prediction of embeddings for out-of-vocabulary words, given their definition. By effectively integrating the benefits of dictionary definitions with LSA-based embeddings, DEF2VEC builds informative semantic representations, all while minimizing data requirements. In this extension, we further investigate the efficacy of sub-word embeddings to our model and our experimentation to assess the quality of our embedding model. Our findings contribute to the ongoing evolution of word embedding methodologies by incorporating structured lexical information and enabling efficient embedding prediction.

Keywords: Natural Language Processing, Deep Learning, Latent Semantic Analysis, Word Embeddings, Wiktonary

1 Introduction

Natural Language Processing (NLP) has undergone a remarkable evolution driven by the adoption of *distributed semantic representations*, which allows providing neural models with language understanding capabilities [1]. In particular, *Word embeddings* have

emerged as key component in the development of these models, offering a means to translate orthogonal and sparse data points into dense and compact vector representations that capture the rich semantic of single word tokens [2, Chapter 6]. These embeddings serve as input for a plethora of neural network-based NLP models, ranging from sentiment analysis to machine translation and chatbot assistants, empowering the learning algorithms with the ability to comprehend and manipulate language (within some extent) [3–6].

More recent advances have witnessed the shift from traditional static word embeddings towards *contextual embeddings*, thanks to the advent of *Transformer Networks* [7]. Contextual embeddings, introduced by models like BERT and GPT, iteratively aggregate information from the static embeddings composing the input sequence into all the elements of the context, capturing the contextualised meaning of the single embeddings [4, 8–10]. Despite the current widespread adoption of contextual embeddings in NLP, static model remain indispensable for their simplicity, interpretability, and computational efficiency [11].

In this paper, we further investigate on our approach to build extensible word embeddings: Def2Vec [12]. Def2Vec bridges the gap between traditional static embeddings and the rich informational context provided by dictionary definitions. The idea is to combine the usefulness of static embeddings and the advantages of having an extensible embedding vocabulary, obtaining the extensibility from the structured knowledge embedded within dictionary definitions. Our method involves constructing term-document matrices from these definitions, followed by factorisation using *Latent Semantic Analysis* (LSA) [13, 14]. The embeddings derived from this process exhibit competitive performance across various NLP tasks with respect to state-of-the-art static embeddings.

As an extension of our previous work, this paper keeps exploring the performances of Def2Vec embeddings. With the additional experiments, we aim at better assessing Def2Vec's capabilities in sequence level analysis. To this end, we selected three additional benchmark to match the two of the previous paper.

We divide the reminder of this paper into the following sections. In Section 2, we provide the background information on the different types of embedding models for text. In Section 3, we detail the aspects of the Def2Vec word embedding model. In Section 4, we describe the selected experimental approach and data sets we adopted to fit and evaluate our model. In Section 5, we present the results of the experiments and we comment over them. Finally, in Section 6, we summarise our work and present ideas for possible future extensions.

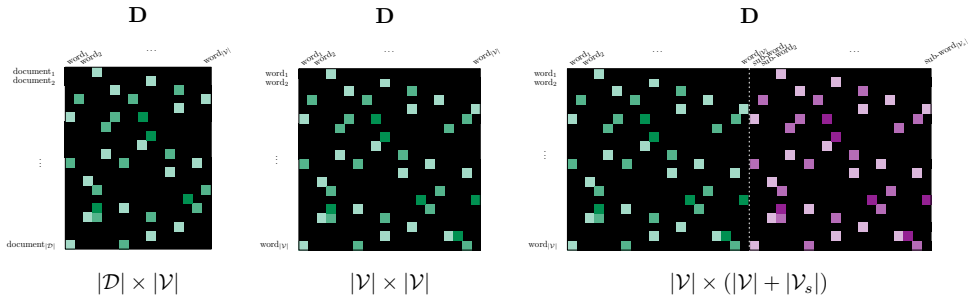
2 Background

In this section, we explain the differences between the two main types of embedding models used to encode text: *token-level embeddings* (Section 2.1) and *sequence-level embeddings* (Section 2.2).

2.1 Token-level embeddings

In this section, we explain the differences between the two main types of token-level embedding models: *static embedding models* (Section 2.1.1) and *contextual embedding models* (Section 2.1.2).

2.1.1 Static models



(a) Document-term matrix (b) Terms co-occurrence matrix (c) Terms co-occurrence matrix with sub-word tokens information

Fig. 1: Examples of term frequency matrices to factorise

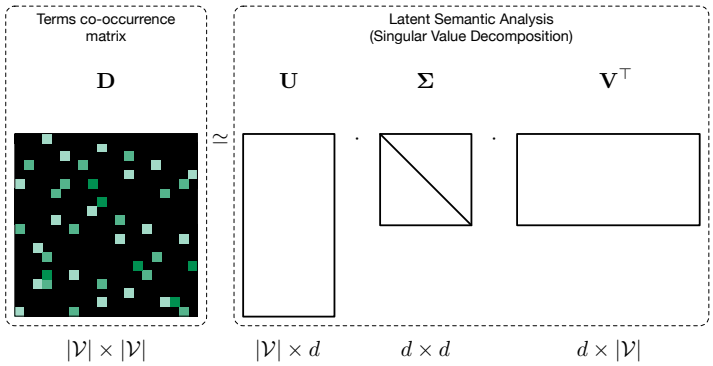


Fig. 2: Terms co-occurrence matrix decomposition via LSA

Static word embeddings, also referred to as shallow embeddings, encode word meanings independently of context, representing words as fixed vectors. The idea behind these embeddings is to learn a *term-document* or *term co-occurrence* matrix factorisation (see Figures 1a and 5b). In more recent models, these matrices include also sub-word (char) token (see Figure 1c), which provide morphological information to the embeddings. Previous approaches to solve this factorisation problem were based on LSA –i.e., *truncated Singular Value Decomposition* (SVD)– applied to one of these matrices [13, 14] (see Figure 2); nowadays, we adopt *prediction*-based or *count*-based solutions which are based on train a neural network to factorise those same matrices. Once the matrix is factorised, the embedding of a given word can be fetched from the rows (or columns) of one of the matrices resulting from the factorisation (see Figure 3).

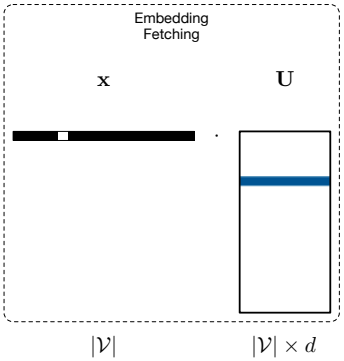


Fig. 3: Fetching a word embedding using the word's one-hot encoding

Within this static embeddings category, prominent methods include those WORD2VEC [15, 16], GLOVE [17], and FASTTEXT [18]. WORD2VEC is a prediction based method that employs techniques such as *skip-gram* or *Continuous-Bag-of-Words* (CBoW) to build the matrix to factorise. FASTTEXT extend this model including he counts of character n-grams to the matrix. GLOVE, on the other side, is a count-based technique and uses global *co-occurrence statistics* of the tokens in the training corpus to learn the embeddings. Differently from the other two models, FASTTEXT can generate embeddings even for out-of-vocabulary words composing them by averaging the character n-gram embeddings.

2.1.2 Contextual models

Contextual embeddings, on the other hand, iteratively integrate contextual information to produce dynamic representations that are sensitive to the surrounding context. Models such as ELMO [19], BERT [8], GPT [9] excel in this domain by leveraging *Deep Neural Networks* (DNNs) for sequence processing [1]. These models build upon static embeddings as initial representations and employ architectures like *Recurrent Neural Networks* (RNNs) [20, 21] or Transformer Networks [7] to capture contextual nuances within sequences of tokens. The DNNs are trained in a *self-supervised* fashion trying to predict the missing of the next token in a given sequence.

Models like ELMO and BERT use a bi-directional encoder architecture to build the contextual representation, using the portion of the sequences preceding and following a given token. This approach makes these models more suitable for discriminative tasks since the produced representations encapsulate both preceding and succeeding tokens. Differently, GPT employs a causal architecture to build the contextual representation, using only the portions of a sequence preceding the current token. The models adopting this architecture are more suitable for text generation, as they allow to autoregressively composing a token generating the sequence one piece at a time.

2.2 Sequence-level embeddings

In this section, we explain the differences between the two main types of embedding models used to encode sequences: *non-parametric embeddings* (Section 2.2.1) and

parametric embeddings (Section 2.2.2). Often, these embedding aggregation techniques are employed to obtain a fixed size vector representation of a sequence to compare sequence similarity. While *cosine similarity* is a common metric to be used with both type of models, there are some valid alternatives that better correlate with human judgment using non-parametric models.

2.2.1 Non-parametric

Non-parametric sequence embeddings rely on pre-trained models for word embeddings or statistical methods to generate representations. These approaches are often simpler and computationally less intensive compared to their parametric counterparts. These non-parametric methods are particularly useful in scenarios where computational resources are limited or where a quick and straightforward representation of the sequence suffices for the task at hand.

Examples include applying *average-pooling* or *max-pooling* to token embeddings, *Smooth Inverse Frequency* (SIF) *weighting* [22], DynaMax [23] and SFBoW [24, 25]. Some of these techniques work better when combined with other metrics rather than cosine similarity to compute semantic similarity between sequences, for example SIF weighting seems to work better with *Jaccard similarity* [23], while other approaches show that there is not need to aggregate the individual embeddings to compute the similarity, like in the case of *Words Mover Distance* (WMD) [26].

2.2.2 Parametric

Parametric sequence embeddings models are obtained through DNNs trained using either supervised or unsupervised approaches. Supervised approaches require labelled data for training, tailoring models to specific tasks like semantic similarity or paraphrasing. Unsupervised approaches, on the other hand, leverage unlabeled data without human annotations. These models often employ *self-supervised learning* or *contrastive learning* tasks to capture semantic similarities and differences between sentences. These parametric models offer the advantage of generating contextually rich representations tailored to specific downstream tasks, making them particularly suitable for tasks like sentence similarity, text classification and information retrieval. In particular, the latter task can be eased when combining these embeddings with vector data bases for fast search like FAISS [27, 28], Annoy [29] and HNSWLib [30].

Parametric sequence embeddings are generated by neural network architectures trained to produce fixed-size vectors from input sequences, typically sentences. Examples of such approaches include Skip-Thought vectors [31], Sent2Vec [32], and Sentence-BERT [33–35]. These models leverage neural network architectures to encode the semantic information of entire sentences into compact representations. For instance, the Skip-Thought vectors model uses an encoder-decoder architecture to predict the next sentence embedding given the current sentence, effectively capturing the contextual information within the sequence. Similarly, Sentence-BERT employs a *siamese network architecture* to learn sentence embeddings in a supervised learning fashion by maximising the similarity between semantically similar sentences and minimising the similarity between dissimilar ones.

3 Def2Vec model

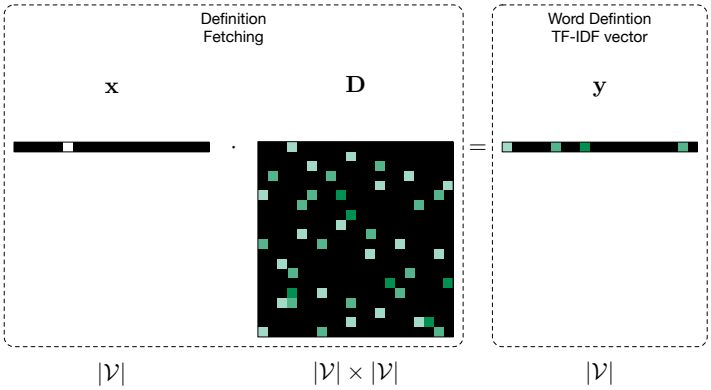


Fig. 4: Fetching a definition from the term-definition matrix

Def2Vec introduces an innovative approach to learning word embeddings that harnesses the structured information contained within dictionary definitions. We extract the embeddings starting from the term-document matrices build from these definitions: a *term-definition* matrix. This matrix is decomposed using LSA to extract the embeddings.

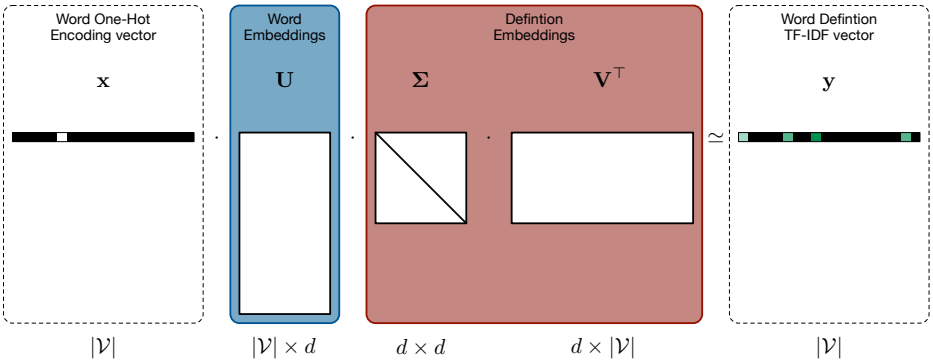
To build the term-definition matrix, each term in the vocabulary $\mathcal{V}$, comprising words or multi-word expressions, is associated with one or more definitions taken from some linguistic resource. Def2Vec constructs a term-definition matrix $\mathbf{D}$, where each row corresponds to the *Term Frequency-Inverse Document Frequency* (TF-IDF) representation of the definitions (or definitions, in the case of polysemous words) associated with a term, as shown in Figure 4. Given a term $w \in \mathcal{V}$ represented as a one-hot vector $\mathbf{x} \in \mathbb{1}^{|\mathcal{V}|}$ and its corresponding TF-IDF definition vector $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}|}$, the relationship between is defined as in Equation (1), where $\mathbf{D}$ is the sparse matrix connecting terms to their definitions.

$$\mathbf{y} = \mathbf{x} \cdot \mathbf{D} \tag{1}$$

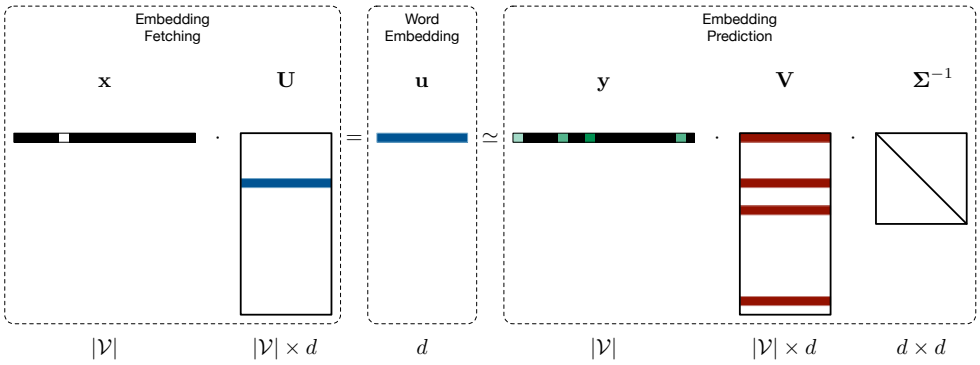
This term-definition matrix can be further extended to include sub-word information by concatenating the TF-IDF vector of the definition with the *Term Frequency* (TF) or TF-IDF vector of the characters n-gram composing the word. The resulting matrix to decompose is similar to the one depicted in Figure 1c.

Starting from this term-definition matrix $\mathbf{D}$, we obtain the Def2Vec embeddings applying LSA to $\mathbf{D}$. The matrix is factorised as reported in Equation (2), where $\mathbf{\Sigma}$ is a diagonal matrix containing singular values, and $\mathbf{U}$ and $\mathbf{V}$ contain left and right singular vectors, respectively. The approximate equality in reconstructing the TF-IDF definition, see Figure 5a, is determined by the internal dimensionality of the LSA projections $d << |\mathcal{V}|$. d is the dimensionality of the learnt embeddings, which are encoded into the $\mathbf{U}$ matrix coming from the decomposition, the i-th row $\mathbf{u}_i$ embeds the i-th term w_i in the vocabulary

$$\mathbf{D} \simeq \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^\top \tag{2}$$



(a) Approximate equality induced by the term-definition matrix decomposition



(b) Predicting and word embedding from the word definition

Fig. 5: DEF2VEC working mechanism

The core feature of the DEF2VEC embeddings lies in the extensibility of the embedding vocabulary. The emebddings of the words defined at training time are extracted by the $\mathbf{U}$ matrix. However, by exploiting the right singular vectors in $\mathbf{V}$ and the singular values in $\mathbf{\Sigma}$, the embeddings of out-of-vocabulary words can be predicted with some approximation from the TF-IDF representation of the new term's definition. This process, depicted in Figure 5b, makes the DEF2VEC embeddings extensible by enabling the addition of new terms without retraining the entire model and requiring only the definition of such term.

In summary, DEF2VEC offers an alternative methodology to learn static embeddings. The method combines the interpretability of linear models and with the property of extensibility, which other embedding models are lacking.

4 Experiments

In this section, we detail the experiments we conducted to assess the quality of the DEF2VEC model. We describe the selected data set to fit the DEF2VEC model and the

Table 1: Comparison of vocabulary size and data set size of the considered word embedding models.

Model	Vocabulary size $\times 10^6$	No. training tokens $\times 10^9$
DEF2VEC	0.76	0.05
WORD2VEC	3	100
GLOVE	2	840
FASTTEXT	2	600

Table 2: Statistics on considered data sets

Data set	Split	No. of samples	Avg. No. of tokens (per seq.)	Vocab. size	Faction of samples w/ infreq. terms	Faction of infreq. terms in vocabulary
CoNLL-2003	Train	14,041	14.5	21,009	0.59	0.59
	Val.	3250	15.8	9002	0.59	0.48
	Test	3453	13.4	8548	0.57	0.50
SST	Train	8117	19.6	16,310	0.69	0.43
	Val.	2125	19.2	7715	0.68	0.33
	Test	1044	19.4	4856	0.70	0.29
20NG	Train	8485	423.5	153,377	0.99	0.87
	Val.	2829	423.1	69,653	0.99	0.78
	Test	7532	379.9	120,763	0.99	0.84
WELFAKE	Train	40,575	633.2	291,389	0.99	0.91
	Val.	13,525	626.0	147,835	0.99	0.85
	Test	18,034	634.4	178,648	0.99	0.87
STS	Train	5749	11.1	12,467	0.53	0.41
	Val.	1500	12.9	6567	0.48	0.26
	Test	1379	11.1	4882	0.37	0.26

data sets to run the evaluation (Section 4.1) and we explain the approach we followed to run the comparisons and ablations on the selected benchmarks (Section 4.2).

4.1 Data

We trained the DEF2VEC models for our experiments using the definitions from the English WIKTIONARY[1]. We re-used the same dump as the previous experiments (the dump was updated to September 2020). As highlighted by Table 1, when compared to the training data sets of other embedding models, this dataset reveals a significantly lower data requirement; this, however, results in a smaller vocabulary size. Inside the WIKTIONARY vocabulary, only ~ 30000 tokens occurr in at least 10 documents, making the rest of the vocabulary infrquent and posing challenges in learning effective embeddings for those words.

In our experiments, we employed different data sets to analysis different characteristics of the DEF2VEC embeddings. We selected the CoNLL-2003 data set [36], which

[1]Website: https://en.wiktionary.org/wiki/Wiktionary:Main_Page.

offers labels for *Part-of-Speech* (POS) tagging, *chunking* (CHUNK), and *Named Entity Recognition* (NER) tasks, aimed at analysing token-level representation capabilities. For sequence-level evaluations, we used the *Stanford Sentiment Treebank* (SST) [37] for sentiment analysis, the *20 News Groups* (20NG) [38] data set for news classification, the WELFake data set [39] for fake.news detection and the *Semantic Textual Similarity* (STS) data set [40] for measuring sentence similarity. These data sets provide a diverse set of linguistic tasks for evaluating the performance of the Def2Vec model, we provide the main statistics on these data sets in Table 2.

4.2 Approach

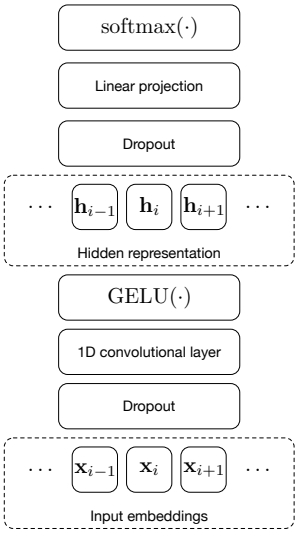
We defined two sets of experiments to assess the quality of the Def2vec embeddings. One group of experiments analyse the extensibility capabilities of the Def2vec model. Another group of experiments to compare Def2vec with other state-of-the-art models. We trained all models using the same configurations depending on the task and independently of the input embeddings.

During the ablations, we re-trained the Def2Vec model using only frequent words (those appearing at least 10 times in the Wiktionary definitions) and, then, we extended the embedding vocabulary by predicting the embeddings of missing words using the extension scheme of Def2vec. We then compared two variants (full vocabualary vs extended from reduced vocabulary) to train neural networks on a subset of the selected benchmarks (namely CoNLL-2003 and STS). These ablations should assess the extensibility capabilities of Def2vec by showing how results vary with an embedding model extracted from the whole dictionary data set and those extracted from a reduced dictionary data sets and then augmented with generated embeddings from definitions for out-of-vocabulary words. Statistics on infrequent words inside the benchmark data sets are available in Table 2.

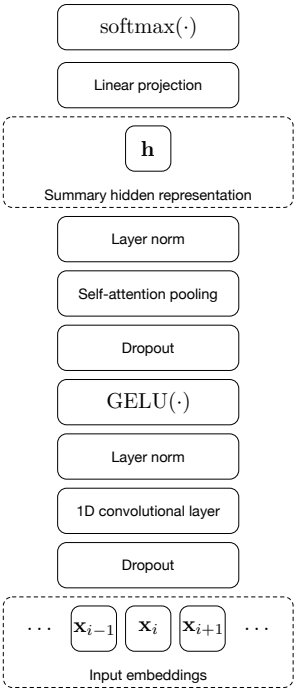
In the other group of experiments, we compare Def2vec with other existing models. We considered Word2vec, GloVE and FastText for our comparisons. We planned these experiments to understand if Def2vec can be a valid alternative to other embedding models.

In all the experiments, we resorted to a convolutional architecture. We avoided extensive architecture and hyper-parameter search since we were only interested in comparing the results between different configurations of Def2Vec or other embedding models in the same conditions. The architectures are depicted task-wise in Figure 6. All models doing sequence classification (SST, 20NG and WELFake) or embedding (STS), which required to aggregate information from the entire sequence, adopted a special self-attention pooling layer where the average of the embeddings in input is used as query in the self attention, producing a single output vector.

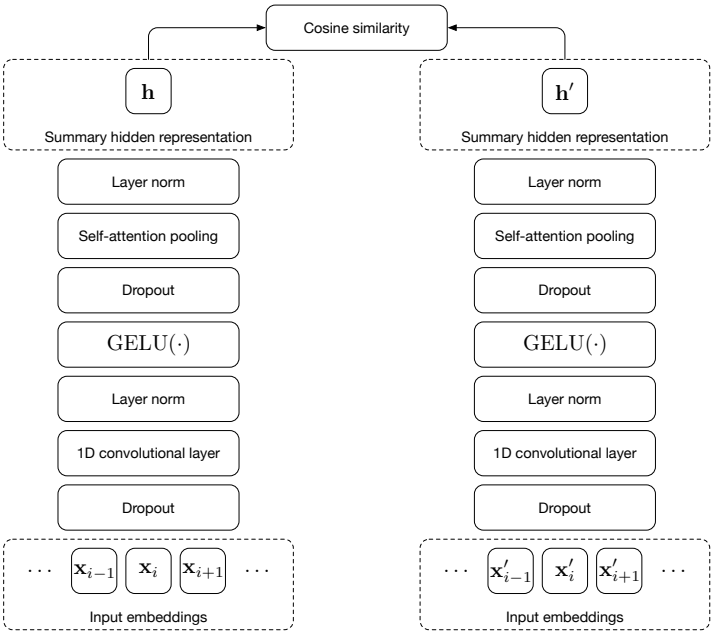
We trained the models for 64 epochs (16 for WELFake), with a learning rate of 10^{-3} (10^{-4} for WELFake) and a batch size of 512, and an early stopping patience of 5. For all token and sequence classification tasks (CoNLL-2003, SST, 20NG, WELFake) we use the negative log-likelihood loss, while with the regression task (STS) we computed the mean squared error loss.



(a) Token classification



(b) Sequence classification



(c) Siamese similarity

Fig. 6: Neural Network architectures used in the experiments

5 Results

In this section, we present and comment on the results of our experiments. We divide the section between the results on DEF2VEC embeddings exensibility (Section 5.1) and the comparison of DEF2VEC results against other embedding models (Section 5.2).

5.1 Extensibility

Tables 3 to 6 highlights the extensibility capabilities of the DEF2VEC model. In these tables we compared on a subset of the benchmarks the results of the base DEF2VEC model trained on all the available definitions (~ 700000), and that of the DEF2VEC

Table 3: Ablation of DEF2VEC classification results on the POS task from CONLL-2003

Hyper-parameter		Split	Metric [%]				
Reduced vocabulary	Extended vocabulary		Accuracy	Precision	Recall	F_1	AUC
		Validation	73.64	85.68	73.64	77.62	95.84
		Test	72.42	85.41	72.42	76.55	94.63
✓	✓	Validation	74.35	86.53	74.35	78.40	96.11
		Test	73.38	86.35	73.38	77.54	94.99

Table 4: Ablation of DEF2VEC classification results on the NER task from CONLL-2003

Hyper-parameter		Split	Metric [%]				
Reduced vocabulary	Extended vocabulary		Accuracy	Precision	Recall	F_1	AUC
		Validation	73.89	99.31	73.89	83.89	97.24
		Test	71.98	99.28	71.98	83.09	96.28
✓	✓	Validation	74.19	99.32	74.19	84.09	97.23
		Test	72.28	99.29	72.28	83.30	96.37

Table 5: Ablation of DEF2VEC classification results on the CHUNK task from CONLL-2003

Hyper-parameter		Split	Metric [%]				
Reduced vocabulary	Extended vocabulary		Accuracy	Precision	Recall	F_1	AUC
		Validation	77.79	86.81	77.79	81.34	94.37
		Test	77.69	86.56	77.69	81.45	93.07
✓	✓	Validation	77.84	86.97	77.84	81.47	94.44
		Test	77.84	86.61	77.84	81.56	93.08

Table 6: Ablation of DEF2VEC correlation scores on the STS benchmark

Hyper-parameter		Split	Spearman correlation [%] - Subset			
Full dictionary	**Extended**		**Caption**	**Forum**	**News**	**All**
		Validation	76.27	30.17	60.84	69.98
		Test	75.52	42.68	57.43	63.72
✓	✓	Validation	75.93	34.15	61.74	70.73
		Test	73.19	43.05	57.47	62.57

Table 7: Classification results on the POS task from CONLL-2003

Model	Split	Metric [%]				
		Accuracy	**Precision**	**Recall**	**F_1**	**AUC**
DEF2VEC	Validation	73.64	85.68	73.64	77.62	95.84
	Test	72.42	85.41	72.42	76.55	94.63
WORD2VEC	Validation	64.13	85.70	64.13	70.20	94.99
	Test	60.01	85.92	60.01	67.45	94.07
GLOVE	Validation	82.53	90.49	82.53	85.43	97.94
	Test	82.38	90.79	82.38	85.51	97.86
FASTTEXT	Validation	80.27	90.47	80.27	83.72	97.81
	Test	79.90	90.31	79.90	83.30	97.73

model trained on a subset of the definitions $\sim$30000, using the resulting model to predict the embeddings of the words missing from the tasks vocabulary exploiting the unused WIKTIONARY definitions.

The results are very close, in most of the cases we observed $<1\%$ difference between the model trained on the full dictionary and that trained on the reduced dictionary. This result highlighted two points: the embedding prediction process generating embeddings from the TF-IDF of the definiton provides embeddings compatible with the original ones and that, given the small difference, even a small dictinary with the most frequent word is sufficient to create valid representation with our model.

5.2 Comparisons

Tables 7 to 9 report the comparisons on token classification tasks from the CONLL-2003 data set. Tables 10 to 12 report the comparisons on sequence classification tasks from the SST, 20NG and WELFAKE data sets. All classification metrics are obtained though weighted micro average, using class frequencies as weight, to take into account possible class imbalances. Finally, Table 10 reports the comparison on semantic similarity from the STS data set.

Results of all models are undoubtedly below current state-of-the-art o these benchmarks. However, as anticipated, at this point we were only interested in a comparison in the same conditions of the different models, rather than on the overall results.

Table 8: Classification results on the NER task from CoNLL-2003

Model	Split	Metric [%]				
		Accuracy	**Precision**	**Recall**	**F_1**	**AUC**
Def2Vec	Validation	73.89	99.31	73.89	83.89	97.24
	Test	71.98	99.28	71.98	83.09	96.28
Word2Vec	Validation	75.00	99.21	75.00	84.50	96.52
	Test	73.31	99.25	73.31	83.95	95.44
GloVe	Validation	91.80	99.58	91.80	95.34	99.29
	Test	90.52	99.47	90.52	94.60	99.21
FastText	Validation	90.30	99.57	90.30	94.51	99.20
	Test	89.32	99.47	89.32	93.93	99.12

Table 9: Classification results on the CHUNK task from CoNLL-2003

Model	Split	Metric [%]				
		Accuracy	**Precision**	**Recall**	**F_1**	**AUC**
Def2Vec	Validation	77.79	86.81	77.79	81.34	94.37
	Test	77.69	86.56	77.69	81.45	93.07
Word2Vec	Validation	66.12	82.97	66.12	71.35	90.28
	Test	64.94	82.19	64.94	71.00	87.91
GloVe	Validation	80.09	89.86	80.09	84.09	95.18
	Test	79.43	89.20	79.43	83.51	94.49
FastText	Validation	82.38	90.21	82.38	85.60	95.03
	Test	82.28	89.63	82.28	85.39	94.55

Results on token classification tasks highlight the competitive performances of Def2Vec. In fact, Def2Vec always perform better or very close to Word2Vec, getting close to more sophisticated embedding techniques like such as GloVe and FastText. Conversely, on sequence level prediction, Def2Vec does not perform at the same level of other models. In these cases, the Def2Vec offers reasonable (sufficient) performances, but still falls below other techniques if we consider SST, 20NS or STS for example. Although, even in these case the distance from GloVe or FastText is reasonable, especially if considering the large difference in the amount of training data for the embedding models.

Finally, we want to point out the results on the WELFake, the largest data set, where Def2Vec performs st the same level of other model (better if considering the AUC, which is robust to label unbalance). The different results on these benchmarks suggest that no embedding model seems to clearly outperform the other, underlying that the choice of the embeddings depends on the specific task. Def2Vec can offer competitive performances, nevertheless it still has space for improvement, especially in sequence level analysis.

Table 10: Classification results on the sentiment analysis task from SST

Model	Split	Metric [%]				
		Accuracy	**Precision**	**Recall**	**F_1**	**AUC**
Def2vec	Validation	58.64	58.66	58.64	58.64	61.21
	Test	59.82	59.99	59.82	59.83	62.77
Word2Vec	Validation	69.95	69.95	69.95	69.94	74.31
	Test	69.31	69.29	69.31	69.28	74.62
GloVE	Validation	65.42	67.56	65.42	64.01	73.40
	Test	64.91	66.33	64.91	63.64	73.85
FastText	Validation	63.75	63.74	63.75	63.72	69.46
	Test	66.46	66.43	66.46	66.40	71.15

Table 11: Classification results on the news-group analysis task from 20NG

Model	Split	Metric [%]				
		Accuracy	**Precision**	**Recall**	**F_1**	**AUC**
Def2vec	Validation	64.86	69.34	64.86	65.27	96.19
	Test	56.40	61.69	56.40	56.68	94.53
Word2Vec	Validation	80.76	81.15	80.76	80.67	98.47
	Test	73.60	74.01	73.60	73.43	97.68
GloVE	Validation	82.35	82.94	82.35	82.21	98.83
	Test	76.52	77.50	76.52	76.26	98.19
FastText	Validation	72.73	73.90	72.73	72.50	97.50
	Test	67.35	68.55	67.35	67.08	96.83

Table 12: Classification results on the fake-news detection analysis task from WELFake

Model	Split	Metric [%]				
		Accuracy	**Precision**	**Recall**	**F_1**	**AUC**
Def2vec	Validation	78.19	80.80	78.19	77.45	90.63
	Test	78.42	80.69	78.42	77.73	90.53
Word2Vec	Validation	78.46	78.45	78.46	78.45	86.08
	Test	78.41	78.40	78.41	78.40	86.11
GloVE	Validation	76.84	78.86	76.84	76.65	87.68
	Test	76.82	78.83	76.82	76.66	87.78
FastText	Validation	79.97	80.23	79.97	79.84	87.84
	Test	79.99	80.20	79.99	79.86	88.03

Table 13: Correlation scores on the STS benchmark

Model	Split	Spearman correlation [%] - Subset			
		Caption	Forum	News	All
Def2Vec	Validation	76.27	30.17	60.84	69.98
	Test	75.52	42.68	57.43	63.72
Word2Vec	Validation	83.33	49.61	63.22	77.67
	Test	81.57	52.46	59.18	69.45
Glove	Validation	83.45	55.96	66.60	78.37
	Test	80.17	53.49	63.16	69.00
FastText	Validation	86.08	58.95	70.08	81.14
	Test	83.27	59.92	61.48	72.49

6 Conclusion

In this study, we extended our analysis of the Def2Vec embeddings, an approach for learning extensible word embeddings leveraging the semantic information found within dictionary definitions. Def2Vec effectively leverages the structured lexical information and LSA to construct embeddings that encapsulate both syntactic and semantic features. The main feature of these embeddings is the capability of predicting the embeddings of new words, given the definition of such words. With this paper, we additionally explored the use of sub-word information to enrich our embeddings.

Through our experiments, encompassing token classification, sequence classification and semantic similarity, we analysed both the embeddings extensibility characterising the Def2Vec model, which allows for out-of-vocabulary embeddings predictions, and we compared our model with state-of-the-art solutions for word embeddings. The results highlight the potential of our technique, which offer competitive results with well established embedding techniques. Finally, as highlighted in our previous work, the prediction of missing embeddings showed how our model can extend its vocabulary with negligible impact on the performances for predicted embeddings.

From this point on, we are looking forward to further extend our exploration on the Def2Vec embeddings. Right now, we are mainly interested in adapting Def2Vec to other languages rather than English to unveil cross-lingual variations in lexical semantics and offer insights into the generality of our approach and we are also interested in exploring the use of sub-word embeddings to enhance our model. Moreover, we are willing to explore the use of these Def2Vec embeddings as input for deep contextual models and observe the performances of language models built upon these embeddings on more sophisticated benchmarks.

Supplementary information. Not applicable.

Acknowledgements. Not applicable.

Declarations

Funding Not applicable.

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval Not applicable.

Data availability We provide a copy of the pre-processed data from the WIKTIONARY at the following link: https://polimi365-my.sharepoint.com/:f:/g/personal/10451445_polimi_it/EmVtl9bPq8pLjs1Bm3w8LegBhWQ51d2YoBq1-a0uKRbJ0Q?e=Erjor8. All data sets used for the experiments are openly available, the references to the specific data sets are mentioned in the paper and the links are shared inside the source code of this project.

Materials availability We provide a copy of the pre-trained DEF2VEC models used during the experiments at the following link: https://polimi365-my.sharepoint.com/:f:/g/personal/10451445_polimi_it/EvmXVdRit2hGqxrCrMbcQ7cB48iljQpbBsVxqdJoigLEdQ?e=2acSPX. We provide a copy of the models trained for the experiments on the evaluation tasks at the following link: https://polimi365-my.sharepoint.com/:f:/g/personal/10451445_polimi_it/EhKZkezHEZJClcjYmSZ_iYIBgys3iSt6ulcuJc_Y8ua-hg?e=8nNVXO. All the baseline models used for comparison are openly available, the references to the specific models are mentioned in the paper and the links are shared inside the source code of this project.

Code availability We provide a reference to the *GitHub* repository with the source code developed within this project at the following link: https://github.com/vincenzo-scotti/def_2_vec.

Author contribution All authors contributed equally to the research presented in this paper.

References

[1] Liu, Z., Lin, Y., Sun, M.: Representation Learning for Natural Language Processing. Springer, ??? (2020). https://doi.org/10.1007/978-981-15-5573-2 . https://doi.org/10.1007/978-981-15-5573-2

[2] Jurafsky, D., Martin, J.H.: Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition (3rd Edition). Draft (2023). https://web.stanford.edu/\texttildelowjurafsky/slp3/

[3] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140–114067 (2020)

[4] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[5] Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M.S., Xu, C., Thakker, U., Sharma, S.S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N.V., Datta, D., Chang, J., Jiang, M.T., Wang, H., Manica, M., Shen, S., Yong, Z.X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Févry, T., Fries, J.A., Teehan, R., Scao, T.L., Biderman, S., Gao, L., Wolf, T., Rush, A.M.: Multitask prompted training enables zero-shot task generalization. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, ??? (2022). https://openreview.net/forum?id=9Vrb9D0WI4

[6] Scotti, V., Sbattella, L., Tedesco, R.: A primer on seq2seq models for generative chatbots. ACM Comput. Surv. **56**(3), 75–17558 (2024) https://doi.org/10.1145/3604281

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp. 5998–6008 (2017). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[8] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: 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), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423 . https://aclanthology.org/N19-1423

[9] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. OpenAI blog **1**(11), 12 (2018)

[10] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)

[11] Hosseini, M., Sabet, A.J., He, S., Aguiar, D.: Interpretable fake news detection with topic and deep variational models. Online Soc. Networks Media **36**, 100249 (2023) https://doi.org/10.1016/J.OSNEM.2023.100249

[12] Morazzoni, I., Scotti, V., Tedesco, R.: Def2vec: Extensible word embeddings from dictionary definitions. In: Abbas, M., Freihat, A.A. (eds.) Proceedings of the 6th International Conference on Natural Language and Speech Processing (ICNLSP 2023), Virtual Event, 16-17 December 2023, pp. 212–222. Association for Computational Linguistics, ??? (2023). https://aclanthology.org/2023.icnlsp-1.21

[13] Deerwester, S.C., Dumais, S.T., Furnas, G.W., Harshman, R.A., Landauer, T.K., Lochbaum, K.E., Streeter, L.A.: Computer information retrieval using latent semantic structure. Google Patents. US Patent 4,839,853 (1989)

[14] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American society for information science **41**(6), 391–407 (1990)

[15] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Bengio, Y., LeCun, Y. (eds.) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013). http://arxiv.org/abs/1301.3781

[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a Meeting Held December 5-8, 2013, Lake Tahoe, Nevada, United States, pp. 3111–3119 (2013). https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[17] Pennington, J., Socher, R., Manning, C.: GloVe: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in

Natural Language Processing (EMNLP), pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/D14-1162 . https://aclanthology.org/D14-1162

[18] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017) https://doi.org/10.1162/tacl_a_00051

[19] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 2227–2237. Association for Computational Linguistics, New Orleans, Louisiana (2018). https://doi.org/10.18653/v1/N18-1202 . https://aclanthology.org/N18-1202

[20] Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990) https://doi.org/10.1207/s15516709cog1402_1

[21] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997) https://doi.org/10.1162/neco.1997.9.8.1735

[22] Arora, S., Liang, Y., Ma, T.: A simple but tough-to-beat baseline for sentence embeddings. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, ??? (2017). https://openreview.net/forum?id=SyK00v5xx

[23] Zhelezniak, V., Savkov, A., Shen, A., Moramarco, F., Flann, J., Hammerla, N.Y.: Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, ??? (2019). https://openreview.net/forum?id=SkxXg2C5FX

[24] Muffo, M., Tedesco, R., Sbattella, L., Scotti, V.: Static fuzzy bag-of-words: a lightweight and fast sentence embedding algorithm. In: Proceedings of the Fourth International Conference on Natural Language and Speech Processing (ICNLSP 2021), pp. 73–82. Association for Computational Linguistics, Trento, Italy (2021). https://aclanthology.org/2021.icnlsp-1.9

[25] Muffo, M., Tedesco, R., Sbattella, L., Scotti, V.: In: Abbas, M. (ed.) Static Fuzzy Bag-of-Words: Exploring Static Universe Matrices for Sentence Embeddings, pp. 191–211. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-11035-1_10 . https://doi.org/10.1007/978-3-031-11035-1_10

[26] Kusner, M.J., Sun, Y., Kolkin, N.I., Weinberger, K.Q.: From word embeddings to document distances. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 957–966.

JMLR.org, ??? (2015). http://proceedings.mlr.press/v37/kusnerb15.html

[27] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Trans. Big Data **7**(3), 535–547 (2021) https://doi.org/10.1109/TBDATA.2019.2921572

[28] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P., Lomeli, M., Hosseini, L., Jégou, H.: The faiss library. CoRR **abs/2401.08281** (2024) https://doi.org/10.48550/ARXIV.2401.08281 2401.08281

[29] Bernhardsson, E.: Annoy. https://github.com/spotify/annoy

[30] Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans. Pattern Anal. Mach. Intell. **42**(4), 824–836 (2020) https://doi.org/10.1109/TPAMI.2018.2889473

[31] Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R.S., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 3294–3302 (2015). https://proceedings.neurips.cc/paper/2015/hash/f442d33fa06832082290ad8544a8da27-Abstract.html

[32] Pagliardini, M., Gupta, P., Jaggi, M.: Unsupervised learning of sentence embeddings using compositional n-gram features. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 528–540. Association for Computational Linguistics, New Orleans, Louisiana (2018). https://doi.org/10.18653/v1/N18-1049 . https://aclanthology.org/N18-1049

[33] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (2019). https://doi.org/10.18653/v1/D19-1410 . https://aclanthology.org/D19-1410

[34] Reimers, N., Gurevych, I.: Making monolingual sentence embeddings multilingual using knowledge distillation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 4512–4525. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.emnlp-main.365 . https://aclanthology.org/2020.emnlp-main.365

[35] Thakur, N., Reimers, N., Daxenberger, J., Gurevych, I.: Augmented SBERT: Data augmentation method for improving bi-encoders for pairwise sentence scoring tasks. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp.

296–310. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.naacl-main.28 . https://aclanthology.org/2021.naacl-main.28

[36] Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp. 142–147 (2003). https://aclanthology.org/W03-0419

[37] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A Meeting of SIGDAT, a Special Interest Group of The ACL, pp. 1631–1642. ACL, ??? (2013). https://aclanthology.org/D13-1170/

[38] Lang, K.: Newsweeder: Learning to filter netnews. In: Prieditis, A., Russell, S. (eds.) Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995, pp. 331–339. Morgan Kaufmann, ??? (1995). https://doi.org/10.1016/B978-1-55860-377-6.50048-7 . https://doi.org/10.1016/b978-1-55860-377-6.50048-7

[39] Verma, P.K., Agrawal, P., Amorim, I., Prodan, R.: Welfake: Word embedding over linguistic features for fake news detection. IEEE Trans. Comput. Soc. Syst. **8**(4), 881–893 (2021) https://doi.org/10.1109/TCSS.2021.3068519

[40] Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp. 1–14. Association for Computational Linguistics, Vancouver, Canada (2017). https://doi.org/10.18653/v1/S17-2001 . https://aclanthology.org/S17-2001