Static Fuzzy Bag-of-Words: a Lightweight and Fast Sentence Embedding Algorithm

Matteo Muffo[1,2], **Roberto Tedesco**[1], **Licia Sbattella**[1] and **Vincenzo Scotti**[1]
[1]DEIB, Politecnico di Milano
Via Golgi 42, 20133, Milano (MI), Italy
[2]Indigo.ai
Via Torino 61, 20123, Milano (MI), Italy
matteo@indigo.ai roberto.tedesco@polimi.it
licia.sbattella@polimi.it vincenzo.scotti@polimi.it

Abstract

The introduction of embedding techniques has pushed forward significantly the Natural Language Processing field. Many of the proposed solutions have been presented for word-level encoding; anyhow, in the last years, new mechanisms to treat information at a higher level of aggregation, like at sentence- and document-level, have emerged. With this work, we address specifically the sentence embeddings problem, presenting the *Static Fuzzy Bag-of-Word* model. Our model is a refinement of the Fuzzy Bag-of-Words approach, providing sentence embeddings with a fixed dimension. SFBoW provides competitive performances in Semantic Textual Similarity benchmarks while requiring low computational resources.

1 Introduction

Natural Language Processing (NLP) has gained much traction in the last years, mainly thanks to learnt semantic representations. Those representations (usually) called embeddings are, in the textual context, real-valued vectors representing the semantic meaning of words, sentences or even documents in a Euclidean space. These vectors are features generated by models trained using a *self-supervised* approach on a vast corpus of unlabelled text. Leveraging features obtained through a self-supervised approach instead of "hand-selected" features is of crucial importance for NLP (Bengio et al., 2013).

Textual learnt representations immediately turned out to be significant in many NLP tasks, from more simple ones, like Part-Of-Speech (POS) tagging, Named Entity Recognition (NER), and language modelling (Collobert et al., 2011), to more complex problems such as Machine Translation (Sutskever et al., 2014), and even Conversational Systems (Sordoni et al., 2015). These representations significantly moved forward state of the art.

Results in the tasks mentioned above have been boosted mainly thanks to learnt word-level semantic vectors, i.e. *word embeddings*. Nevertheless, for many problems like web search, question answering and image captioning, having access to higher-level representations is crucial: this is where *sentence embeddings* find their usefulness (Yih et al., 2015).

Recent outcomes show that *contextual* representations, learnt through *Transformer Network* (Vaswani et al., 2017) Language Models (LMs) (Devlin et al., 2019; Radford et al., 2018), provide better performances in all those tasks and are slowly substituting "static" embeddings. Even though the results of these models are remarkable, their usability is strongly restricted because of their high demand in terms of computational resources; hence we decided to focus on more lightweight solutions.

With this work, we introduce the Static Fuzzy Bag-of-Words (SFBoW) model, an improvement of DynaMax Fuzzy Bag-of-Words model (Zhelezniak et al., 2019). Differently from its predecessor, the size of universe matrix (and thus the dimension of the generated embeddings) is fixed, hence the name *static*. SFBoW is characterised by low analysis time and interesting Semantic Textual Similarity (STS) results without demanding high computational power, making it an interesting solution for embedded systems and, in general, for applications where resources are limited and power consumption is a concern.

The rest of this document is organised as follows. In Section 2 we present the related works in the field of learnt semantic representations. In Section 3 we present the SFBoW model. In Section 4 we present the experiments to assess the quality of our model. In Section 5 we present the results of experiments. In Section 6 we sum up the entire work and propose possible future directions.

2 Related work

Our work revolves around the concept of *vector semantics*: the idea that the meaning of a word or a sentence can be modelled as a vector (Osgood et al., 1958).

The first steps on this subject were made in Information Retrieval (IR) context with the vector space model (Salton, 1971), where documents and queries were represented as high dimensional (vocabulary size) sparse embedding vectors. In this model, each dimension is used to represent a word, so that given a vocabulary $\mathcal{V}$:

- A word $w_i \in \mathcal{V}$, with $i \in [1, |\mathcal{V}|] \subseteq \mathbb{N}$, is expressed as a so called "one hot" binary vector $\mathbf{v}_{w_i} \in \mathbb{1}^{|\mathcal{V}|}$, where, calling $v_{w_i,j}$ the j-th element of the word vector, it holds that $v_{w_i,j} = 1 \iff j = i$.
- A sentence S is expressed as vector $\boldsymbol{\mu}_S \in \mathbb{N}^{|\mathcal{V}|}$, where $\mu_{S,i}$, the i-th element of vector $\boldsymbol{\mu}_S$, namely $c_{S,i}$, represents the number of times word w_i appears in sentence S.

The resulting sentence representation, used also for text documents, is called *Bag-of-Words* (BoW), and can be summarised as

$$\boldsymbol{\mu}_S = \sum_{i=1}^{|\mathcal{V}|} c_{S,i} \cdot \mathbf{v}_{w_i} \tag{1}$$

These representation models needed to be replaced because of the sparsity, which made them resource consuming, and the induced orthogonality among vectors with similar meanings.

2.1 Word and sentence embeddings

Word embeddings refer to the dense semantic vector representation of words. Approaches for word embeddings can be divided into: *prediction-based* and *count-based* (Baroni et al., 2014).

The former group identifies the embeddings obtained through the training of models for next/missing word prediction given a context. It encompasses models like *Word2Vec* (Mikolov et al., 2013a,b) and *fastText* (Bojanowski et al., 2017). The latter group refers to the embeddings obtained leveraging words co-occurrence counts in a corpus. One of the most recent solutions of this group is *GloVe* (Pennington et al., 2014).

All the models mentioned above belong to the class of *shallow* models, where the embedding of a word w_i can be extracted through lookup over the rows of the embedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$, with d being the desired dimensionality of the embedding space. Given the word (column) vector $\mathbf{v}_{w_i}$, the corresponding word embedding $\mathbf{u}_{w_i} \in \mathbb{R}^d$ can be computed as (see Section 2.2)

$$\mathbf{u}_{w_i} = \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{2}$$

More recently, the introduction of Transformer-based LMs, like *BERT* (Devlin et al., 2019) or *GPT* (Radford et al., 2018), has spread the concept of *contextual embeddings*; such embeddings proved to be particularly helpful for a wide variety of NLP problems, as shown by the leader-boards of NLP benchmarks (Wang et al., 2019; Rajpurkar et al., 2016).

The inherent hierarchical structure of the human language makes it hard to understand a text from single words; thus, the birth of higher-level semantic representations for sentences, which are the sentence embeddings, was just a natural consequence. As for the Word embeddings, also sentence embeddings are organised into two groups: *parametrised* and *non-parametrised*, depending on whether the model requires parameter training or not.

Clear examples of parametric model are the *Skip-Thoughts vectors* (Kiros et al., 2015) and *Sent2Vec* (Pagliardini et al., 2018), which generalises Word2Vec. Non-parametric models, instead, show that simply aggregating the information from pre-trained word embeddings, for example through averaging, as in *SIF weighting* (Arora et al., 2017), is sufficient to represent higher-level entities like sentences and paragraphs.

Transformer LMs are also usable at sentence level. An example is the parametric model *Sentence-BERT* (Reimers and Gurevych, 2019), obtained by fine-tuning on Natural Language Inference corpora.

All these models rely on the assumption that cosine similarity is the correct metric to compute "meaning distance" between sentences. This is why parametric models are explicitly trained to minimise this measure for similar sentences and maximise it for dissimilar sentences. However, this may not be the only and best measure. The *DynaMax* model (Zhelezniak et al., 2019) proposed to follow a fuzzy set representation of sentences and to rely on fuzzy Jaccard similarity instead of the cosine one. As a result, the DynaMax model outperformed many non-parametric models and performed comparably to parametric ones under cosine similarity

measurements, even if competitors were trained directly to optimise that metric, while the DynaMax approach was utterly unrelated to that objective.

The use of fuzzy sets to represent documents is not new, it was already proposed by Zhao and Mao (2018). With respect to DynaMax, previous results were inferior because of their approach to compute fuzzy membership.

2.2 Fuzzy Bag-of-Words and DynaMax for sentence embeddings

The *Fuzzy Bag-of-Words* (FBoW) model for text representation (Zhao and Mao, 2018) – and its generalised and improved variant DynaMax (Zhelezniak et al., 2019), which introduced a better similarity metric – represent the starting point of our work, which is described in Section 3.

The BoW approach, described at the beginning of Section 2, can be seen as a multi-set representation of text. It enables to measure similarity between two sentences with set similarity measures, like Jaccard, Otsuka and Dice indexes. These indexes share all a common pattern to measure the similarity σ between two sets A and B (Zhelezniak et al., 2019):

$$\sigma(A, B) = n_{shared}(A, B) / n_{total}(A, B) \quad (3)$$

where $n_{shared}(A, B)$ denotes the count of shared elements and $n_{total}(A, B)$ is the count of total elements. In particular, the Jaccard index is defined as

$$\sigma_{Jaccard}(A, B) = |A \cap B| / |A \cup B| \quad (4)$$

However, the simple set similarity is a rigid approach as it allows for some degree of similarity when the very same words appear in both sentences, but fails in the presence of synonyms. This is where *Fuzzy Sets theory* comes handy: in fact, fuzzy sets enable to interpret each word in $\mathcal{V}$ as a singleton and measure the degree of membership of any word to this singleton as the similarity between the two considered words (Zhao and Mao, 2018).

The FBoW model prescribes to work in this way (Zhao and Mao, 2018):

- Each word w_i is interpreted as a singleton $\{w_i\}$; thus, the membership degree of any word w_j in the vocabulary (with $j \in [1, |\mathcal{V}|] \subseteq \mathbb{N}$) with respect to this set is computed as the similarity σ between w_i and w_j. These similarities can be used to fill a $|\mathcal{V}|$-sized vector $\hat{\mathbf{v}}_{w_i}$ used to provide the fuzzy representation of w_i (the j-th element $\hat{\mathbf{v}}_{w_i,j}$ being $\sigma(w_i, w_j)$).
- A sentence S is simply defined through the fuzzy union operator, which is determined by the $\max$ operator over the membership degrees. In this case the S is represented by a vector of $|\mathcal{V}|$ elements.

The generalised FBoW approach (Zhelezniak et al., 2019), prescribes to computes the *fuzzy embedding* of a word singleton as

$$\hat{\mathbf{v}}_{w_i} = \mathbf{U} \cdot \mathbf{u}_{w_i} = \mathbf{U} \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \quad (5)$$

to reduce the dimension of the output vector for S. Where, $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is a word embedding matrix (defined as in Section 2.1), $\mathbf{u}_{w_i}$ is defined in Equation (2) and $\mathbf{U} \in \mathbb{R}^{u \times d}$ (with u being the desired dimension of the fuzzy embeddings) is the *universe matrix*, derived from the *universe set* U, which is defined as "the set of all possible terms that occur in a certain domain". The generalised FBoW produces vectors of u elements, where $u = |U|$.

Given the fuzzy embeddings of the words in a sentence S, the generalised FBoW representation of S is a vector $\hat{\boldsymbol{\mu}}_S$ whose j-th element $\hat{\mu}_{S,j}$ ($j \in [1, u] \subseteq \mathbb{N}$) can be computed as:

$$\hat{\mu}_{S,j} = \max_{w_i \in S} c_{S,i} \cdot \hat{v}_{w_i,j} \quad (6)$$

where $c_{S,i}$ and $\hat{v}_{w_i,j}$ are, respectively, the number of occurrences of word w_i in sentence S and the j-th element of the $\hat{\mathbf{v}}_{w_i}$ vector.

The universe set can be defined in different ways, same applies for the universe matrix (Zhelezniak et al., 2019). Among the possible solutions, the DynaMax algorithm for fuzzy sentence embeddings builds the universe matrix from the word embedding matrix, stacking solely the embedding vectors of the words appearing in the sentences to be compared.

Notice that in this way the resulting universe matrix is not unique, as a consequence neither are the embeddings. This condition can be noticed from the description of the algorithm and from the definition of the universe matrix: when comparing two sentences S_a and S_b, the universe set U used in their comparison is $U \equiv S_a \cup S_b$, so the resulting sentence embeddings have size $u = |U| = |S_a \cup S_b|$. In fact, the universe matrix is given by

$$\mathbf{U} = \left[\mathbf{u}_{w_i} \forall w_i \in U\right]^\top \quad (7)$$

This characteristic is unfortunate as, for example, in IR it requires a complete re-encoding of the entire document achieve for each query.

The real improvement of DynaMax is in the introduction of the fuzzy Jaccard index to compute the semantic similarity between two sentences S_a and S_b, rather than the generalisation of the FBoW, which replaced the original use of the cosine similarity (Zhao and Mao, 2018); see Equation (8):

$$\hat{\sigma}_{Jaccard}\left(\hat{\boldsymbol{\mu}}_{S_a}, \hat{\boldsymbol{\mu}}_{S_b}\right) = \\ = \frac{\sum_{i=1}^{u} \min\left(\hat{\mu}_{S_a,i}, \hat{\mu}_{S_b,i}\right)}{\sum_{i=1}^{u} \max\left(\hat{\mu}_{S_a,i}, \hat{\mu}_{S_b,i}\right)} \tag{8}$$

3 Static Fuzzy Bag-of-Words model

Starting from the DynaMax, which evolved from the FBoW model, we developed our follow up aimed at providing a unique matrix $\mathbf{U}$ and thus embeddings with a fixed dimension. In Figure 1 is represented the visualisation of our approach.

3.1 Word embeddings

Word embeddings play a central role in our algorithm as they also provide the start point of the construction of the universe matrix. For this work, we leveraged pre-trained shallow models (more details in Section 4.1) for two main reasons:

- The model is encoded in a matrix where each row corresponds to a word.
- We want to provide a sentence embedding approach that does not require training, easing its accessibility.

The vocabulary of these models, composed starting from all the tokens in the training corpora, is usually more extensive than the English vocabulary, as it contains named entities, incorrectly spelt words, non-existing words, URLs, email addresses, and similar. To reduce the computational effort needed to construct and use the universe matrix, we have considered some subsets of the employed word embedding model's vocabulary. Depending on the experiment, we work with either the 100 000 most frequently used terms, the 50 000 most frequently used terms (terms frequencies are given by the corpora used to train the word embedding model) or the subset composed of all the spell-checked terms present in a reference English dictionary (obtained through the Aspell English spell-checker[1]).

[1] http://aspell.net

In the following sections, the $\check{\mathbf{W}}$ symbol refers to these as *reduced* word embedding matrices/models.

3.2 Universe matrix

During the experiments, we tried three main approaches to build the universe matrix $\mathbf{U}$: the first two – proposed, but not explored, by the original authors of DynaMax (Zhelezniak et al., 2019) – consist, respectively, in the usage of a clustered embedding matrix and an identity matrix with the rank equal to the size of the word embeddings. Instead, the last approach consists of applying a multivariate analysis techniques to the word embedding matrix to build the universe one. In the following formulae, we refer to d as the dimensionality of the word embedding vectors, while the SFBoW embedding of the singleton of word w_i is represented as $\check{\mathbf{v}}_{w_i}$.

Clustering The idea is to group the embedding vectors into clusters and use their centroids; in this way, the fuzzy membership will be computed over the clusters – which are expected to host semantically similar words – instead of all the word singletons. The universe set is thus built out of abstract entities only, which are the centroids. Considering k centroids, the universe matrix $\mathbf{U} = \mathbf{K}^\top \in \mathbb{R}^{k \times d}$, and thus SFBoW k-dimensional embedding $\check{\mathbf{v}}_{w_i}$ of the singleton of word w_i is

$$\check{\mathbf{v}}_{w_i} = \mathbf{K}^\top \cdot \mathbf{u}_{w_i} = \left[\mathbf{k}_1, \ldots, \mathbf{k}_k\right]^\top \cdot \mathbf{u}_{w_i} = \\ = \mathbf{K}^\top \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{9}$$

where $\mathbf{k}_j$, the j-th (with $j \in [1, k] \subseteq \mathbb{N}$) column of $\mathbf{K}$, corresponds to the centroid of the j-th cluster. This approach generates k-dimensional word and sentence embeddings.

Identity Alternatively, instead of looking for a group of semantically similar words that may form a significant group, useful for semantic similarity, we consider the possibility of re-using the word embedding dimensions (features) to represent the semantic content of a sentence. So, we just use the identity matrix as the universe: $\mathbf{U} = \mathbf{I}$, with $|\mathbf{I}| = d \times d$, so that $\check{\mathbf{v}}_{w_i} \in \mathbb{R}^d$ is

$$\check{\mathbf{v}}_{w_i} = \mathbf{I} \cdot \mathbf{u}_{w_i} = \mathbf{I} \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \tag{10}$$

This approach generates d-dimensional word embeddings and sentence embeddings.

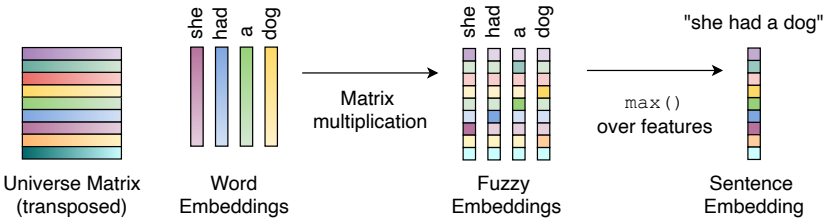

Figure 1: Visualisation of the Sentence Embedding computation process using SFBoW.

Multivariate analysis The same idea moves our multivariate analysis proposal. Judging by previous results, word embeddings aggregated correctly might be sufficient to provide a semantically valid representation of a sentence. What can bring better results might be as simple as roto-translate the reference system of the embedding representation. In this sense, we propose to use to compute the fuzzy membership, and hence the fuzzy Jaccard similarity index, over these dimensions resulting from roto-translation, expecting that this "new perspective" will expose better the semantic content. So, defining $\mathbf{U} = \mathbf{M}$, where $\mathbf{M}$ is the transformation matrix, with $|\mathbf{M}| = d \times d$, we have that $\check{\mathbf{v}}_{w_i} \in \mathbb{R}^d$ is

$$\check{\mathbf{v}}_{w_i} = \mathbf{M} \cdot \mathbf{u}_{w_i} = \mathbf{M} \cdot \mathbf{W}^\top \cdot \mathbf{v}_{w_i} \qquad (11)$$

Thus yielding d-dimensional word and sentence embeddings.

Clustering and multivariate analysis can be applied to the whole embedding vocabulary or the subsets of the vocabulary introduced in Section 3.1. Apart from reducing the computational time, we did so to see if these subsets are sufficient to provide a helpful representation.

4 Experiments

In order to find the best solution in terms of word embedding matrix and universe matrix, we explored various possibilities. Then, to measure the goodness of our sentence embeddings, we leveraged a series of STS tasks and compared the results with the preceding models.

4.1 Word embeddings

For what concerns the word embeddings, we have decided to work with a selection of four models:

- Word2Vec, with 300-dimensional embeddings;
- GloVe, with 300-dimensional embeddings;
- fastText, with 300-dimensional embeddings;
- Sent2Vec, with 700-dimensional embeddings.

As shown by the word embedding models list, we are also employing a Sent2Vec sentence embedding model. The embedding matrix of this model can be used for word embeddings too. During the experiments, we focused on the universe matrix construction. For this reason, we relied on pretrained models for word embeddings, available on the web.

4.2 Universe matrices

The universe matrices we considered are divided into three buckets, as described in Section 3.2.

Clustering Universe matrices built using clustering leverage four different algorithms: k-Means, Spherical k-Means, DBSCAN and HDBSCAN.

We selected k-Means and Spherical k-Means because they usually lead to good clustering results; the latter was specifically designed for textual purposes, with low demand in time and computation resources. For all algorithms, we considered the same values for k (the number of centroids), which were 100, 1000, 10 000 and 25 000. For all the values of k, we performed clustering on different subsets of the vocabulary: k-Means was applied on the whole English vocabulary as well as to the top 100 000 frequently used words subset, while Spherical k-Means was applied to the subset of the first 50 000 frequently used words (in order to reduce computational time).

We also explored density-based algorithms (DBSCAN and HDBSCAN), which do not require defining in advance the number of clusters, using euclidean and cosine distance between the word embedding. For DBSCAN with euclidean distance, we varied the radius of the neighbourhood ε between 3 and 8 and worked over the same

two subsets considered for k-Means, while for cosine distance ε was between 0.1 and 0.55 and it was applied over the subset of the first 50 000 frequently used words (for computational reasons, as we did for Spherical k-Means). Concerning HDBSCAN, we varied the smallest size grouping of clusters in the set $\{2, 4, 30, 50, 100\}$ and the minimum neighbourhood size of core samples in the set $\{1, 2, 5, 10, 50\}$. We considered this latter density-based algorithm since basic DBSCAN happens to fail with high-dimensional data.

Identity This approach consists of using the identity matrix as the universe, in this way, the singletons we use to compute the fuzzy membership are the dimensions of the word embeddings, which corresponds to the learnt features. This is the most lightweight method as it just requires to compute the word embeddings of a sentence and then the fuzzy membership over the exact d dimensions.

Multivariate analysis We adopted the Principal Component Analysis (PCA) to get a rotation matrix to serve as a universe matrix to the SFBoW. In fact, through PCA, the d-dimensional word embedding vectors are decomposed along with the d orthogonal directions of their variance. These components are then reordered to decrease explained variance and represent our fuzzy semantic sets.

The principal component of the reduced word embedding matrix $\check{\mathbf{W}}$ are described by the matrix $\mathbf{T} = \mathbf{P}^\top \cdot \check{\mathbf{W}}$, where $\mathbf{P}$ is a $d \times d$ matrix whose columns are the eigenvectors of the matrix $\check{\mathbf{W}}^\top \cdot \check{\mathbf{W}}$. With our approach, the matrix $\mathbf{P}^\top$, sometimes called the *whitening* or *sphering transformation matrix*, serves as universe matrix $\mathbf{U}$. In this way, the SFBoW embedding of a word singleton becomes

$$\check{\mathbf{v}}_{w_i} = \mathbf{P}^\top \cdot \mathbf{u}_{w_i} = \mathbf{P}^\top \cdot \check{\mathbf{W}}^\top \cdot \mathbf{v}_{w_i} \qquad (12)$$

As for the clustering approach, we experimented with both the whole vocabulary and the most 100 000 used words.

4.3 Data

We evaluated our SFBoW through a series of reference benchmarks; we selected the STS benchmark series, one of the tasks of the International Workshop on Semantic Evaluation (SemEval)[2].

SemEval is a series of evaluations on computational semantics; among these, the *Semantic Textual Similarity* STS benchmark[3] (Cer et al., 2017) has become a reference for scoring of sentence embedding algorithms. All the previous models we are considering for comparison have been benched against STS; this is because the benchmark highlights a model capability to provide a meaningful semantic representation by scoring the correlation between model's and human's judgements. For this reason, and also to allow comparisons, we decided to evaluate SFBoW on STS.

We worked only on the English language, using the editions of STS from 2012 to 2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016). Each year, a collection of corpora coming from different sources has been created and manually labelled, in Table 1 is possible to have a reference in terms of support for each edition. Thanks to the high number of samples, we are confident about the robustness of our results.

Table 1: Support of the corpora of the STS benchmark series.

Edition	No. sentence pairs
STS 2012	5250
STS 2013	2250
STS 2014	3750
STS 2015	3000
STS 2016	1186
Total	15 436

To preprocess the input text strings, we lowecased each character and tokenised in correspondence of spaces and punctuation symbols. Then, from the resulting sequence, we retained only the tokens for which a corresponding embedding was found in the vocabulary known by the model. Finally, we calculated the SFBoW sentence embedding from the word embeddings of such tokens.

The samples constituting the corpora are pair of sentences with a human-given similarity score (the *gold labels*). The provided score is a real-valued index obtained averaging those of multiple crowd-sourced workers and is scaled in a $[0, 1] \in \mathbb{R}$ interval. The final goal of our work is to provide a model able to provide a score as close as possible

[2] https://aclweb.org/aclwiki/SemEval_Portal

[3] https://ixa2.si.ehu.eus/stswiki/index.php/Main_Page

to that of humans.

4.4 Evaluation approach

To assess the quality of our model, we used it to compute the similarity score between the sentence pairs provided by the five tasks, and we compared the output with the target labels. The results are computed as the correlation between the similarity score produced by SFBoW and the human one, using Spearman's ρ measure (Reimers et al., 2016). SFBoW employs fuzzy Jaccard similarity index (Zhelezniak et al., 2019) to compute word similarity.

To have terms of comparison, we establish a baseline through the most straightforward models possible, the average word embedding in a sentence, leveraging three different word embedding models: Word2Vec, GloVe and fastText. We also provide results from more complex models: SIF weighting (applied to GloVe), Sent2Vec, DynaMax (built using Word2Vec, GloVe and fastText) and Sentence-BERT.

All the embedding models except DynaMax and the baselines are scored using cosine similarity; DynaMax scores are obtained using fuzzy Jaccard similarity index.

5 Results

The results of the Spearman's ρ correlation in the STS benchmark of our SFBoW are reported in the last three rows of Table 2. The reported values belong to the FSBoW configurations that achieved the best score, among the variants we considered for the experiments, in at least one task.

FastText is the best among the four-word embeddings models, confirming the results of DynaMax. The best scores in terms of universe matrix are achieved either with Identity matrix or with PCA rotation matrix, highlighting how the features yield by word embeddings provide a better semantic content representation of sentences.

Clusterings results turned out to be very poor, independently of the starting embeddings. For this reason, we avoid discussing them.

As premised, we compare our results with three baseline models and other sentence embedding approaches, all reported in Table 2. The first group of scores is from the baselines, the second one is from other sentence embedding models and, finally, the last group is from our SFBoW model. Additionally, the best values in each column are highlighted in bold, while the second ones are underlined.

The key features about our model, which can be derived from the results, are the following:

- low number of parameters;
- faster inference time
- no training phase;
- results (in terms of ρ) comparable to similar models;
- fixed-size and easily re-usable embeddings.

About the number of parameters, we can notice that even if Sentence-BERT outperforms all the other models in every task, it relies on a much deeper feature extraction model and was trained on a much bigger corpus. Moreover, this model requires a considerably higher computational effort without an equally consistent difference in performances. BERT alone requires more than 100 million parameters just for its base version (and above 300 million for the large one), hence taking a lot of (memory) space, not to mention the amount of time necessary for the self-supervised training and the fine-tuning. On the other hand, non-parametric models (like SIF, DynaMax or SFBoW) or shallow parametric ones (Sent2Vec) require fewer parameters: just those for the embedding matrix $|\mathcal{V}| \times d$.

A similar discourse applies to inference speed. Even though Sentence-BERT achieves the best results on all tasks, SFBoW turns out to be four times faster at inferring the similarity, as can be noticed by the reported analysis times.

Being a non-parametric model, SFBoW does not require a training phase. It may require clustering the embeddings to build the universe matrix, but our experiments showed that clustering does not yield good results. Because of its simplicity, SFBoW can generally be easily deployed, requiring only the word embedding model to compute the sentence representation. Notice also that the SFBoW algorithm is agnostic to the word embedding model.

Regarding the results we obtained, compared to other models, SFBoW provided interesting figures: either considering the majority of tasks with higher Spearman's ρ rank or higher average score, it outperforms all the baselines, as well as SIF weighting and Sent2Vec. Finally, we see as our model performs closely to its predecessor, especially considering the weighted average of the results of the

Table 2: Comparison of results over the STS benchmark. SFBoW models are in the last block. Weighted averages are expressed as: *avg.*±*std.* Bold and underlined values represent, respectively, first and second best result of column. Inference time refers to to the time, in seconds, to carry out an evaluation on the entire STS corpus.

Model	**Results** (Spearman's ρ)						**Analysis time** (s)
	STS					**Total**	
	2012	**2013**	**2014**	**2015**	**2016**		
Word2Vec [a]	55.46	58.23	64.05	67.97	66.28	61.21±5.04	–
GloVe [a]	53.28	50.76	55.63	59.22	57.88	54.99±2.80	–
fastText [a]	58.82	58.83	63.42	69.05	68.24	62.65±4.20	–
SIF weighting [b]	56.04	<u>62.74</u>	64.29	69.89	70.71	62.84±5.54	–
Sent2Vec	56.26	57.02	65.82	74.46	69.01	63.21±7.13	–
DynaMax [c]	55.95	60.17	65.32	73.93	71.46	63.53±6.92	–
DynaMax [b]	57.62	55.18	63.56	70.40	71.36	62.25±5.85	–
DynaMax [d]	61.32	61.71	66.87	<u>76.51</u>	<u>74.71</u>	<u>66.71</u>±6.10	–
Sentence-BERT	**72.27**	**78.46**	**74.90**	**80.99**	**76.25**	**75.81**±3.27	218.3
SFBoW [d,e,f]	61.31	51.21	<u>67.47</u>	72.90	73.88	64.55±7.20	56.5
SFBoW [d,g,h]	<u>61.42</u>	51.36	66.44	72.74	73.72	64.32±7.00	56.8
SFBoW [d,g,i]	60.03	51.96	66.36	72.39	73.25	63.81±6.93	56.6

[a] Used as baseline. [b] Built upon a GloVe model for word embeddings. [c] Built upon a Word2Vec model for word embeddings. [d] Built upon a fastText model for word embeddings.
[e] Best average score. [f] Universe matrix is the identity matrix.
[g] Universe matrix is the PCA projection matrix. [h] Universe matrix is built from the English vocabulary.
[i] Universe matrix is built from the top 100 000 most frequent words.

single tasks. SFBoW bests out DynaMax in STS 2014 and gets almost the same results in STS 2012 (the difference is 0.01), which are the first two corpora in terms of samples; however, the difference in STS 2013 goes in favour of DynaMax.

About the comparison against DynaMax, it is worth underlining a few additional points. First of all, in both cases, fuzzy Jaccard similarity correlates better with human judgement as a measure of sentence similarity. Secondly, both models manage to achieve better results when using fastText word embedding, possibly underling that they lend better than other models at sentence level combination; the baseline performances also show this.

Finally, we remind that SFBoW generates embeddings with a fixed size, resulting in much easier applicability with respect to DynaMax.

6 Conclusion

With this work, we have proposed SFBoW, a refinement of the FBoW and DynaMax models for sentence embedding. Even if SFBoW does not achieve state-of-the-art performances on the considered STS benchmark, our solution performs comparably to its predecessor while enabling the possibility of re-usable embeddings as their dimension is fixed. Moreover, as can be seen from the results, it outperforms in most tasks all the other compared models except Sentence-BERT, without the need for specific training or fine-tuning on sentence similarity corpora and still being as lightweight and fast as possible. As a result, SFBoW seems a reasonable solution in low resources or constrained computational scenarios. In the future, we plan to investigate other clustering techniques and other methodologies for computing the universe matrix.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 252–263. The Association for Computer Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pages 81–91. The Association for Computer Linguistics.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 497–511. The Association for Computer Linguistics.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012, Montréal, Canada, June 7-8, 2012*, pages 385–393. The Association for Computer Linguistics.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*, pages 32–43. Association for Computational Linguistics.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. 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.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland. Association for Computational Linguistics.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 1–14. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *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*, pages 3111–3119.

Charles Egerton Osgood, George J Suci, and Percy H. Tannenbaum. 1958. The measurement of meaning. *American Journal of Sociology*, 63(5):550–551.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. 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, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 528–540. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1532–1543. ACL.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Nils Reimers, Philip Beyer, and Iryna Gurevych. 2016. Task-oriented intrinsic evaluation of semantic textual similarity. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 87–96, Osaka, Japan. The COLING 2016 Organizing Committee.

Nils Reimers and Iryna Gurevych. 2019. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Gerard Salton. 1971. *The SMART retrieval system: experiments in automatic document processing*. Prentice-Hall series in automatic computation. Prentice-Hall.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Wen-tau Yih, Xiaodong He, and Jianfeng Gao. 2015. Deep learning and continuous representations for natural language processing. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 6–8. The Association for Computational Linguistics.

Rui Zhao and Kezhi Mao. 2018. Fuzzy bag-of-words model for document representation. *IEEE Trans. Fuzzy Syst.*, 26(2):794–804.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. 2019. 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.