Sequence-based word embeddings for effective text classification

Bruno Guilherme Gomes[1], Fabricio Murai[1], Olga Goussevskaia[1], Ana Paula Couto da Silva[1]

Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
{brunoguilherme, murai, olga, ana.coutosilva}@dcc.ufmg.br

Abstract. In this work we present DiVe (Distance-based Vector Embedding), a new word embedding technique based on the Logistic Markov Embedding (LME). First, we generalize LME to consider different distance metrics and address existing scalability issues using negative sampling, thus making DiVe scalable for large datasets. In order to evaluate the quality of word embeddings produced by DiVe, we used them to train standard machine learning classifiers, with the goal of performing different Natural Language Processing (NLP) tasks. Our experiments demonstrated that DiVe is able to outperform existing (more complex) machine learning approaches, while preserving simplicity and scalability.

Keywords: Word embeedings · Logistic Markov Embedding · NLP.

1 Introduction

Word embedding techniques compute representations of words as vectors in a continuous space in order to capture some notion of similarity between them. More precisely, words from a corpus are mapped onto a low-dimensional Euclidean space, while preserving certain similarity properties of the input data. Learning good word representations has led to breakthroughs in several Natural Language Processing (NLP) tasks, such as document classification [11], sentiment analysis [7], hate speech detection [16], among others.

Embeddings techniques such as Word2Vec [10,11] and Glove [14] gained popularity for their performance in NLP tasks and for being easy to train. More recently, the generating effective embeddings using deep neural networks became possible through the larger availability of data and of GPU-based computational resources. Notable examples of these techniques are BERT [3] and ELMo [15].

Embedding techniques tend to represent related words, such as "check" and "bank", as points close to each other in space, as they are trained to reconstruct the context in which a word appeared. This characteristic is known as semantic similarity. Although embeddings trained from large corpora containing up to tens of billions of words are available for download on the Internet, difficulty to find pre-trained embeddings for less used languages and problems associated with polysemy (i.e., multiple meanings) can make it beneficial to train embeddings from data specifically related to the task at hand.

In this work, we present DiVe (Distance-based Vector Embedding), a new word embedding technique based on the Logistic Markov Embeddings, a Markovian model [7, chapter 3] originally designed to represent sequences of songs in a playlist [12]. A drawback of the original model is that it is not possible to shift from a relatively restricted universe of songs to the much larger universe of words due to scalability issues related to the computation of the so-called *partition function*. In essence, a partition function is a normalization constant used to ensure that the sum of the probabilities associated with each event of the sample space is one, given a set of observations. Therefore, each partition function is a sum over all the words in the vocabulary. As a first contribution, we use the negative sampling [10] method to approximate the partition function, making DiVe scalable for large datasets.

Second, we generalize LME to consider other distance metrics. Specifically, instead of using either the negative Euclidean distance or the cosine similarity, we investigate the performance of a convex interpolation between the two metrics. Third, we investigate benefits of using a single vs. a dual point model for DiVe. In language models, a "center" word is said to be surrounded by a context, even when the context appears strictly before or after the center word. In the dual point model, each word has two representations, one for when the word is in the center and another for when the word is part of the context, whereas in the single point model, the representation is the same in both cases.

Then, we compare DiVe to 5 word embedding baselines. All techniques are trained on one of 9 different datasets that together represent 6 different classification tasks (hate speech, user review, text polarity, question type, and subjective vs. objective text). From the embeddings locally generated by each technique, we trained standard machine learning classifiers to predict labels of the sentences that compose each of these textual datasets. Also, using the same datasets, we compare DiVe to two state-of-the-art classification techniques based on deep learning (DL) that make use of pre-trained embeddings. We show that DiVe (i) outperforms the five baselines on several datasets and (ii) yields comparable performance to the two DL methods at a much smaller computational cost.

The rest of this paper is organized as follows. §2 discusses existing work on word embeddings. In §3, we define the models and algorithms behind DiVe. §4 presents our experimental results. §5 concludes the paper. We provide an appendix in an external repository[1] with further details on the analytical model.

2 Related Work

The task of learning word embeddings has received a significant amount of interest in the last years. We discuss three fronts of research related to our work:
Shallow window-based methods: This body of works studies vector representations of words. The basis of these techniques lies in the local learning of the representations of words within the same context window. The authors of [13]

[1] `https://github.com/DiVeWord/DiVeWordEmbedding`

introduced a model that learns word vector representations using a simple neural network architecture for language modeling. *Word2vec* [10] is a more recent technique, based on a two-layered artificial neural network, trained to reconstruct linguistic contexts of words. Following a similar approach, *FastText* [6] presents an extension of Word2Vec by taking into account *information from subwords* to compose the representation of a word. Bayesian Skip-Gram [1] is another word embedding algorithm, based on a Bayesian neural network.

Statistical estimation of word representation: Statistical models have been widely used to tackle NLP tasks, such as part-of-speech [2] and sense disambiguation [16]. In terms of word representation, primarily, many papers sought to capture the similarity between words by the probability that they occur in a sequence [7]. Later, Bayesian models for the semantic representation of words have also been proposed [8,5]. Recently, following a similar perspective, we can highlight *GloVe* [14], which presents an efficient statistical model for grouping words together with their synonyms and allegories.

Pre-trained deep learning: On this research front there are architectural neural networks based on seq2seq, LSTM and encode-decode, which can be used in various tasks, such as machine translation, word embedding, sentiment analysis, and question answering. CoVe [9] is a model based on seq2seq (sequence-to-sequence) machine translation, whose learned representation considers the entire input sentence. ELMo [15] is a neural network based on a bi-directional language model (biLM), in which each word presents a contextualized representation. Word vectors are functions learned from the internal states of biLM, which is pre-trained on a large text corpus. Another technique based on biLM is BERT [3], which was also shown to perform well in the task of determining if one sentence follows another.

Finally, we point out the work of Globerson et al. [5] and LME [12], which are based on an approach similar to ours. One of the general aspects that distinguishes our work is how we estimate the partition function and compute distance in space. In this way, DiVe can be seen as an approximation approach to LME.

3 The DiVe Model

Our goal is to estimate a generative model for continuous word representation from sentences of words. Let $\mathcal{D} = \{s^{(1)}, \ldots, s^{(n)}\}$ be a set of sentences and $\mathcal{V}$ be the vocabulary (set of unique words) that composes sentences $s \in \mathcal{D}$. We define $s = (w_1, w_2, \ldots, w_m)$ as a sentence containing m words, where each word $w \in \mathcal{V}$. Hence, we want to obtain a language model from $\mathcal{D}$ that defines a probability distribution over sentences (i.e., maps a sentence s to a probability mass $\Pr(s)$).

A natural approach for modeling language is to decompose sentences into word-to-word transitions, where each word represents a state of a Markov Chain. The probability of a sentence, comprised by a sequence of adjacent words, is defined as the product of transition probabilities between consecutive words.

Using a first order Markov Chain, we can write the probability of sentence s as

$$\Pr(s) = \prod_{i=1}^{k} \Pr(w_i|w_{i-1}). \tag{1}$$

As usual, the conditional probability $\Pr(w_i|w_{i-1})$ is defined to be proportional to a function of the embeddings of the words that characterize the current state w_{i-1}, or context, and the next state w_i. In most embedding techniques (for example, [11,10]), each word $w \in \mathcal{V}$ has two vector representations, depending on whether it is used to encode the current or the next state. We refer to this as the dual point model. In this work, we also investigate a simpler variant of this model, called the single point model, where each word is represented by the same vector regardless of whether it is the current or the next state.

3.1 DiVe Single Point Model

In the single point model, we represent each word $w \in \mathcal{V}$ as a vector $\mathcal{X}(w) \in \mathbb{R}^d$ for some dimension d. We denote by $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ some similarity measure between two word vector representations. To obtain a valid conditional distribution, we define $\Pr(w_i|w_{i-1})$ as the normalized value of some non-linear transformation σ applied to the similarity between w_i and w_{i-1}:

$$\Pr(w_i|w_{i-1}) = \frac{\sigma(f(\mathcal{X}(w_i), \mathcal{X}(w_{i-1})))}{Z(w_{i-1})}, \quad \text{for } w_i \in \mathcal{V}, \tag{2}$$

where $Z(w_{i-1}) = \sum_{v \in \mathcal{V}} \sigma(f(\mathcal{X}(w_v), \mathcal{X}(w_{i-1})))$ is the partition function.

In this work we investigate three choices of functions for the non-linearity σ: sigmoid, tanh and exp. In addition, instead of measuring similarity by the angle between word vectors as usual [10,11,6], we investigate a more flexible way of measuring similarity based on a linear interpolation between the inner product and the negative square Euclidean distance.

First, note that we can express the dot product of vectors v and u in terms of their Euclidean distance $\|v-u\|^2 = (v-u)\cdot(v-u)$ and their norms:

$$\|v-u\|^2 = v\cdot v + u\cdot u - 2(v\cdot u) \Rightarrow v\cdot u = \frac{1}{2}(\|v\|^2 + \|u\|^2 - \|v-u\|^2). \tag{3}$$

On one hand, if we compute the similarity between two embeddings using the RHS of Eq. (3), we are using the dot product as the similarity measure. On the other hand, if ignore the norms of the embeddings, we recover the negative Euclidean distance (times the constant 1/2). Rather than choosing between the dot product or the negative Euclidean distance as the similarity measure f, we propose the use of a convex combination of both:

$$f(\mathcal{X}(w_i), \mathcal{X}(w_{i-1})) = -\frac{1}{2}\|\mathcal{X}(w_i) - \mathcal{X}(w_{i-1})\|^2 + \frac{\alpha}{2}\|\mathcal{X}(w_i)\|^2 + \frac{\alpha}{2}\|\mathcal{X}(w_{i-1})\|^2, \tag{4}$$

where $0 \leq \alpha \leq 1$. When α is 0, the similarity measure f is the negative Euclidean distance, and when α is 1, f is (twice) the inner product between word vectors.

We generalize Eq. (1) to consider the case where the context, in this case c_i, of word w_i is formed by the j previous words, i.e., $w_{i-1}, \ldots, w_{i-j}$. This results in a j-th order Markov Chain and thus, the probability of a sentence becomes

$$\Pr(s) = \prod_{i=j}^{k} \Pr(w_i|c_i) = \prod_{i=j}^{k} \frac{\sigma(f(\mathcal{X}(w_i), \mathcal{X}(c_i)))}{Z(c_i)}, \tag{5}$$

where we set $\mathcal{X}(c_i) = \sum_{j \in c_i} \mathcal{X}(j)/|c_i|$ to be the element-wise average of the embeddings of words in c_i. We have also conducted experiments setting $\mathcal{X}(c_i)$ to the element-wise maximum, but we obtained slightly inferior results.

We are now ready to define the cost function to be optimized as the negative likelihood of $\mathcal{D}$ given the embeddings $\mathcal{X}(w)$ for all $w \in \mathcal{V}$:

$$\text{NLL}(\mathcal{D}) = -\sum_{s \in \mathcal{D}} \sum_{i=j}^{k} \log \Pr(w_i|c_i) = -\sum_{s \in \mathcal{D}} \sum_{i=j}^{k} \left[\log \sigma(f(\mathcal{X}(w_i), \mathcal{X}(c_i))) - \log Z(c_i)\right].$$

3.2 DiVe Dual Point Model

In the previous section, we described the single point model, that represents each word $w \in \mathcal{V}$ as a d-dimensional vector $\mathcal{X}(w)$. This model has two key limitations. First, natural choices for a similarity function f between two vectors are symmetric and, therefore, even if there are several transitions from w_i to w_j in the corpus $\mathcal{D}$, and no transitions in the opposite direction, the transition probabilities estimated by the model will be the same in both directions. Second, the representation of words can undergo drastic modifications at each stage of learning, making it more difficult to find good representation of words in space.

To overcome these issues, we also consider a dual point model, where each word w_i is represented as a vector pair $(\mathcal{I}(w_i), \mathcal{O}(w_i))$. We call $\mathcal{I}(w_i)$ the "entry vector" of word w_i, and $\mathcal{O}(w_i)$ the "exit vector". The cost function becomes

$$\text{NLL}(\mathcal{D}) = -\sum_{s \in D} \sum_{i=j}^{k} \log \Pr(w_i|c_i) = -\sum_{s \in D} \sum_{i=j}^{k} \left[\log \sigma(f(\mathcal{O}(w_i), \mathcal{I}(c_i))) - \log Z(c_i)\right].$$

3.3 Estimating the Partition Function

One of the limitations of LME is the cost of computing the partition function $Z(c)$ exactly for a context c [4]. Since during parameter optimization this computation must be performed several times for each training iteration and at least once for each different context, the resulting complexity is $\text{O}(|\mathcal{D}||\mathcal{V}|)$. To address this issue, we resort to negative sampling [11] to approximate the partition function and estimate model parameters more efficiently. The resulting complexity is $\text{O}(|\mathcal{D}|k)$, where k is a constant equal to the number of negative samples drawn for each word in $\mathcal{D}$. To compensate for highly unbalanced word frequencies, we adopt the heuristic of sampling a word w in proportion to $\pi_w^{3/4}$ where π_w is the word frequency of w in the corpus.

Using the negative sampling method, the term corresponding to the log of the partition function in Eqs. (6) and (6) is replaced by a sum over the negative instances $\mathcal{V}'$ that were sampled according to the heuristic described above. In the single point model, the new cost function is given by

$$\text{NLL}_{ns}(\mathcal{D}) = -\sum_{s \in D} \sum_{i=j}^{k} \left[\log \sigma(f(\mathcal{X}(w_i), \mathcal{X}(c_i))) - \sum_{v \in \mathcal{V}'} \log \sigma(-f(\mathcal{X}(w_v), \mathcal{X}(c_i))) \right].$$

The corresponding equation for the dual point model is analogous.

Finally, we found that the stochastic gradient algorithm finds a good solution for approximating cost functions for the single and the dual point models. In order to enable the replication of the results in this paper, all the code used in this work, including the baselines is available in a public repository[2].

4 Experiments and Results

4.1 Experimental Setup

We now conduct an experimental study of DiVe, comparing its performance to state-of-the-art word embedding techniques on text classification tasks.

We use the performance of models trained for text classification as a proxy to evaluate the quality of the embeddings obtained by DiVe and by the word embedding baselines: GloVe, Word2vec, fastText, Bayesian Skip Gram and deep learning baselines: ELMo and BERT. We use 9 publicly available datasets:

- **Customer reviews (CR)**: A dataset for binary sentiment classification based on user reviews of 5 products.
- **Hate Speech Twitter Annotations (HSTW)**: A collection of tweets labeled according to 3 categories: sexism, racism, neutral.
- **Polarity of Opinion (PO)**: This data was extracted from Rotten Tomatoes webpages, with reviews marked as "fresh" (positive) and "rotten" (negative).
- **Question Type Classification (QTS)**: This dataset contains questions asked by users, labeled in 6 different categories.
- **Subjectivity and objectivity of sentences (SUBJ)**: A set of sentences containing at least 10 words and labeled as either "subjective" or "objective".
- **IMDB reviews (IM) and (SIM)**: Datasets with large (IM) and small (SIM) number of movie and TV show reviews.
- **Yelp reviews (YR)**: Dataset with sentences from user reviews, about restaurants and bars, labeled with positive or negative sentiment.
- **Amazon reviews (AR)**: A set of sentences labeled with positive or negative sentiment, extracted from Amazon product review.

Table 1 lists the vocabulary size of each dataset, many of them used as benchmarks in prior works [16,8]. We refer to the datasets with $\leq 40K$ words as "small", and as "large" otherwise.

[2] `https://github.com/DiVeWord/DiVeWordEmbedding`

Table 1: Datasets used in classification tasks ($|V|$ is the vocabulary size).

acron.	description	$\|V\|$	# words	acron.	description	$\|V\|$	# words
AR	User product review	1 741	5 275	QTS	Question Answering	16 504	30 134
CR	User review polarity	5 176	33 665	SIM	Movie and TV Review	2 933	7 471
HSTW	Hate speech detect	23 739	155 804	SUBJ	Subjectivity and objectivity	20 745	121 366
IM	Movie and TV Review	74 337	3 124 867	YR	Food review polarity	1 919	5 563
PO	Sentence polarity	18 179	114 485				

For the word embedding classification task, we consider 8 "shallow" classifiers implemented on scikit-learn[3]: probabilistic models Logistic Regression (LR), Quadratic Discriminant Analysis (QDA), Linear Discriminant Analysis (LDA) and Naive Bayes (NB); structural models Support Vector Classification (Linear-SVM) and K-Nearest Neighbors (KNN); the ensemble model Random Forest (RF); and a Neural Network (NN). Moreover, we used two deep learning techniques as baselines: BERT and ELMo (see our repository for setup details).

The experiment setup is as follows. For each combination of dataset, word embedding technique and classifier, we use 5-fold cross-validation by learning the word embeddings on 4 folds in an unsupervised fashion, then training a classifier using these embeddings and the labels associated with each sentence, and finally testing on the left-out fold. We then take the performance to be the average weighted F_1 score over the 5 folds. For the deep learning baselines, we performed the 5-fold cross-validation and average weighted F_1 score for evaluation.

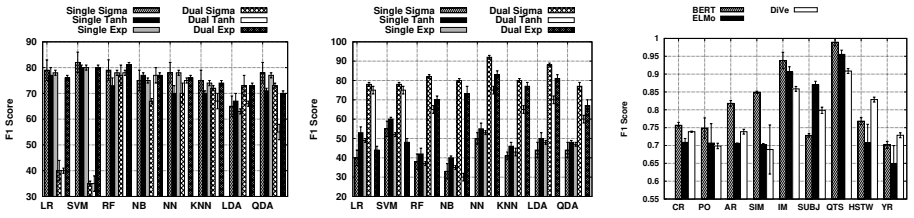
Ideally, we would also use cross-validation to jointly optimize the hyperparameters. However, due to the computational demands of running experiments with several large datasets, number of the combinations of embedding techniques and classifiers and the cost of tuning the deep networks, we used fixed values for the hyperparameters. For a fair comparison, we fixed the number of dimensions and context size respectively to 400 and 5 to train DiVe, Word2Vec, Glove, Bayesian Skip Gram and FastText. For BERT and ELMo we did not change the default network settings to represent text, with 1024 dimensions.

It is clear that the quality of the learning representations plays a major role in the classification performance. Since our interest here is to evaluate the embeddings produced by each technique, we argue that not tuning the hyperparameters of the classifiers is not a major problem. In fact, this allows us to better evaluate the robustness of the resulting embeddings.

4.2 Comparison of DiVe's variants

We compare Dive's single and dual point models while keeping the dimension of the embeddings fixed. Note that, in the **dual point** model, each word is represented by twice as many numbers as in the **single point** model. Therefore, we expect the former model to yield better performance in more complex tasks, but also to require more training data. We also consider the impact of the choice of the activation function – **sigmoid()**, **tanh()** and **exp()**.

[3] `http://scikit-learn.org/stable/index.html`



(a) YR (small) dataset (b) SUBJ (large) dataset (c) Deep-learning baselines.

Fig. 1: Comparing Dive variants using F1 accuracy.

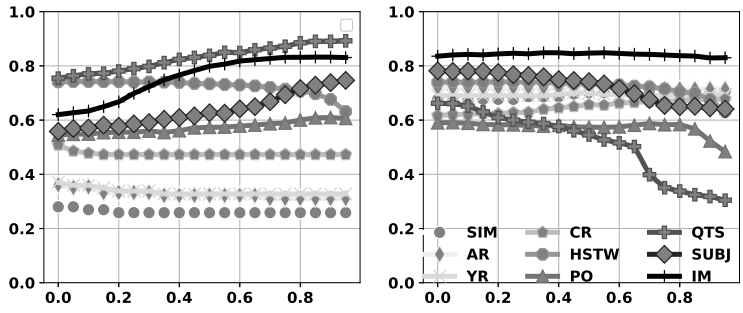


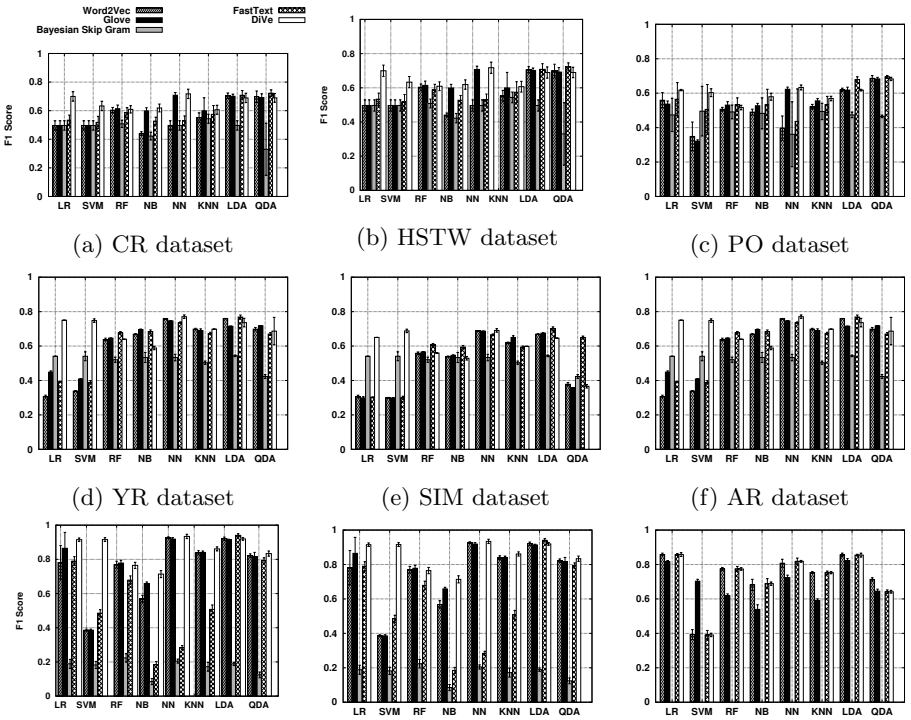
Fig. 2: Impact of DiVe's hyperparameter α (left: dual-point; right: single-point).

We compared the performance of the six models on each dataset. The results were very consistent among "small" datasets and among "large" ones. Hence, we present the results for two representative cases, YR (small) and SUBJ (large).

Figures 1a and 1b show the results for the YR and the SUBJ datasets, respectively. The groups in the x-axis indicate the classifier. And, within each group, a bar corresponds to one of DiVe's variants. The height of each bar is the average F_1 score and the whiskers represent 95%-confidence intervals. In general, we observe that the single point model significantly outperforms the dual counterpart on the small dataset, and among the single point variants, the sigmoid function yields the best results. Conversely, the dual point model significantly outperforms the single counterpart on the large dataset, but among the dual point variants, the sigmoid function is still the best choice. For this reason, in the next experiments we fix the activation function to be the sigmoid(.).

4.3 Analysis of parameter α in similarity function f

The similarity function f, defined in (4), is an interpolation between the negative Euclidean distance ($\alpha = 0$) and the inner product ($\alpha = 1$). In this section, we use the experimental setup described in Section 4.1 to investigate the impact of α on the tasks' performance. More precisely, we vary α from 0 to 1 in increments of 0.05 and compute the resulting F_1 score. As indicated before, we fix the



(a) CR dataset (b) HSTW dataset (c) PO dataset

(d) YR dataset (e) SIM dataset (f) AR dataset

(g) SUBJ dataset (h) QTS dataset (i) IM dataset

Fig. 3: Embeddings' performance on text classification.

activation function f to be the sigmoid(). Figures 2 (left) and 2 (right) compare results of DiVe Single and Dual models. In both cases we observe a large variation in terms of F_1 depending on α. For example, for the QTS dataset, the F_1 score has almost 30% variation for the Single Point model, and 10.5% variation for the Dual Point model, and for PO dataset 12% for Single Point and almost 10% for Dual Point. This shows that α can significantly influence an estimator's accuracy, for example, in some datasets the best embedding are obtained when $\alpha = 1$ can also lead to very poor results (see IM single point model). On the other hand, $\alpha = 0$ is not ideal either (see subj with dual point model). Then, we believe of setting $\alpha = 0.5$ yields a good trade-off between performance and simplicity, and it avoids additional hyper-parameters.

4.4 Performance of classifiers with trained embeddings

Now, we compare the quality of the embeddings obtained with DiVe to Word2Vec, Glove, Bayesian SkipGram and FastText techniques. The embeddings were trained on the specific dataset whose sentences we want to classify.

The results for each dataset are shown in Figures 3a–3i. In Figures 3a, 3b and 3c, we analyze the performance of text classification from user reviews, hate

speech detection and sentence polarity, respectively. DiVe yields higher F_1 scores than the baselines for nearly all classifiers. DiVe's performance is also less variable across classifiers than the other embedding techniques. In particular, other embeddings often result in a poor performance when combined with SVM (e.g., Fig. 3c), which does not occur with DiVe. Some of these issues with SVM could be circumvented with appropriate choices of nonlinear kernels, but we emphasize that the focus of this work is on evaluating the quality of the embeddings.

The small datasets consist of user reviews extracted from popular websites. The results obtained for them are shown in Figures 3d, 3e and 3f. We observe that DiVe presented higher F_1 with almost all classifiers.

In Figures 3g and 3h, we observe once again that DiVe's performance varies less across classifiers than that of the other techniques and that SVM can yields poor results. Figure 3i shows the results for the IM dataset, which consists of movie and TV show reviews. Overall, Word2Vec and FastText achieved the best performances. However, with exception of the QDA classifier, DiVe's embeddings resulted in very similar F_1 scores. On this dataset, all embedding techniques, except for GloVe, suffered with the SVM issues described above.

4.5 Performance of classifiers with pre-trained embeddings

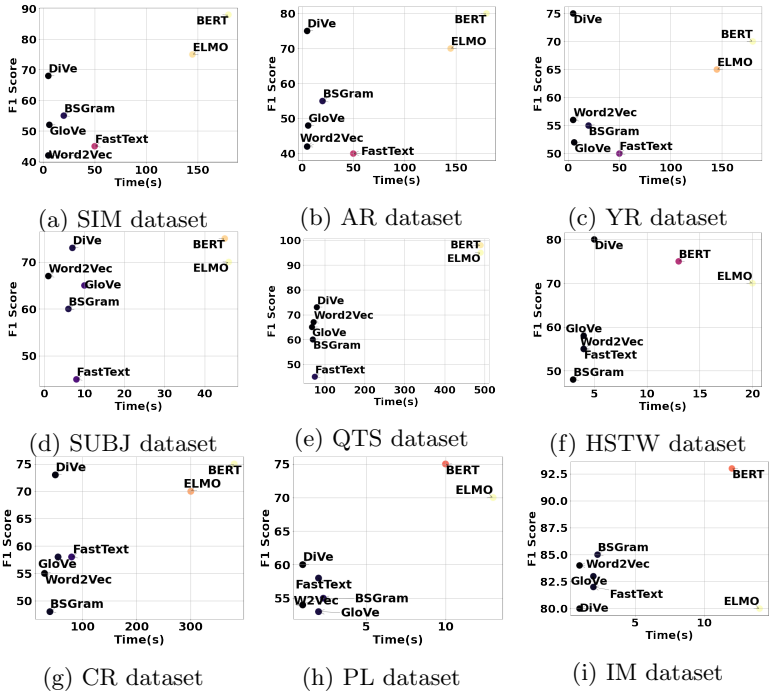
In this section, we evaluate results of deep techniques ELMo[4] and BERT[5]. We used these baselines as pre-trained embeddings, as recommended in the literature [15,3]. They were trained on a large dataset and used for classification tasks. Yet, prediction using either technique is very computationally expensive. In some cases, several hours of GPU/TPU processing were needed.

We compare the performance of the deep learning techniques with a simple Logistic Regression classifier trained from DiVe's embeddings when $\alpha = 0.5$. Figure 1c summarizes the results obtained using both techniques on all 9 datasets. DiVe outperforms ELMo in 4 classification tasks (CR, AR, HSTW and YR) and BERT in 3 classification tasks (SUBJ, HSTW and YR).

We emphasize that both BERT and ELMo have approximately 100 million parameters, thus requiring much longer training times than DiVe. For each technique, the average time of 5 training sessions carried out in each dataset, on a computer with an Intel Xeon CPU@2.40GHz, 128G of RAM.

In order to put both time requirements and performance into perspective, in Figure 4, we present scatterplots of these dimensions for each dataset. We state that one method "dominates" the other on a dataset when it appears above (better performance) and to the left (smaller training time) of the latter. We observe that while DiVe often dominates other shallow methods, no other method – either shallow or deep – dominates DiVe on any of the datasets. Furthermore, the F1 score achieved by DiVe is almost always close to that achieved by BERT and ElMo (except in the PL dataset) and, in some cases, even superior to that (see YR and HTSW datasets).

[4] `https://allennlp.org/elmo`
[5] `https://github.com/google-research/bert`



(a) SIM dataset (b) AR dataset (c) YR dataset

(d) SUBJ dataset (e) QTS dataset (f) HSTW dataset

(g) CR dataset (h) PL dataset (i) IM dataset

Fig. 4: Scatterplots of training time vs. F1 for all techniques on each dataset.

Finally, we conclude that, even though DiVe is a relatively simpler technique and easier to train than the state-of-the-art deep learning solutions, it was able to outperform these more complex techniques.

5 Conclusion

In this work we presented DiVe, a novel word embedding technique based on a variation of the Markovian statistical model. In order to address the scalability problems that arise due to the cost of computing the partition function, we proposed a sampling approach to approximate the latter. Moreover we evaluated a new way of measuring similarity between word vectors, based on a linear interpolation between the inner product and the square Euclidean distance function. Through extensive experiments we demonstrated the efficiency of DiVe on 9 datasets that represent 6 different text classification tasks: hate speech, user review, text polarity, question type, and subjective and objective text. Finally, using the obtained embeddings, we trained shallow and deep machine learning classifiers to predict labels of the sentences that compose each of these textual datasets. DiVe outperformed existing approaches in several tasks.

References

1. Brazinskas, A., Havrylov, S., Titov, I.: Embedding words as distributions with a bayesian skip-gram model. In: COLING (2018)
2. Cheng, J., Druzdzel, M.J.: AIS-BN: an adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. J. Artif. Intell. Res. **13** (2000)
3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)
4. Figueiredo, F., Ribeiro, B., Almeida, J.M., Faloutsos, C.: Tribeflow: Mining & predicting user trajectories (2015)
5. Globerson, A., Chechik, G., Pereira, F., Tishby, N.: Euclidean embedding of co-occurrence data. J. Mach. Learn. Res. **8** (2007)
6. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext.zip: Compressing text classification models. CoRR **abs/1612.03651** (2016), http://arxiv.org/abs/1612.03651
7. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to NLP, Computational Linguistics, and Speech Recognition (2009)
8. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: The 49th Annual Meeting of the Association for Computational Linguistics (2011)
9. McCann, B., Bradbury, J., Xiong, C., Socher, R.: Learned in translation: Contextualized word vectors. In: Advances in Neural Information Processing Systems (2017)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR **abs/1301.3781** (2013)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (2013)
12. Moore, J.L., Joachims, T., Turnbull, D.: Taste space versus the world: an embedding analysis of listening habits and geography. In: ISMIR (2014)
13. Okita, T.: Neural probabilistic language model for system combination. In: COLING (2012)
14. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP (2014)
15. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: NAACL-HLT (2018)
16. Xia, Y., Cambria, E., Hussain, A., Zhao, H.: Word polarity disambiguation using bayesian model and opinion-level features. Cogn. Comput. **7**(3) (2015)