Fusing Vector Space Models for Domain-Specific Applications

Laura Rettig
eXascale Infolab
University of Fribourg
Fribourg, Switzerland
laura.rettig@unifr.ch

Julien Audiffren
CMLA, CNRS, ENS Paris-Saclay
Paris-Saclay University
Cachan, France
audiffren@cmla.ens-cachan.fr

Philippe Cudré-Mauroux
eXascale Infolab
University of Fribourg
Fribourg, Switzerland
philippe.cudre-mauroux@unifr.ch

***Abstract*—We address the problem of tuning word embeddings for specific use cases and domains. We propose a new method that automatically combines multiple domain-specific embeddings, selected from a wide range of pre-trained domain-specific embeddings, to improve their combined expressive power. Our approach relies on two key components: 1) a ranking function, based on a new embedding similarity measure, that selects the most relevant embeddings to use given a domain and 2) a dimensionality reduction method that combines the selected embeddings to produce a more compact and efficient encoding that preserves the expressiveness. We empirically show that our method produces effective domain-specific embeddings that consistently improve the performance of state-of-the-art machine learning algorithms on multiple tasks, compared to generic embeddings trained on large text corpora.**

***Index Terms*—Word Embeddings, Dimensionality Reduction, Similarity Measure**

I. Introduction

Word embedding techniques such as word2vec [1] have become a key building block of many NLP applications. These techniques capture the semantic similarities between linguistic terms based on their distributional properties from large textual contents and allow to easily represent words or phrases in a low-dimensional vector space that can be leveraged by downstream applications (e.g., language translation or sentiment analysis).

As training word embedding models is computationally intensive and requires large text corpora, pre-trained models such as those provided by Facebook Research[1] or Google[2] are widely used today. These models are trained on very large unlabeled text corpora of billions of words and provide high-quality embeddings for a variety of languages.

Despite the convenience they bring, using such readily-available, pre-trained models is often suboptimal in vertical applications [2], [3]; as these models are pre-trained on large, non-specific sources (e.g., Wikipedia and the Common Crawl for FastText, news articles for Google's word2vec models), they often cannot capture important semantic information from specific sub-domains or applications. Indeed, many words carry specific meaning depending on their context, which can be difficult to capture from generic or encyclopedic contents only. In addition, particular domains may use specific vocabulary terms that are not present in pre-trained embeddings.

Retraining word embeddings for a specific context is possible, though it is also extremely costly, both in terms of computational power and in the availability of large quantities of text that have to be fed into the model. Instead, we suggest in this paper a third solution combining the convenience of pre-trained embeddings with the effectiveness of dedicated models. The main intuition behind our method is to leverage a collection of domain-specific embeddings and to efficiently *combine* them in order to capture the peculiarities of a given application domain as closely as possible. While sensible, this approach also raises two new challenges that have to be tackled: i) How to automatically select the most adequate embeddings from a library of pre-trained embeddings for a specific task and ii) how to efficiently and effectively combine different embeddings in order to obtain high-quality embeddings suitable for the task at hand.

We take on these challenges through a new method introduced in this paper that we call *Embedding Fusion*: We solve the first challenge by introducing a ranking technique based on a comparative analysis of the language used in the task and in the embeddings text corpora. We tackle the second problem by using dimensionality reduction methods that combine the selected embeddings to produce a more compact and efficient – yet similarly expressive – encoding.

Our main contributions are hence as follows:

1) To the best of our knowledge, we are the first to propose the idea of *dynamically selecting* and *combining* several word embeddings to better capture a particular application domain;
2) We introduce a new technique to rank word embeddings, based on their relevance to a particular domain;
3) We describe several techniques to combine a set of domain-specific embeddings into a new encoding better suited for the task at hand;
4) Finally, we demonstrate through a comprehensive empirical evaluation on multiple test corpora that the use of our approach – i.e., the combination of automatically selected and relevant domain-specific embeddings – leads

[1]fastText pre-trained word vectors: https://fasttext.cc/docs/en/crawl-vectors.html

[2]word2vec models from Google's Code Archive: https://code.google.com/archive/p/word2vec/

to consistently improved results compared to the use of generic embeddings.

II. METHOD

As discussed above, the key idea of *Embedding Fusion* is to replace a general-purpose embedding by a carefully constructed combination of domain-specific embeddings. Our method relies on two main ingredients, namely i) a principled algorithm to automatically select the most relevant embedding(s), depending on the task at hand, and ii) an efficient dimensionality reduction algorithm that combines the previously selected embeddings into a single embedding of fixed dimension d. These two contributions are presented in Sections II-A and II-B, respectively.

A. Ranking Embeddings

In the following, we use the terms *embedding* or *encoding* interchangeably to refer to word embeddings. Encodings are characterized by their two core features: the learning algorithm used for training (see e.g. [4], [5]), and the text corpus used as training set (such as Wikipedia). While the choice of the learning algorithm has a significant impact on the resulting embedding, in this work we are more interested in the influence of the associated text corpus and more precisely the characteristics of the corpus (topic, type of language, etc.). Therefore, throughout this paper, we use the exact same algorithm for learning *all* text encodings, namely word2vec [4] with the CBOW model and a window size of 5, outputting embeddings of dimension $d = 300$, which is a common choice in state-of-the-art word embeddings. The text corpora are then the only difference between the embeddings and we use them to define the notions of similarity and ranking of pairs of encodings.

a) Domain-Specific Embeddings: We say that an encoding is domain-specific if its associated text corpus is almost exclusively constituted of documents pertaining to a specific topic. Examples of domain-specific embeddings that we use in our subsequent evaluations include *drugs*, an embedding constructed on Wikipedia articles relating to pharmaceuticals, or *twitter*, an embedding constructed from a series of tweets. In theory, the use of domain-specific embeddings suited to the task at hand entails significant advantages: the presence of specialized words in the vocabulary such as colloquial expressions and a better encoding of homonyms, e.g., when given words may have a distinct meaning in a specific domain, such as *calculus* that refers to kidney stones in a medical context rather than to a branch of mathematics.

However, one significant drawback of domain-specific embeddings is that by definition, each encoding may only be suitable for a narrow range of tasks. To solve this issue we provide a generic solution that can automatically analyze multiple domain-specific contents and evaluate their relevance to a given task. More precisely, we propose a new approach that uses both i) a large library of pre-trained domain-specific embeddings and ii) a new similarity function to automatically score the relevance of each encoding of the library with respect to the corpus of the target task, resulting in a ranking of their usefulness. The similarity function is introduced below.

b) Notations: Let $\left(\mathcal{E}_i\right)_{i=1}^N$ denote a collection of N different domain-specific embeddings. For each encoding $\mathcal{E}_i$, we denote as $\mathcal{T}_i$ its corresponding text corpus containing $|\mathcal{T}_i|$ words. For any word $w \in \mathcal{T}_i$, let $\mathcal{E}_i(w)$ designate the embedding of w in $\mathcal{E}_i$ and $\#\mathcal{T}_i(w)$ the number of occurences of w in $\mathcal{T}_i$. In the following and without loss of generality, all the embeddings are assumed to have the same dimensionality $d = 300$. Finally, let $\mathcal{T}$ be the text corpus of the target task and $\mathcal{E}$ be a general-purpose embedding method.

c) Text corpus as distribution: In order to define a similarity between text corpora, we proceed as follows. First, we associate with each word w in each corpus $\mathcal{T}_i$ a weight $\alpha_i(w)$ that reflects its relative frequency:

$$\alpha_i(w) = \frac{\#\mathcal{T}_i(w)}{\sum_{j=1}^N \#\mathcal{T}_j(w)} \frac{\sum_{j=1}^N |\mathcal{T}_j|}{|\mathcal{T}_i|} \tag{1}$$

$\alpha_i(w)$ can be seen as the ratio between the frequency of the word w in $\mathcal{T}_i$ and its frequency among the collection of all text corpora. Therefore, $\alpha_i(w)$ increases with how *specific* w is to $\mathcal{T}_i$; a word w that is frequent in $\mathcal{T}_i$ but rare in the other text corpora will have a large weight $\alpha_i(w)$. The idea is that such a word might be representative of the domain specific to $\mathcal{T}_i$ and therefore is associated with a higher weight.

Then, each text corpus is seen as a distribution over the space of embeddings of words, where the probability p of each word w is directly proportional to its weight. Formally, given a *generic* encoding $\mathcal{E}$ (i.e., a general-purpose embedding used to represent all words w), $\mathcal{T}_i$ induces a discrete distribution P_i over $\mathbb{R}^d$, such that

$$\forall w \in \mathcal{T}_i, \quad P_i(\mathcal{E}(w)) = \frac{\#\alpha_i(w)}{\sum_{w' \in \mathcal{T}_i} \#\alpha_i(w')}. \tag{2}$$

The reasoning behind this definition is twofold. First, words with large weights should give insight into the corpus topic (1), while words with small weights may be the result of errors (e.g., a document that is wrongly associated with the corpus). Therefore, we set the probabilities to reflect this relative importance. Second, we use embeddings instead of raw words, as embeddings naturally include a notion of similarity between words [5]. The generic embedding is used here to provide a common encoding for all words of each text corpus, as opposed to embeddings that are specific to a topic.

d) Similarity between text corpora: We define the similarity between two text corpora as the similarity between their respective probability distributions over $\mathbb{R}^d$, using the RBF kernel. In other words,

$$\begin{aligned} s(\mathcal{T}_i, \mathcal{T}_j) &\doteq s(P_i, P_j) \\ &= \sum_{w \in \mathcal{T}_i} \sum_{w' \in \mathcal{T}_j} P_i(\mathcal{E}(w)) P_j(\mathcal{E}(w')) \times \\ &\qquad \exp\left(-\frac{\|\mathcal{E}(w) - \mathcal{E}(w')\|_2^2}{\sigma^2}\right) \end{aligned} \tag{3}$$

where σ is the bandwidth of the RBF kernel (in this work we used $\sigma = 0.01$), and $P_i(\mathcal{E}(w))$ is defined in (2). The exponential term represents the similarity between the words; words with closely related meanings will hence have a similarity close to one due to the combined use of a common embedding $\mathcal{E}$ with a smooth kernel.

Remark. *While in theory* (3) *includes all the words of both corpora, it is generally-speaking intractable in practice due to the size of the vocabulary of each corpus. Moreover, since rare words are likely to be unrelated to the text corpus topic and have little influence on the similarity metric (due to their negligible probability weights), we choose to use only the top* 500 *most frequent non-stop words of each corpus when computing similarities. In our experiments, this value achieved the best trade-off between reducing the computation time and capturing most of the corpus characteristics.*

e) Ranking Embeddings: Finally, the ranking of the embeddings is derived from the similarity metric defined in (3). For a task associated to a text corpus $\mathcal{T}$, we define α by

$$\alpha_i(w) = \begin{cases} \dfrac{\#\mathcal{T}(w)}{\sum_{j=1}^{N} \#\mathcal{T}_j(w)} \dfrac{\sum_{j=1}^{N} |\mathcal{T}_j|}{|\mathcal{T}|} & \text{if } \exists i \in [1, N] \\ & \text{s.t. } w \in \mathcal{T}_i \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Note that compared to (1), $\alpha(w)$ is the ratio between the frequency of the word w in $\mathcal{T}$ and its frequency among the concatenated text corpora of the embeddings, not including $\mathcal{T}$. Consequently, a word that is present in $\mathcal{T}$ but not in any $\mathcal{T}_i$ is ignored, as its relative frequency cannot be computed. Then, $P(\mathcal{E}(w))$ and $s(\mathcal{T}_i, \mathcal{T})$ are defined using (4), (2) and (3). Finally, each encoding is ranked according to the similarity between its corpus and the task corpus:

$$\forall 1 \leq i \leq N, \qquad s(\mathcal{E}_i, \mathcal{T}) \doteq s(\mathcal{T}_i, \mathcal{T})$$

The performance of this ranking approach is evaluated in Section III.

B. Fusing Word Embeddings

While in some cases a domain-specific encoding is able to outperform the general-purpose embedding, our experiments show that we can further improve the performance by *merging* multiple domain-specific embeddings (see Section III). Therefore, the second step of our approach is to combine top-k embeddings from the previous ranking into a new encoding suited for the task at hand. We distinguish four different families of embedding combination algorithms: concatenation, averaging, PCA, and autoencoders, which are discussed in their respective subsections below.

1) Concatenation: The first approach $\mathcal{E}_{concat}$ simply uses the cartesian product of the different embeddings, i.e.

$$\mathcal{E}_{concat}(\cdot) = \big(\mathcal{E}_1(\cdot), \ldots, \mathcal{E}_k(\cdot)\big) \in \mathbb{R}^{kd}.$$

By definition, $\mathcal{E}_{concat}$ retains all semantic information carried by the word in the top-k embeddings. However, this is done at the cost of an increased dimensionality (in this case, kd). Since many of these dimensions may yield redundant – or non-relevant – semantic meaning, this increased dimensionality may reduce the efficiency of learning algorithms leveraging $\mathcal{E}_{concat}$ [6], [7]. Therefore, the three following algorithms preserve the original dimensionality d.

2) Averaging: Recent approaches have shown that $\mathcal{E}_{avg}$ constructed from the arithmetic mean between pairs of vectors drawn from distinct $\mathcal{E}_i$ performs comparably to more complex combination methods under the assumption that the two embeddings are approximately orthogonal [22]. This method preserves the original dimensionality d, however at the cost of a greater loss of information than the following two approaches, which use dimensionality reduction on $\mathcal{E}_{concat}$ to obtain a final embedding of dimension d.

3) PCA: The gold standard for linear methods of dimensionality reduction is the Principal Component Analysis (PCA) [8]. In our setting, we define $\mathcal{E}_{pca}$ as the result of the PCA algorithm applied on the covariance matrix of the $\mathcal{E}_{concat}$ embedding of all words in the target corpus $\mathcal{T}$. We set the target dimension of $\mathcal{E}_{pca}$ to $d = 300$.

4) Autoencoders: The last family of embeddings is obtained by using unsupervised deep learning-based dimensionality reduction algorithms, namely autoencoders [9], generalized autoencoders [10] and variational autoencoders [11]. By applying these algorithms to the concatenated encoding of the target task $\mathcal{E}_{concat}\big(\mathcal{T}\big)$ with a target dimension of $d = 300$, we obtain $\mathcal{E}_{auto}$, $\mathcal{E}_{G\,auto}$ and $\mathcal{E}_{V\,auto}$, respectively. These non-linear methods have been shown to produce interesting results when applied to embedding-related problems (e.g. [12]).

III. Evaluation

In this section we study in detail the performance of *Embedding Fusion* and in particular the pros and cons of the dimensionality reduction methods discussed in Section II-B. To evaluate the embeddings produced by *Embedding Fusion*, we use the common task of sentiment analysis on user reviews [13], where the learning algorithm is tasked with finding whether the author of a review expresses a positive or negative sentiment towards the object of their review.

a) Sentiment Analysis on Vector Space Models: As noted by Guggilla et al. [14], both recurrent neural networks (RNN) and convolutional neural networks (CNN) achieve comparable performance on sentiment analysis (see [15] for an in-depth review). In this work we selected a CNN architecture for our evaluation as an examplary task. The network takes as input the embeddings of the words of one review and outputs the probability of the review to express a positive sentiment. The architecture of the CNN used in the experiments is summarized in Fig. 1. For regularization, we used dropout [16] with $p = 0.3$ after both the maxpooling layer and the dense layer. The network was trained using Adagrad [17] with batches of 100 sentences over 100 epochs and a learning rate of 0.01.

b) Annotated Review Datasets: To evaluate *Embedding Fusion*, we used annotated datasets where each review is either labeled as positive/negative or associated with a numeric rating which can be split into positive (upper half of the scale) or negative (lower half of the scale). We use the following openly

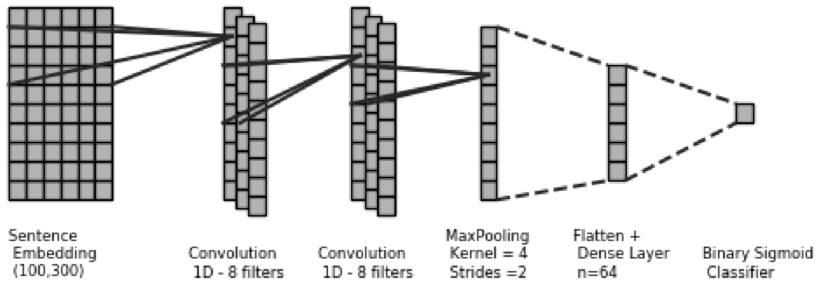

Fig. 1. CNN Architecture used in the evaluation process. Each of the convolutional layers as well as the dense layer is followed by a RELU activation function.

TABLE I
COMPOSITION OF THE TEST DATASETS CONSISTING OF ANNOTATED REVIEWS FOR SENTIMENT ANALYSIS AND THE LIBRARY OF CORPORA $\mathcal{T}$ USED TO TRAIN THE EMBEDDINGS.

Datasets	**#unique tokens**
Movies	300k
Airlines	17k
Full Yelp	1840k
Yelp:Restaurants	1207k
Yelp:PSG	75k
Embeddings	
Wikipedia	2546k
Wiki:Actors	314k
Wiki:Drugs	246k
Wiki:Schools	348k
Wiki:Rail transport	291k
Wiki:Cuisine	256k
Enron	389k
Legal	83k
Twitter	1920k
Subset of Yelp	38k

TABLE II
TOP-3 WORDS FOR EACH ENCODING $\mathcal{T}_i$, RANKED IN DECREASING ORDER WITH RESPECT TO α_i.

Text Corpus	**Top-3 words**
Actors	protofeminist, swordfights, storywriter
Drugs	polydrug, angiography, cyclos
Schools	auditoria, schoolmaster, Rabelaisian
Rail	southeasternmost, podcar, transponder
Cuisine	folktale, ripeness, anisette
Enron	Pinales, rebuttable, customers
Legal	hamermill, indictable, mitigator
Twitter	queening, punches, theworst
Syelp	ambience, chipotle, houseware

available annotated datasets from different domains: movie reviews [18], annotated tweets about US airlines[3], Yelp reviews[4] including two sets of reviews sampled by selecting reviews within specific categories from the full corpus (*Restaurants* and *Public Services & Government (PSG)*). Table I gives an overview of these datasets and their cardinality in terms of unique tokens.

c) Domain-Specific Text Corpora: Table I also lists the text corpora $\mathcal{T}$ which we used to train the domain-specific word2vec embeddings $\mathcal{E}$. *Wikipedia* refers to the corpus of all Wikipedia articles. Corpora labeled *Wiki:* have been obtained by sampling the complete collection of Wikipedia articles, taking into account articles belonging to the given category or its sub-categories only by parsing the Wikipedia category tree. While covering distinct domains, the nature of Wikipedia articles is however such that their language is rather uniform, in order to be understood by a large audience. The Enron dataset [19] contains e-mails between employees of the Enron corporation. Given the nature of corporate e-mails, these emails contain mainly colloquial business-related as well as formal communications. The corpus of legal texts contains court cases from the Federal Court of Australia [20] which are written in very specific legal language and terminology. We also created an embedding based on a small subset – 2.5% of the data – of the full Yelp reviews corpus (abbreviated as *syelp*). In terms of language and terminology, we are expecting this corpus to be similar to other review datasets. The *twitter* dataset was gathered from the Twitter API by sampling from the open stream over a period of 3 months in early 2016. The number of unique tokens for these corpora reflects the size of the embeddings' vocabularies, i.e., it only includes words that occurred at least 5 times.

It should be pointed out that the evaluation of *Embedding Fusion* is nontrivial, as it requires a large and diverse set of test corpora as well as domain-specific corpora that have to be trained individually and then evaluated against the test corpora.

A. Ranking Embeddings

First, we evaluate the relevance of the similarity function s (3) and its induced ranking function introduced in Section II-A. Table II shows the top-3 words according to the α weights (4) for each embedding. This highlights that our weight model α is successful in identifying uncommon words that may encode part of the corpus characteristics. However, it also shows that a large number of words may be required to fully capture the aforementioned characteristics.

Table III lists for each dataset the top-2 ranked embeddings, as well as their respective similarity values (3) computed with the top 500 words and $\sigma = 0.01$. We compare our approach to the widely used tf-idf weighted cosine similarity method,

[3]https://www.kaggle.com/crowdflower/twitter-airline-sentiment
[4]https://www.yelp.com/dataset

TABLE III

TOP-2 EMBEDDINGS AUTOMATICALLY SELECTED BY THE RANKING ALGORITHM AND BY TF-IDF FOR EACH TEST SET, WITH THEIR SIMILARITY VALUES, AS WELL AS THE TOP-2 EXPERT-PICKED EMBEDDINGS FOR THIS TASK.

Test set	Automatic selection	Similarity s	Manual selection	tf-idf-cos selection
Movies	actors	1.4×10^{-4}	actors	syelp
	syelp	7.5×10^{-5}	syelp	twitter
Airlines	twitter	5.1×10^{-6}	twitter	syelp
	rail	1.7×10^{-6}	rail	twitter
Yelp	syelp	1.1×10^{-3}	syelp	syelp
	cuisine	1.7×10^{-5}	cuisine	cuisine
Restaurants	syelp	1.0×10^{-3}	syelp	cuisine
	cuisine	2.3×10^{-4}	cuisine	twitter
PSG	syelp	4.7×10^{-4}	syelp	syelp
	rail	2.3×10^{-5}	legal	cuisine

TABLE IV

INDIVIDUAL DOMAIN-SPECIFIC AND GENERAL-PURPOSE EMBEDDINGS' PERFORMANCE FOR EACH TEST DATASET.

Embedding	Movies	Airlines	Yelp	Restaurants	PSG
Actors	0.79	0.86	0.84	0.83	0.77
Drugs	0.77	0.85	0.79	0.79	0.75
Schools	0.81	0.87	0.82	0.82	0.78
Rail	0.78	0.88	0.81	0.81	0.77
Cuisine	0.76	0.85	0.80	0.82	0.77
Enron	0.80	0.88	0.82	0.84	0.80
Legal	0.79	0.86	0.80	0.79	0.77
Twitter	0.82	**0.91**	0.85	0.85	0.79
Syelp	0.84	0.90	**0.86**	**0.86**	**0.81**
Wikipedia	**0.85**	0.90	0.84	0.84	**0.81**

tf-idf-cos for short (see for example [21]) and against the top-2 embeddings selected manually by an expert, i.e., those that a human expert would expect to be relevant to the domain of the task. All the selected embeddings match what would be expected from the text corpus domains, except for the *PSG* dataset, for which the *rail* embedding was chosen instead of an intuitively more relevant legal-oriented embedding.

It should be noted that *syelp* is selected in almost every case, which is coherent with the fact that the datasets contain reviews and opinions – the main subject of the *syelp* corpus. Also, the choice of the *twitter* embedding for the *airline* dataset is coherent with the fact that the *airline* corpus is constituted of tweets.

While the similarity scores in Table III may appear low, they are larger than similarities between unrelated text corpora by several orders of magnitude: For instance, the similarity between *airline* and *actors* is 4.0×10^{-51}, which highlights the fact that our similarity metric is successful in identifying related text corpora. On the other hand, *tf-idf-cos* always selects a relevant embedding as the first choice, but makes two mistakes for the second choices, potentially related to the highly specialized nature of our text corpus.

B. Domain-Specific Embeddings

Here we evaluate the individual performance of the topic-specific embeddings. We also compare them to general-purpose word2vec encodings trained on the full corpus of English Wikipedia articles. For each of these embeddings, we evaluate the previously defined CNN network and average the results over five runs (the dataset was randomly split into train and test sets (70%-30%) for each run). The results are presented in Table IV. As expected, the general *Wikipedia* embedding is always best or a close second on every task (see also in the plot for the PSG dataset in Fig. 2). On the other hand, the domain-specific embeddings are worse for most of the tasks in prediction accuracy. At the same time, we can observe that the *syelp* embedding performs well on all three Yelp-derived test sets, which is not unexpected given that the embedding was trained on a corpus partially overlapping the test set and is particularly relevant to all opinion-related tasks. These results highlight the need of both the ranking function, to select relevant domain specific embeddings, as well as the combination of these embeddings.

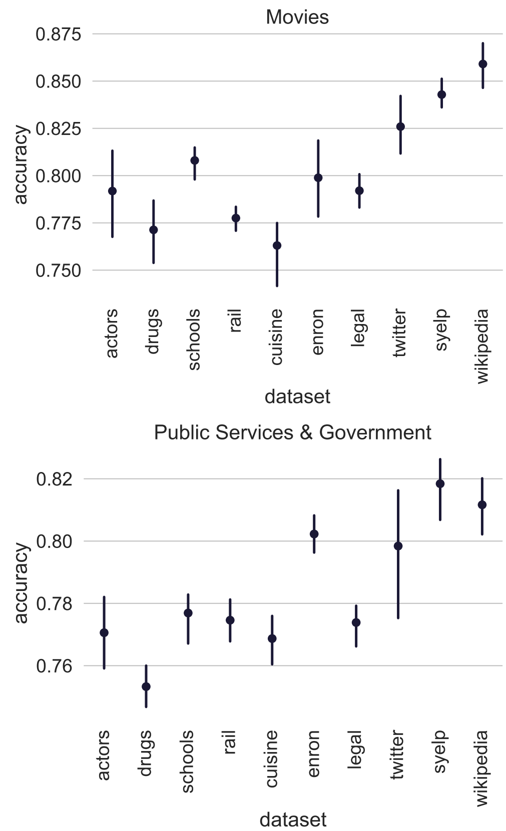

Fig. 2. Accuracy and standard deviation of all individual domain-specific embeddings compared to the general-purpose *Wikipedia* embedding for two test datasets, *movies* and *PSG*.

C. Embedding Fusion

We now evaluate the performance of the different dimensionality reduction methods described in Section II-B for the different datasets, using the two top-ranked domain-specific embeddings (cf. Table III) in all cases. For each of the aforementioned embeddings, we perform five runs of the CNN network (with a random split (70%-30%) for train and test

TABLE V

COMPARISON OF DIMENSIONALITY REDUCTION METHODS FOR THE TOP-2 AUTOMATICALLY SELECTED TEXT CORPORA FOR ALL GIVEN TASKS. THE FIRST ROW REPORTS AVERAGE ACCURACY, THE SECOND ROW THE STANDARD DEVIATION OVER 5 RUNS.

Test set	**Corpora**	$\mathcal{E}_{concat}$	$\mathcal{E}_{PCA}$	$\mathcal{E}_{auto}$	$\mathcal{E}_{G\ auto}$	$\mathcal{E}_{V\ auto}$	*Wikipedia*	$\mathcal{E}_{avg}$
Movies	actors, syelp	0.84	**0.87**	0.86	0.82	0.83	0.85	0.84
		0.013	0.0052	0.017	0.015	0.014	0.013	0.011
Airlines	twitter, rail	**0.91**	**0.91**	0.90	0.86	0.90	0.90	0.90
		0.013	0.0071	0.0074	0.016	0.012	0.0081	0.020
Yelp	syelp, cuisine	0.86	0.86	**0.87**	0.86	0.86	0.84	0.86
		0.0037	0.0012	0.0023	0.0044	0.0021	0.013	0.0040
Restaurants	syelp, cuisine	0.85	**0.86**	**0.86**	0.85	0.85	0.84	**0.86**
		0.012	0.0014	0.0026	0.0025	0.0046	0.0096	0.0028
PSG	syelp, rail	0.81	**0.83**	0.82	0.80	0.81	0.81	0.82
		0.012	0.0098	0.0079	0.0082	0.0058	0.010	0.0033

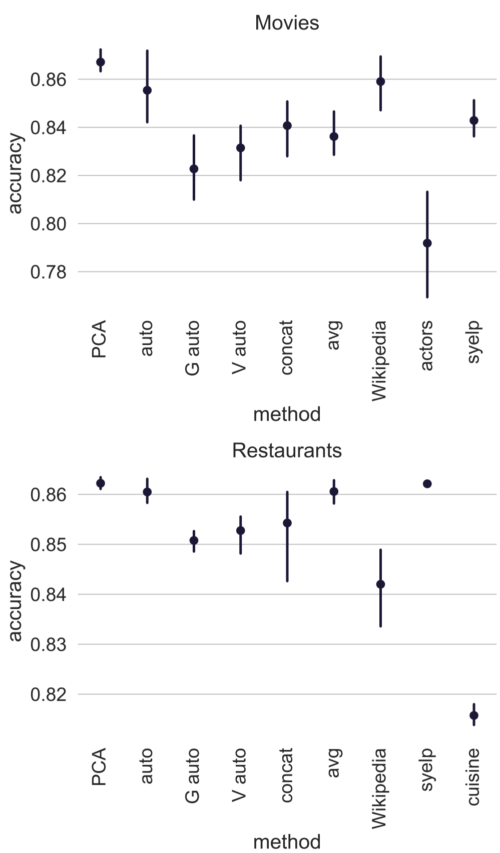

Fig. 3. Comparison of the accuracy and standard deviation of individual embeddings and fused embeddings obtained by the different dimensionality reduction methods on the movie review and restaurant review datasets.

sets). The results are reported in Table V and compared against $\mathcal{E}_{avg}$ [22] as well as the general-purpose *Wikipedia* word2vec model used individually. The results show that among the different embedding reduction methods, $\mathcal{E}_{PCA}$ outperforms the other methods – although $\mathcal{E}_{auto}$ is a close second, slightly outperforming $\mathcal{E}_{PCA}$ on the Yelp review test set. $\mathcal{E}_{PCA}$ also performs slightly better than *Wikipedia*, even though $\mathcal{E}_{PCA}$ was trained on text corpora that are smaller by several orders of magnitude. Overall, $\mathcal{E}_{PCA}$ never deteriorates the performance of the individual embeddings and outperforms both individual embeddings by a significant margin in two cases (*Movies* and *PSG*).

IV. DISCUSSION

a) Domain-Specific Embeddings: As can be seen from Section III-B, domain-specific encodings tend to be less effective than general-purpose embeddings when used individually. This is in line with the fact that encodings based on algorithms such as word2vec are typically trained using very large text corpora and therefore contain extensive vocabulary as well as information-rich encodings for many words due to their repeated occurrence in text. As can be seen in both Table IV and Fig. 2, the individual embedding built from the subsampled set of Yelp reviews performs better than the general-purpose *Wikipedia* embedding for the three Yelp-derived test sets. This is easy to explain by the fact that there is a very high similarity, potentially even an overlap, between training and test data. We nevertheless chose to include the *syelp* embedding in the tests as it performed well also on the non-Yelp test sets (cf. Table IV). While we consistently used the well-established word2vec algorithm to train our set of domain-specific embeddings, we believe our method translates equally to embeddings obtained through more recently proposed methods [5], [23].

b) Combining Domain-Specific Embeddings: The combination of relevant domain-specific embeddings was shown to consistently outperform the general-purpose embedding (see Section III-C) – despite using significantly smaller text corpora. This might be explained by the fact that a very large text corpus may induce a loss of information for some words that have different meanings in specific contexts, a loss that is also present with $\mathcal{E}_{avg}$. Indeed, embeddings might "average" those meanings during their training – a problem that is not encountered by domain-specific encodings due to their narrower usage of specific terms. These semantics are preserved in an embedding that is combined at a later stage as opposed to trained on a concatenation of the original corpora, where this averaging of meanings still occurs.

c) Ranking Embeddings: Table III shows that our ranking function selects the same embeddings as in the manual selection by human experts in all cases except one (PSG). It is interesting to note that the selected encodings include *actors* and *cuisine* – corpora related to the topic of the task – but also *twitter* – a general corpus related to the medium and language of the task – and *syelp* – a corpus related to the content of the task, i.e., opinions. This highlights the generality and the multi-purpose usefulness of our similarity score and offers a promising first step in devising a more refined algorithm that combines representative encodings derived from three separate libraries of embeddings: topic, medium, and content.

d) Dimension Reduction: According to Table V and Fig. 3, linear dimensionality reduction methods (i.e., PCA-based approaches) perform better than nonlinear methods (in this case, autoencoders and in particular their extensions) reliably, i.e., with a comparably low standard deviation. This result might be explained by the fact that Principal Component Analysis extracts the directions that account for the largest amount of variation. Since differences between words (i.e., variations) have been shown to be linked to relations [24], PCA may extract directions that are linked to important relations for this particular topic – since the different corpora have been created to be domain-specific.

V. Related Work

a) Multi-Corpus Word Embeddings: One common NLP task using two or more embeddings is machine translation, where there is one embedding per language. Methods are often put in place to align the vector spaces in order to find matching words without knowing the full ground-truth [25], [26]. When not performing alignment, machine translation requires parallel or aligned text corpora [27], [28]. Unlike these methods, our approach does not require the vector spaces to be aligned; however, since we are dealing with monolingual cases, our corpora are implicitly parallel. Another case for joining embeddings is when dealing with joint embeddings of different types of data, such as knowledge graphs with text [29], or images with text [30]. While this is in essence an alignment between different structures, our method operates solely on textual contents and requires corpora with a textual overlap. State-of-the-art approaches to multi-domain embeddings [31] require a joint training of the embeddings into a common space. Our method does not require retraining when choosing to combine embeddings of certain domains.

b) Combining Word Embeddings: Previous works have proposed different approaches to combine word embeddings, the most fundamental being the concatenation of embeddings trained with different algorithms [32]. However, this work does not consider domain-specific embeddings. Similarly, the method of averaging embeddings [22] creates a "meta-embedding" from the arithmetic mean of the vectors that is comparable in accuracy to concatenation of vectors while offering the performance benefits of lower-dimensional embeddings. We have shown to outperform this method in Table V. Yin and Schütze [33] use a multi-channel approach to word embedding, borrowing from image processing techniques and using a different encoding for each channel, while Zhang et al. [34] compute the first layers of the CNN with different word embeddings in parallel and concatenate them at the very last layer. However, and to the best of our knowledge, we are the first to propose the idea of dynamically selecting and combining several domain-specific word embeddings.

c) Sentiment Analysis: Sentiment analysis, and in particular predicting whether a given text conveys a positive or negative message, has been the subject of many publications in the recent past [13]. Deep learning networks have been a tool of choice for this task, particularly when combined with word embeddings [35]. While both recurrent neural networks and convolutional neural networks have been successfully used for this task, both families have been shown to achieve similar performance [15], [36]. Tang et al. [37] introduced the use of specific sentiment embeddings, i.e. embeddings of words with their sentiment, for the specific task of sentiment analysis. This is not suitable for our approach, as we intend to generalize to various AI tasks.

d) Dimensionality Reduction: Dimensionality reduction is a key component of many statistical learning approaches [38]. The combination of concatenation followed by dimensionality reduction is not new, as it is at the heart of many tensor-based learning methods [39]. In these cases, the dimension reduction is achieved through the choice of an appropriate regularization, such as constraints on the CP rank of the tensor [40] – an approach that can be related to the construction of the PCA embedding in our case. Similarly, the use of autoencoders as the first layers of deep neural networks is a common approach when the data are too complex (such as living in a very high-dimensional space) [9]. To the authors' knowledge, none of the methods proposed in this paper have been previously used to combine embeddings.

VI. Conclusions

In this paper, we introduced the idea of combining individual embeddings to capture domain-specific semantics. In that sense, we presented *Embedding Fusion*, a two-step process consisting in ranking and then combining domain-specific embeddings. Our ranking method captures the similarity between the corpus used by the downstream application and the various domain-specific embeddings that are available. *Embedding Fusion* was shown to select embeddings that are highly relevant and of which the combined performance is higher than a general-purpose embedding. We showed that the best performing combination method was a PCA approach, $\mathcal{E}_{PCA}$, that fuses different embeddings into a single efficient and effective embedding, which outperforms each of the embeddings taken individually as well as its nonlinear counterparts. Compared to the general-purpose *Wikipedia* embedding, $\mathcal{E}_{PCA}$ yields a consistent and significant performance improvement (2% improvement on average on already highly-accurate scores) despite being trained on data that is several orders of magnitude smaller.

In future work, we plan to improve our ranking method by incorporating additional information, e.g., by linking both the application and embedding corpora to a knowledge graph in order to capture their semantic overlap more precisely, and to develop a new approach to effectively combine three or more embeddings by using their characteristics. For this, we intend to investigate the possibility of combining embeddings that reflect specific properties of a target corpus, nameley, topic, medium, and content from different libraries of embeddings.

ACKNOWLEDGMENT

This work was funded by the *Hasler Foundation* in the context of the *City-Stories* project.

REFERENCES

[1] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013, pp. 3111–3119.

[2] W. L. Hamilton, K. Clark, J. Leskovec, and D. Jurafsky, "Inducing domain-specific sentiment lexicons from unlabeled corpora," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016, p. 595.

[3] H. Xu, B. Liu, L. Shu, and P. Yu, "Lifelong domain word embedding via meta-learning," in *International Joint Conference on Artificial Intelligence*, 2018.

[4] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[6] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of machine learning research*, vol. 2, pp. 499–526, 2002.

[7] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[8] I. Jolliffe, "Principal component analysis," in *International encyclopedia of statistical science*. Springer, 2011, pp. 1094–1096.

[9] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[10] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 490–497.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[12] J. Li, T. Luong, and D. Jurafsky, "A hierarchical neural autoencoder for paragraphs and documents," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1106–1115.

[13] B. Liu and L. Zhang, "A survey of opinion mining and sentiment analysis," in *Mining text data*. Springer, 2012, pp. 415–463.

[14] C. Guggilla, T. Miller, and I. Gurevych, "Cnn-and lstm-based claim classification in online user comments," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2740–2751.

[15] Q. T. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat, and A. Rehman, "Sentiment analysis using deep learning techniques: a review," *Int J Adv Comput Sci Appl*, vol. 8, no. 6, p. 424, 2017.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[18] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 142–150.

[19] B. Klimt and Y. Yang, "Introducing the Enron corpus." in *CEAS*, 2004.

[20] F. Galgani and A. Hoffmann, "Lexa: Towards automatic legal citation classification," in *AI 2010: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Li, Ed., vol. 6464. Springer Berlin Heidelberg, 2010, pp. 445 –454.

[21] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *Journal of artificial intelligence research*, vol. 37, pp. 141–188, 2010.

[22] J. Coates and D. Bollegala, "Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2018.

[23] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "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)*, 2018, pp. 2227–2237.

[24] O. Levy and Y. Goldberg, "Linguistic regularities in sparse and explicit word representations," in *Proceedings of the eighteenth conference on computational natural language learning*, 2014, pp. 171–180.

[25] G. Lample, M. Ott, A. Conneau, L. Denoyer *et al.*, "Phrase-based & neural unsupervised machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 5039–5049.

[26] N. Yang, S. Liu, M. Li, M. Zhou, and N. Yu, "Word alignment modeling with context dependent deep neural network," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013, pp. 166–175.

[27] S. Gouws, Y. Bengio, and G. Corrado, "Bilbowa: Fast bilingual distributed representations without word alignments," in *International Conference on Machine Learning*, 2015, pp. 748–756.

[28] I. Vulic and M.-F. Moens, "Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*, vol. 2. ACL; East Stroudsburg, PA, 2015, pp. 719–725.

[29] H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 267–272.

[30] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean, "Zero-shot learning by convex combination of semantic embeddings," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[31] W. Yang, W. Lu, and V. Zheng, "A simple regularization-based algorithm for learning cross-domain word embeddings," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2898–2904.

[32] S. Ghannay, B. Favre, Y. Esteve, and N. Camelin, "Word embedding evaluation and combination," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2016, pp. 300–305.

[33] W. Yin and H. Schütze, "Multichannel variable-size convolution for sentence classification," in *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, 2015, pp. 204–214.

[34] Y. Zhang, S. Roller, and B. C. Wallace, "MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1522–1527.

[35] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[36] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *CoRR*, vol. abs/1702.01923, 2017.

[37] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, pp. 496–509, 2016.

[38] C. O. S. Sorzano, J. Vargas, and A. D. Pascual-Montano, "A survey of dimensionality reduction techniques," *CoRR*, vol. abs/1403.2877, 2014.

[39] G. Rabusseau, B. Balle, and J. Pineau, "Multitask spectral learning of weighted automata," in *Advances in Neural Information Processing Systems*, 2017, pp. 2588–2597.

[40] D. Goldfarb and Z. Qin, "Robust low-rank tensor recovery: Models and algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 1, pp. 225–253, 2014.