In Defense of Word Embedding for Generic Text Representation

Guy Lev, Benjamin Klein, and Lior Wolf

The Blavatnik School of Computer Science
Tel Aviv University, Tel Aviv, Israel

Abstract. Statistical methods have shown a remarkable ability to capture semantics. The word2vec method is a frequently cited method for capturing meaningful semantic relations between words from a large text corpus. It has the advantage of not requiring any tagging while training. The prevailing view is, however, that it lacks the ability to capture semantics of word sequences and is virtually useless for most purposes, unless combined with heavy machinery. This paper challenges that view, by showing that by augmenting the word2vec representation with one of a few pooling techniques, results are obtained surpassing or comparable with the best literature algorithms. This improved performance is justified by theory and verified by extensive experiments on well studied NLP benchmarks.[1]

1 Introduction

Document retrieval and text analytics, in general, benefit from a fixed-size representation of variable sized text. The most basic method in the field, and still highly influential, is the bag-of-words (BOW) method. It has obvious shortcomings, such as uniform distances between the contribution of every two words to the vector representation and invariance to word order. However, these shortcomings can be partially ameliorated by incorporating techniques such as tf-idf and by considering n-grams instead of single words. However, the usage of one dimension per dictionary word leads to a representation that is sparse with respect to the information content and does not capture even the simplest synonyms.

Recently, semantic embeddings of words in vector spaces have gained a renewed interest, especially the word2vec method [19] and related methods. It has been demonstrated that not only are words with similar meanings embedded nearby, but natural word arithmetic can also be convincingly applied. For example, the calculated difference in the embedding vector space between "London" and "'England" is similar to the one obtained between "Paris" and "France". Word2vec representations are learned in a very weakly supervised manner from large corpora, and are not explicitly constrained to abide by such regularities.

Despite the apparent ability to capture semantic similarities, and the surprising emergence of semantic regularities that support additivity, word2vec embeddings have been criticized as a tool for higher level NLP. First, the Neural Network employed to learn the word2vec embeddings is a simple "shallow" (not deep) network, capable, by common conception, of capturing only low-level information. Taking an analogy from the field of image recognition, where very deep

[1] This work is inspired by [10].

networks are being deployed, word2vec is considered to be a low-level "edge detection" operator, incapable of capturing complex compositional semantics. Second, word2vec has been criticized for being almost equivalent to the much earlier methods of frequency matrix factorization [17]. Third, it has been argued that in order to capture more than single words, mechanisms should be added in order to account for order and hierarchical compositions [32]. The alleged inability of vector embeddings to solve mid- and high-level NLP problems was also demonstrated in various NLP papers, where an average of vector embeddings served as a baseline method.

It is the purpose of this paper to challenge the commonly held view that the word2vec representation is inadequate and markedly inferior to more sophisticated algorithms. The poor performance of the word2vec representation can probably be traced to aggregation techniques that do not take sufficient account of numerical and statistical considerations. It is shown in this paper that proper pooling techniques of the vectors of the text words leads to state of the art or at least very competitive results.

Given a text to represent, we consider it as a multi-set, i.e., as a generalized set in which each element can appear multiple times. We advocate the use of principal component analysis (PCA) or independent component analysis (ICA) as an unsupervised preprocessing step that transforms the semantic vector space into independent semantic channels. For pooling, as shown, the mean vector performs well. In some situations, the more powerful Fisher Vector (FV) [22] representation provides improved results.

Fisher Vectors provide state-of-the-art results on many different applications in the domain of computer vision [29, 21, 7, 23]. In all of these contributions, the FV of a set of local descriptors is obtained as a concatenation of gradients of the log-likelihood of the descriptors in the set with respect to the parameters of a Gaussian Mixture Model (GMM) that was fitted on a training set in an unsupervised manner. In our experiments, we do not observe a clear benefit to GMM over a simple Gaussian Model. Due to the clear disadvantage of the extra parameter (the number of mixture components), we focus on modeling by a unimodal Gaussian. Furthermore, to account for the non-Gaussian nature of the data incurred by the ICA transformation, we propose to use Generalized Gaussian Models. The corresponding Fisher Vectors are derived and formulas are also given to the approximation of the Fisher Information Matrix in order to allow for normalization of the dynamic range of the FV variant presented.

2 Previous work

Representing text as vectors Word2vec [18, 19] is a recently developed technique for building a neural network that maps words to real-number vectors, with the desideratum that words with similar meanings will map to similar vectors. This technique belongs to the class of methods called "neural language models". It uses a scheme that is much simpler than previous work in this domain, where neural networks with many hidden units and several non-linear layers were normally constructed (e.g., [5]), word2vec [18] constructs a simple log-linear classification network [20]. Two such networks are proposed: the Skip-gram and the Continuous Bag-of-words (CBOW) architectures. In our experiments, we employ the Skip-gram architecture, which is considered preferable.

Attention has recently shifted into representing sentences and paragraphs and not just words. The classical method in this domain is Bag of Words [30]. Socher et al. [31] have analyzed sentences using a recursive parse tree. The combination of two subtrees connected at the root, by means of generating a new semantic vector representation based on the vector representations of the two trees, is performed by concatenating their semantic vector representations and multiplying by a matrix of learned parameters. In a recent contribution by Le et al. [15], the neural network learns to predict the following word in a paragraph based on a representation that concatenates the vector representation of the previous text and the vector representations of a few words from the paragraph. This method, called the paragraph vector, achieves state-of-the-art results on the Stanford Sentiment Treebank dataset surpassing a model that averages neural word vectors and ignores word order.

In [40], Yu et al. are using distributed representations that are based on deep learning for the task of identifying sentences that contain the answer to a given question. Given word embeddings, their first model generates the vector representation of a sentence by taking the mean of the word vectors that compose the sentence. Since their first model does not account for word ordering and other structural information, they developed a more complex model that works on the word embedding of the bigrams. Their model matches state of the art performance on the TREC answer selection dataset.

Pooling methods were one of the primary steps in many computer vision pipelines in the era before the advent of Deep Learning. Many different pooling methods were suggested in the last decade, each contributing to the improvement in accuracy on the standard object recognition benchmarks. One of the most known and basic pooling techniques was borrowed from the NLP community when Sivic et al. [30] used clustering over local features of image patches in order to create a bag of words representation for computer vision applications. Richer representations like VLAD [13] and FV [22] were later introduced and were the main contributors to the increasing in accuracy in object recognition benchmarks.

Specifically, the FV representation is today the leading pooling technique in traditional computer vision pipelines and provided state-of-the-art results on many different applications [29, 21, 7, 23]. Although already introduced in 2007, the FV pooling method was able to surpass the bag of words representation only after introducing improvements such as normalization techniques that have dramatically enhanced its performance. Some of the most widely used improvements were introduced by Perronnin et al. [23]. The first improvement is to apply an element-wise power normalization function, $f(z) = \mathsf{sign}(z)|z|^{\alpha}$ where $0 \leq \alpha \leq 1$ is a parameter of the normalization. The second improvement is to apply a L2 normalization on the FV after applying the power normalization function. By applying these two operations [23] achieved state-of-the-art accuracy on an image recognition benchmark called CalTech 256 and showed superiority over the traditional Bag of Visual Words model.

3 Pooling

In our approach, a single sentence is represented as a multi-set of word2vec vectors. The notation of a multi-set is used to clarify that the order of the words in a sentence does not affect the final representation and that a vector can appear more than once (if the matching word appears more than once in the sentence). In order to apply machine learning models to the sentences, it is useful to transform this multi-set into a single high dimensional vector with a constant length. This can be achieved by applying pooling.

Since word2vec is already an extremely powerful representation, we find that conventional pooling techniques or their extensions are sufficiently powerful to obtain competitive performance. The pooling methods that are used in this paper are: (1) Mean vector pooling; (2) FV of a single multivariate Gaussian; (3) FV of a single multivariate generalized Gaussian. These are described in the next sections.

3.1 Mean vector

This pooling technique takes a multiset of vectors, $X = \{x_1, x_2, \ldots, x_N\} \in R^D$, and computes its mean: $v = \frac{1}{N}\sum_{i=1}^{N} x_i$. Therefore, the vector v that results from the pooling is in R^D.

The disadvantage of this method is the blurring of the text's meaning. By adding multiple vectors together, the location obtained – in the semantic embedding space – is somewhere in the convex hull of the words that belong to the multi-set. A better approach might be to allow additivity without interference.

3.2 Fisher Vector of a multivariate Gaussian

Given a multiset of vectors, $X = \{x_1, x_2, \ldots, x_N\} \in R^D$, the standard FV [22] is defined as the gradients of the log-likelihood of X with respect to the parameters of a pre-trained Diagonal Covariance Gaussian Mixture Model. It is common practice to limit the FV representation to the gradients of the means, μ and to the gradients of the standard deviations, σ (the gradients of the mixture weights are ignored).

Since we did not notice a global improvement in accuracy when increasing the number of Gaussian in the mixture, we focus on a single multivariate Gaussian. As a consequence, there are no latent variables in the model and it is, therefore, possible to estimate the parameters $\lambda = \{\mu, \sigma\}$ of this single diagonal covariance Gaussian by using maximum likelihood derivations, instead of using the EM algorithm which is usually employed when estimating the parameters of the Gaussian Mixture Model. Under this simplified version of the FV, the gradients from which the FV is comprised are:

$$\frac{\partial \mathcal{L}\left(X|\lambda\right)}{\partial \mu_d} = \sum_{i=1}^{N} \frac{x_{i,d} - \mu_d}{\sigma_d^2} \;\; ; \quad \frac{\partial \mathcal{L}\left(X|\lambda\right)}{\partial \sigma_d} = \sum_{i=1}^{N} \left(\frac{\left(x_{i,d} - \mu_d\right)^2}{\sigma_d^3} - \frac{1}{\sigma_{j,d}} \right) \tag{1}$$

and, therefore, the resulting representation is in R^{2D}. Applying PCA and ICA as a preprocessing step is investigated in this work with the purpose of sustaining the diagonal covariance assumption.

As in [22], the diagonal of the Fisher Information Matrix, F, is approximated in order to normalize the dynamic range of the different dimensions of the gradient vectors. For a single Gaussian model, the terms of the approximated diagonal Fisher Information Matrix become: $F_{\mu_d} = \frac{N}{\sigma_{k,d}^2}$; $F_{\sigma_d} = \frac{2N}{\sigma_{k,d}^2}$.

The FV is the concatenation of two normalized partial derivative vectors: $F_{\mu_d}^{-1/2} \frac{\partial \mathcal{L}(X|\lambda)}{\partial \mu_d}$ and $F_{\sigma_d}^{-1/2} \frac{\partial \mathcal{L}(X|\lambda)}{\partial \sigma_d}$.

It is worth noting the linear structure of the FV pooling, which is apparent from the equations above. Since the likelihood of the multi-set is the multiplication of the likelihoods of the individual elements, the log-likelihood is linear. Therefore, the Fisher Vectors of the individual words can be computed once for each word and then reused. For all of our experiments, the multivariate Guassian (or the generalized Gaussian presented next) is estimated only once, from all word2vec vectors. These vectors are obtained, precomputed on a subset of the Google News dataset, from `https://code.google.com/p/word2vec/`. Therefore, the encoding is independent of the dataset used in each experiment, is completely generic, and is very efficient to compute as a simple summation of precomputed Fisher Vectors (same runtime complexity as mean pooling).

Following the summation of the Fisher Vectors of the individual words, the Power Normalization and the L2 Normalization that were introduced in [24] (see Section 2) are employed, using a constant $a = 1/2$.

3.3 Fisher Vector of a generalized multivariate Gaussian

A generalization of the FV that is presented here for the first time, in which the FV is redefined according to a single multivariate generalized Gaussian distribution. The need for this derivation is based on the observation (see below) that word2vec vectors are not distributed in accordance with the multivariate Gaussian distribution.

The generalized Gaussian distribution is, in fact, a parametric family of symmetric distributions and is defined by three parameters: m which is the location parameter and is the mean of the distribution, s the scale parameter and p the shape parameter. The probability density function of the Generalized Gaussian Distribution (GGD) in the univariate case is:

$$ggd(x; m, s, p) = \frac{1}{2sp^{1/p}\Gamma(1 + 1/p)} exp\left(-\frac{|x - m|^p}{ps^p}\right) \quad (2)$$

The estimation of the parameters of a univariate Generalized Gaussian Distribution is done according to [2].

Under the common assumption in the FV that the covariance matrix is diagonal, the multivariate generalized Gaussian distribution is defined:

$$ggd(\boldsymbol{x}; \boldsymbol{m}, \boldsymbol{s}, \boldsymbol{p}) = \prod_{d=1}^{D} \frac{1}{2s_d p_d^{1/p_d}\Gamma(1 + 1/p_d)} exp\left(-\frac{|x_d - m_d|^{p_d}}{p_d s_d^{p_d}}\right) \quad (3)$$

Since the dimensions of the multivariate GGD are independent, the parameters of the GGD can be estimated dimension-wise.

The FV can now be redefined as the gradients of the log-likelihood of $X = \{x_1, x_2, \ldots, x_N\} \in R^D$ with respect to the parameters of a pre-trained Diagonal Covariance Multivariate Generalized Gaussian Distribution. In practice, the FV is defined in this work only according to the gradients of m and s, since the gradients according to p do not seem to improve the results.

The log likelihood is defined as:

$$\mathcal{L}(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{p}|X) = \sum_{d=1}^{D} \left[-N \log \left(2 s_d p_d^{1/p_d} \mathit{\Gamma}(1 + 1/p_d) \right) - \frac{\sum_{i=1}^{N} |x_{id} - m_d|^{p_d}}{p_d s_d^{p_d}} \right] \tag{4}$$

The resulting FV in R^{2D} is given by:

$$\frac{\partial \mathcal{L}(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{p}|X)}{\partial m_d} = s_d^{-p_d} \sum_{i=1}^{N} |x_{id} - m_d|^{p_d - 1} \mathsf{sign}(x_{id} - m_d) \tag{5}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{p}|X)}{\partial s_d} = -N/s_d + s_d^{-p_d - 1} \sum_{i=1}^{N} |x_{id} - m_d|^{p_d} \tag{6}$$

The diagonal of Fisher Information Matrix, F, for this distribution is approximated in order to normalize the dynamic range of the different dimensions of the gradient vectors. Let F_{m_d} and F_{s_d} be the terms of diagonal of F that correspond respectively to $\frac{\partial \mathcal{L}(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{p}|X)}{\partial m_d}$ and $\frac{\partial \mathcal{L}(\boldsymbol{m}, \boldsymbol{s}, \boldsymbol{p}|X)}{\partial s_d}$. Then:

$$F_{m_d} = \int_X ggd\left(X|\lambda\right) \left[\sum_{i=1}^{N} \frac{\partial \mathcal{L}\left(x_i|\lambda\right)}{\partial m_d} \right]^2 dX \tag{7}$$

Where $\lambda = \{m, s, p\}$ Then:

$$\begin{aligned} F_{m_d} = & \sum_{\substack{t=1\ldots N \\ u=1\ldots N \\ t \neq u}} \int_{x_t, x_u} \frac{\partial \mathcal{L}\left(x_t|\lambda\right)}{\partial m_d} \frac{\partial \mathcal{L}\left(x_u|\lambda\right)}{\partial m_d} ggd\left(x_t, x_u|\lambda\right) dx_t dx_u \\ & + \sum_{t=1}^{N} \int_{x_t} \left[\frac{\partial \mathcal{L}\left(x_t|\lambda\right)}{\partial m_d} \right]^2 ggd\left(x_t|\lambda\right) dx_t \end{aligned} \tag{8}$$

Since the samples are i.i.d given λ and also the dimensions are independent:

$$\begin{aligned} & \int_{x_t, x_u} \frac{\partial \mathcal{L}\left(x_t|\lambda\right)}{\partial m_d} \frac{\partial \mathcal{L}\left(x_u|\lambda\right)}{\partial m_d} ggd\left(x_t, x_u|\lambda\right) dx_t dx_u \\ = & \int_{x_{t,d}} \frac{\partial \mathcal{L}\left(x_{t,d}|\lambda\right)}{\partial m_d} ggd\left(x_{t,d}|\lambda\right) dx_{t,d} \int_{x_{u,d}} \frac{\partial \mathcal{L}\left(x_{u,d}|\lambda\right)}{\partial m_d} ggd\left(x_{u,d}|\lambda\right) dx_{u,d} \end{aligned}$$

Using the fact that $\frac{\partial \mathcal{L}(x_{t,d}|\lambda)}{\partial m_d} = \frac{\partial}{\partial m_d} log(ggd\left(x_{t,d}|\lambda\right)) = \frac{\frac{\partial}{\partial m_d} ggd(x_{t,d}|\lambda)}{ggd(x_{t,d}|\lambda)}$:

$$\int_{x_{t,d}} \frac{\partial \mathcal{L}\left(x_{t,d}|\lambda\right)}{\partial m_d} ggd\left(x_{t,d}|\lambda\right) dx_{t,d} = \int_{x_{t,d}} \frac{\partial}{\partial m_d} ggd\left(x_{t,d}|\lambda\right) dx_t = \frac{\partial}{\partial m_d} \int_{x_{t,d}} ggd\left(x_{t,d}|\lambda\right) dx_t = 0$$

Therefore, the first expression in the sum of (8) is equal to 0. Assuming that the dimensions are independent, the second expression in the sum of (8) is equal to $\sum_{t=1}^{N} \int_{x_{t_d}} \left[\frac{\partial \mathcal{L}\left(x_{t_d}|\lambda\right)}{\partial m_d}\right]^2 ggd\left(x_{t_d}|\lambda\right) dx_{t_d}$.

Note that $\int_{x_{t_d}} \left[\frac{\partial \mathcal{L}\left(x_{t_d}|\lambda\right)}{\partial m_d}\right]^2 ggd\left(x_{t_d}|\lambda\right) dx_{t_d}$ is the value of the Fisher Information Matrix of a univariate generalized Gaussian distribution for a single sample. Therefore according to [2]:

$$\int_{x_{t_d}} \left[\frac{\partial \mathcal{L}\left(x_{t_d}|\lambda\right)}{\partial m_d}\right]^2 ggd\left(x_{t_d}|\lambda\right) dx_{t_d} = \frac{(p-1)\Gamma\left(\frac{p-1}{p}\right)}{s^2 \Gamma\left(\frac{1}{p}\right) p^{(2-p)/p}} \tag{9}$$

Therefore:

$$F_{m_d} = N \cdot \frac{(p-1)\Gamma\left(\frac{p-1}{p}\right)}{s^2 \Gamma\left(\frac{1}{p}\right) p^{(2-p)/p}} \tag{10}$$

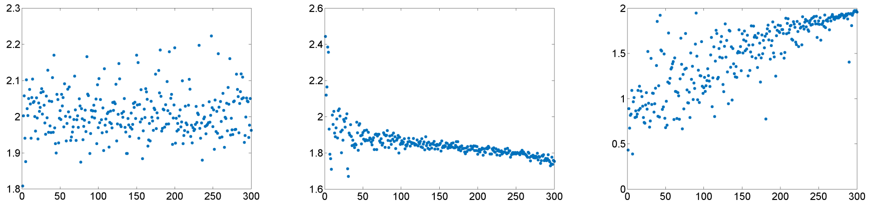
Similarly, since $\int_{x_{t_d}} \left[\frac{\partial \mathcal{L}\left(x_{t_d}|\lambda\right)}{\partial s_d}\right]^2 ggd\left(x_{t_d}|\lambda\right) dx_{t_d} = \frac{p}{s^2}$ according to [2], it can be shown that: $F_{s_d} = N \cdot \frac{p}{s^2}$.

The normalized partial derivatives of the FV are then $F_{m_d}^{-1/2} \frac{\partial \mathcal{L}(X|\lambda)}{\partial m_d}$ and $F_{s_d}^{-1/2} \frac{\partial \mathcal{L}(X|\lambda)}{\partial s_d}$.

In [27], Sanchez et al. state that applying the Principal Components Analysis (PCA) on the data before fitting the GMM is the key to make the FV perform well. In experiments on PASCAL VOC 2007, they show that accuracy does not seem to be overly sensitive to the exact number of PCA components. The explanation is that transforming the descriptors by using PCA is a better fit to the diagonal covariance matrix assumption.

Following this observation, a transformation that will cause the transformed descriptors to be a better fit to the diagonal covariance matrix assumption is sought for the generalized gaussian FV. The optimal transformation will result in transformed descriptors that are dimension independent and are non-Gaussian signals. While PCA suffers from the implicit assumption of an underlying Gaussian distribution [14], the Independent Component Analysis (ICA) [16] explicitly encourages non-Gaussian distributions.

Figure 1 depicts the estimated shape parameters p for each dimension of the word2vec representation, and for all dictionary words used. As can be seen, the shape varies between the dimensions, depending on whether we consider the raw word2vec representation, the representation post-PCA, or that after



(a) (b) (c)

Fig. 1. The shape parameter p of the generalized Gaussian distribution. This parameter was estimated for each dimension of the word2vec representation, based on all word2vec vectors, i.e., a distribution was fit to each coordinate separately. (a) the raw word2vec vectors; (b) after applying PCA, retaining the original dimensionality; (c) after applying ICA. In all three plots, x-axis is the vector coordinate index from 1 to 300, y-axis is the estimated p. Note that the range of the y-axis differs between the plots.

applying ICA. The baseline distribution is not a Gaussian one, but most shape parameters are between 1.9 and 2.1. Post-PCA, the shape parameters are mostly in a narrow band around 1.9. Post-ICA, the shape parameters follow an almost linear trend between 0.8 and 2.

Finally, The Power Normalization and L2 Normalization are applied using $a = 1/p$ on the resulting FV. While similar to the conventional FV, this constant is not justified directly, we found it experimentally to slightly outperform $a = 1/2$ for this case.

3.4 Classification

The pooled representation of a sentence can be used in combination with any classifier to make predictions based on the sentence. In addition, many of our experiments require the comparison of two sentences. Let u and v be the pooled representations of the two sentences. Our unified representation is given by the concatenation of their difference and their mean: $\begin{bmatrix} |u-v| \\ (u+v)/2 \end{bmatrix}$. This provides information on both the location of the two vectors and the difference between them, in a symmetric manner.

4 Experiments

We perform our experiments on multiple benchmarks: the TREC Answer Selection Dataset, The SemEval-2012 Semantic Sentence Similarity benchmark, and the very recent Yahoo! and AG topic classification benchmarks.

4.1 Answer selection

The answer sentence selection dataset contains factoid questions each associated with a list of answer sentences. It was created by Wang et al. [36] from the Text REtrieval Conference (TREC) QA track (8-13) dataset, with candidate answers automatically selected from each question's document pool. This selection was based on a combination of overlapping non-stop word counts and pattern matching, and was followed by manual tagging for parts of the dataset. Overall, there are 4718, 1148, and 1517 question-answer pairs in the train, validation, and test set, respectively.

The task is to rank the candidate answers based on their relation to the question. Two standard success metrics are used and in both higher is better: Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). MRR measures the rank of any correct answer and MAP examines the ranks of all the correct answers and accounts for recall. The two scores are calculated using the official `trec_eval` evaluation scripts.

We compare our results with the state of the art [36, 11, 35, 38, 28, 39, 40]. Our method employs the concatenated diff+mean vector of Section 3.4. Linear SVM is used with a parameter C tuned on the development set.

As can be seen in Table 1, the most basic pooling method of average pooling is already preferable, when applied to word2vec transformed by PCA, to the literature methods. Moreover, when adding FV pooling, the results further improve. Best results are obtained using the ICA + generalized Gaussian FV representation.

It is interesting to compare our method to the method of [40], which also relies on word embedding. While our method employs word2vec, [40] employs the Collobert and Westons neural language model [8] as provided by Turian et al. [33]. The unigram model of [40] is similar to our mean pooling method. However, it uses the classification model of [6]: given vector representations of a question $\mathbf{q}$ and an answer $\mathbf{a}$ (both in $\mathbb{R}^d$), the probability of the answer being correct is $p(y = 1|\mathbf{q}, \mathbf{a}) = \sigma(\mathbf{q}^T \ \mathbf{M} \ \mathbf{a} + b)$, where the transformation matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ and the bias term b are learned model parameters. The bigram model of [40] is a 1D Convolution Neural Network (CNN) with a single convolution layer and a filter size of 2.

The authors of [40] suggest that vector representation based approaches are "not very well equipped for dealing with cardinal numbers and proper nouns, especially considering the small dataset". Therefore, they augment these with two counting based features: word co-occurrence count and word co-occurrence count weighted by idf values. The output of the unigram or bigram model is concatenated in their experiments with these features and then a logistic classifier is applied. In our experiments, we do not observe the need to add such features.

Recently, an extended training set called TRAIN-ALL was proposed [40]. This is a significantly larger training set that was labeled automatically, using pattern matching, and contains many labeling errors. The best result obtained on this dataset [40] has a MAP of 0.711 (MRR 0.785) using the deep learning bigram + count method. Our best result is superior on this training set as well: MAP of 0.720 (MRR 0.824). Stacking [37], using a fourth linear SVM, all three ICA variants, improves results on TRAIN-ALL to MAP 0.7372 (MRR 0.8511).

4.2 Semantic Sentence Similarity

The task of Semantic Sentence Similarity (STS) has gained considerable attention. Semantic embedding models are at a disadvantage for this task, since the

Method	MAP	MRR
Wang et al. [36]	0.603	0.685
Heilman and Smith [11]	0.609	0.692
Wang and Manning [35]	0.595	0.695
Yao et al. [38]	0.631	0.748
Severyn and Moschitti [28]	0.678	0.736
baseline: word counts [39]	0.571	0.627
baseline: tf-idf Word Count [39]	0.596	0.652
Yih et al. LR [39]	0.682	0.762
Yih et al. BDT [39]	0.694	0.789
Yih et al. LCLR [39]	0.709	0.770
deep learning unigram [40]	0.539	0.628
deep learning unigram+count [40]	0.689	0.773
deep learning bigram [40]	0.548	0.644
deep learning bigram+count [40]	0.706	0.780
mean pooling	0.665	0.752
PCA + mean pooling	0.710	0.807
ICA + mean pooling	0.679	0.783
Gaussian FV	0.662	0.763
PCA+Gaussian FV	0.621	0.743
ICA + Gaussian FV	0.705	0.810
generalised Gaussian FV	0.654	0.757
PCA + generalized Gaussian FV	0.623	0.729
ICA + generalized Gaussian FV	**0.719**	**0.824**

Table 1. Experimental results on the TREC Answer Selection benchmark. A long list of literature results are presented, including the state of the art results obtained by Yih et al. [39] and the very recent results of Yu et al. [40]. PCA followed by mean pooling outperform all literature results; ICA + generalized Gaussian FV performs even better. Yu et al. [40] also present results on a larger and noisier training set called TRAIN ALL. Training on this training set (not shown in the table), we obtain a slight improvement only; However, our results are still better than Yu et al. [40]: MAP 0.720 vs. 0.711; MRR 0.824 vs. 0.785.

structure of the sentences is complex, and explicit matching between parts of the sentence greatly aids the similarity judgment. In our experiments below, we aim to show that word2vec pooling provides a reasonable pipeline, and that when added to a set of literature scores, state of the art results are obtained.

The experimental setup used in the STS task [1] was followed, and for technical reasons (availability of DKPro scores) we employ 3 out of the 5 datasets presented: msr-par, msr-vid, and smt-eur. Each sentence pair in the datasets was given a score from 0 (lowest similarity) to 5 (highest similarity) by human judges. We compare our results to the state of the art results [25, 3, 34, 12, 26].

The authors of [3] have released a toolbox called DKPro that contains code for the computation of 75 similarities [4] that is a superset of the 20 similarities used in [3]. Unable to completely identify the 20 similarities, we have rerun the entire set of 75 similarities as an additional pipeline. When taking log scale of the similarities, it seems to outperform [3] on the msr-par benchmark but not on the other two.

We compute the two representations of each pair of sentences and combine them (Section 3.4). For the regression problem of the STS benchmarks, we use the effective K-clusters Regression Forests (KRF) [9] method, with the default parameters. Interestingly, on the STS benchmarks the exact combinations of PCA or ICA and pooling method did not show any clear winners. The results of all 9 combinations (including no feature transformation) were almost indistinguishable. We, therefore, present the results of PCA followed by average pooling, which is the most basic method we recommend. We also present results obtained

Method	msr-par	msr-vid	smt-eur
ADW [25]	0.694	0.887	0.555
UKP2 [3]	0.683	0.873	0.528
TLsyn [34]	0.698	0.862	0.361
TLsim [34]	0.734	0.880	0.477
VD [12]	-	0.890	-
MTL-GP [26]	0.732	0.888	0.562
DKPro scores [4] (log transformed)	0.734	0.887	0.540
PCA+mean pooling	0.537	0.827	0.513
PCA+mean pooling ∪ DKPro scores	**0.739**	**0.895**	**0.617**

Table 2. Results on the STS benchmarks. Our results are shown for PCA followed by mean pooling only, since other pooling options gave almost identical results.

when combining the mean pooling similarity with the DKPro similarities. This is done by the ridge regression method on the 76 similarities, where the regularization parameter was obtained using cross validation on the training set.

The results are presented in Table 2. The results obtained by average pooling would have placed this system as one of the top systems of the SemEval-2012 competition [3, 34]. When combined with the DKPro similarities, state-of-the-art results are obtained.

4.3 Topic classification

A week before the submission date, Zhang and LeCun have published a Technical Report presenting topic classification results obtained using deep temporal convolutional networks [41]. The paper presents word2vec as an inferior baseline, performing even worse than the basic bag-of-words method. It is claimed that this might be a result of using the same word2vec representation for all datasets, or "it might also be the case that the hope for linear separability of word2vec is not valid at all". As we show below, this is not the case, and word2vec performs on par with the best results of [41].

Pooling of word2vec in [41] is performed by running k-means on the word vectors (k = 5000), and then using histograms of length 5000 to represent the text, based on nearest centroid association. This is followed by logistic regression. This metod is vastly different from the pooling methods we advocate for.

We performed experiments on two of the datasets used in [41]: Yahoo! and AG. While the exact splits used were not made available yet (personal communication), the protocols for building the benchmarks are available. We verified that different random sampling of train/test have only a minimal effect on the results, with a SD of about 0.005 accuracy. The Yahoo! Answers Topic Classification benchmark is based on the Yahoo! Answers Comprehensive Questions and Answers version 1.0 dataset available through the Yahoo! Webscope program. Topic classification is performed on the 10 largest main categories, where each class contains 140,000 (5,000) random training (testing) samples. Out of all the answers and other meta-information, only the best answer content and the main category information are used for the benchmark. From the AG's corpus of news article `http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`, the 4 largest categories are used, employing only the title and description fields. From each category, random 40,000 (1,100) samples are taken for training (testing).

Method	Yahoo!	AG
Large ConvNet + Thesaurus [41]	0.699	**0.916**
Bag of Words [41]	0.666	0.883
word2vec bag-of-centroids [41]	0.588	0.853
PCA+mean pooling+linear SVM	0.688	0.896
PCA+mean pooling + KNN	0.672	0.906
ICA+3 pooling methods +KNN	**0.703**	0.910

Table 3. Results on the topic classification benchmarks (accuracy). Our word2vec based methods are much better than the word2vec baseline of [41] and nearly as good as the best reported method of [41].

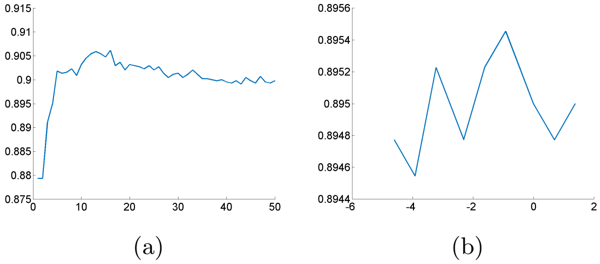


Fig. 2. Results on the AG benchmark when varying the papameters of the learning algorithm. (a) varying the parameter k of the KNN algorithm. (b) varying the parameter C of linear SVM (log scale).

Since each vector is classified independently (no pairs), we simply employ linear SVM or the k-nearest neighbor algorithms. The results are depicted in Table 3. As can be seen, our word2vec considerably outperforms the baseline given in [41] and is only slightly worse than the results of the deep networks. Needless to say, the deep networks were completely retrained for each benchmark, and are extremely resource-heavy; A single epoch on the Yahoo! benchmark took a day to train. Also, our system has only the parameters of the classifiers, and as can be seen in Figure 2, it is insensitive to the choice of these parameter. This, in comparison to the tens of hyperparameters of the deep network solutions.

In this experiment too, the pooling method did almost no difference. For example, for AG KNN classification, all 9 options where at an accuracy level above 0.899. However, by stacking the results obtained, for example, by the three ICA-based pooling methods, performance is slightly improved to 0.910 on this benchmark, and 0.703 on the Yahoo! benchmark.

5 Conclusion

With proper pooling, vector embeddings perform almost as well, if not better, than the best available methods. On the other hand, the proposed pipeline is generic and mostly unsupervised, and only requires a shallow off-the-shelf training in order to adapt to the problem at hand. The Fisher Vector pooling methods share the same runtime complexity as the baseline mean pooling method, and improve results significantly in two out of the three tasks we examined.

Word order is not properly addressed, as is apparent in the STS experiments. We plan to tackle this using a hierarchical pooling scheme that represents text by a list of pooled vectors. In addition, we plan to study pooling of other types of vector embedding such as co-occurance based ones.

Acknowledgments

This research is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

1. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics. pp. 385–393. SemEval '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2387636.2387697
2. Agro, G.: Maximum likelihood estimation for the exponential power function parameters. Communications in Statistics-Simulation and Computation 24(2), 523–536 (1995)
3. Bär, D., Biemann, C., Gurevych, I., Zesch, T.: Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. pp. 435–440. SemEval '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2387636.2387707
4. Bär, D., Zesch, T., Gurevych, I.: Dkpro similarity: An open source framework for text similarity. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 121–126. Association for Computational Linguistics (2013), http://aclweb.org/anthology/P13-4021
5. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. 3, 1137–1155 (Mar 2003), http://dl.acm.org/citation.cfm?id=944919.944966
6. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. CoRR abs/1406.3676 (2014), http://arxiv.org/abs/1406.3676
7. Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: British Machine Vision Conference (2011)
8. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning. pp. 160–167. ICML '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1390156.1390177
9. Hara, K., Chellappa, R.: Growing regression forests by classification: Applications to object pose estimation. In: Computer Vision–ECCV 2014, pp. 552–567. Springer (2014)
10. Hartley, R.: In defense of the eight-point algorithm. Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(6), 580–593 (Jun 1997)
11. Heilman, M., Smith, N.A.: Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 1011–1019. HLT '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010), http://dl.acm.org/citation.cfm?id=1857999.1858143

12. Hodosh, P.Y.A.L.M., Hockenmaier, J.: From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics (2014)
13. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: Proc. IEEE Conf. Comput. Vision Pattern Recognition. pp. 3304–3311 (jun 2010), http://lear.inrialpes.fr/pubs/2010/JDSP10
14. Ke, Y., Sukthankar, R.: Pca-sift: A more distinctive representation for local image descriptors. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. vol. 2, pp. II–506. IEEE (2004)
15. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. JMLR Proceedings, vol. 32, pp. 1188–1196. JMLR.org (2014), http://jmlr.org/proceedings/papers/v32/le14.html
16. Lee, T.W.: Independent component analysis: theory and applications [book review]. IEEE Transactions on Neural Networks 10(4), 982–982 (1999), http://dblp.uni-trier.de/db/journals/tnn/tnn10.html#Lee99a
17. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 27, pp. 2177–2185. Curran Associates, Inc. (2014), http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
19. 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. pp. 3111–3119 (2013)
20. Mnih, A., Hinton, G.: Three new graphical models for statistical language modelling. In: Proceedings of the 24th International Conference on Machine Learning. pp. 641–648. ICML '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1273496.1273577
21. Peng, X., Zou, C., Qiao, Y., Peng, Q.: Action recognition with stacked fisher vectors. In: Computer Vision–ECCV 2014, pp. 581–595. Springer (2014)
22. Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. pp. 1–8. IEEE (2007)
23. Perronnin, F., Liu, Y., Sánchez, J., Poirier, H.: Large-scale image retrieval with compressed fisher vectors. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 3384–3391. IEEE (2010)
24. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Computer Vision–ECCV 2010, pp. 143–156. Springer (2010)
25. Pilehvar, T.M., Jurgens, D., Navigli, R.: Align, disambiguate and walk: A unified approach for measuring semantic similarity. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1341–1351. Association for Computational Linguistics (2013), http://aclweb.org/anthology/P13-1132
26. Rios, M., Specia, L.: Uow: Multi-task learning gaussian process for semantic textual similarity. In: Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). pp. 779–784. Association for Computational Linguistics and

Dublin City University, Dublin, Ireland (August 2014), http://www.aclweb.org/anthology/S14-2138

27. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. International journal of computer vision 105(3), 222–245 (2013)
28. Severyn, A., Moschitti, A.: Automatic feature engineering for answer selection and extraction. In: EMNLP. pp. 458–467. ACL (2013), http://dblp.uni-trier.de/db/conf/emnlp/emnlp2013.html#SeverynM13
29. Simonyan, K., Parkhi, O.M., Vedaldi, A., Zisserman, A.: Fisher vector faces in the wild. In: Proc. BMVC. vol. 1, p. 7 (2013)
30. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their location in images. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. vol. 1, pp. 370–377. IEEE (2005)
31. Socher, R., Lin, C.C., Ng, A.Y., Manning, C.D.: Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In: Proceedings of the 26th International Conference on Machine Learning (ICML) (2011)
32. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. pp. 1631–1642. Association for Computational Linguistics, Seattle, Washington, USA (Oct 2013), http://www.aclweb.org/anthology-new/D/D13/D13-1170.bib
33. Turian, J., Ratinov, L., Bengio, Y.: Word representations: A simple and general method for semi-supervised learning. In: ACL (2010), http://cogcomp.cs.illinois.edu/papers/TurianRaBe2010.pdf
34. Šarić, F., Glavaš, G., Karan, M., Šnajder, J., Bašić, B.D.: Takelab: Systems for measuring semantic text similarity. In: Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation. pp. 441–448. SemEval '12, Association for Computational Linguistics, Stroudsburg, PA, USA (2012), http://dl.acm.org/citation.cfm?id=2387636.2387708
35. Wang, M., Manning, C.D.: Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In: Proceedings of the 23rd International Conference on Computational Linguistics. pp. 1164–1172. COLING '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010), http://dl.acm.org/citation.cfm?id=1873781.1873912
36. Wang, M., Smith, N.A., Mitamura, T.: What is the jeopardy model? a quasi-synchronous grammar for qa. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). pp. 22–32. Association for Computational Linguistics, Prague, Czech Republic (June 2007), http://www.aclweb.org/anthology/D07-1003
37. Wolpert, D.H.: Stacked generalization. Neural Netw. 5(2), 241–259 (Feb 1992), http://dx.doi.org/10.1016/S0893-6080(05)80023-1
38. Yao, X., Van Durme, B., Callison-Burch, C., Clark, P.: Answer extraction as sequence tagging with tree edit distance. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 858–867. Association for Computational Linguistics, Atlanta, Georgia (June 2013), http://www.aclweb.org/anthology/N13-1106

39. tau Yih, W., Chang, M.W., Meek, C., Pastusiak, A.: Question answering using enhanced lexical semantic models. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. ACL Association for Computational Linguistics (August 2013), http://research.microsoft.com/apps/pubs/default.aspx?id=192357
40. Yu, L., Hermann, K.M., Blunsom, P., Pulman, S.: Deep Learning for Answer Sentence Selection. In: NIPS Deep Learning Workshop (Dec 2014), http://arxiv.org/abs/1412.1632
41. Zhang, X., LeCun, Y.: Text Understanding from Scratch. ArXiv e-prints (Feb 2015)