

An empirical study on large scale text classification with skip-gram embeddings

Georgios Balikas
University of Grenoble Alpes/Coffreo
Grenoble, France
Georgios.Balikas@imag.fr

Massih-Reza Amini
University of Grenoble Alpes
Grenoble, France
Massih-Reza.Amini@imag.fr

ABSTRACT

We investigate the integration of word embeddings as classification features in the setting of large scale text classification. Such representations have been used in a plethora of tasks, however their application in classification scenarios with thousands of classes has not been extensively researched, partially due to hardware limitations. In this work, we examine efficient composition functions to obtain document-level from word-level embeddings and we subsequently investigate their combination with the traditional one-hot-encoding representations. By presenting empirical evidence on large, multi-class, multi-label classification problems, we demonstrate the efficiency and the performance benefits of this combination.

Keywords

Distributed Representations; One-hot-encoding Representations; Neural Networks; Text Classification

1. INTRODUCTION AND PRELIMINARIES

With the proliferation of text data available online, text classification has attracted a lot of interest. Traditionally, N -grams are considered as document features and are subsequently fed to a classifier such as Support Vector Machines (SVMs) [4, 7]. One-hot-encoding representations, although prominent in the literature, have two significant drawbacks: (i) they result in a very high dimensional and sparse feature space, and (ii) they do not encode similarity between words. Lately, a lot of research has been devoted to the direction of distributed representations [6]. Distributed representations of words, are continuous, low dimensional, dense vectors that characterize the meaning and the semantic content of words. Each dimension of the embedding represents a latent feature of the word, hopefully capturing useful syntactic and semantic properties. As a result, semantically similar words, such as “strong” and “powerful”, will be close in the output vectorial space.

In this work, we present a focused contribution as part of an interesting classification application. We investigate the performance of word embeddings learned using the skip-gram model [17] in the context of large scale, multi-label document classification. We report results on real-world classification problems with up to 10,000 classes and we demonstrate a straightforward way to combine document-level embeddings with one-hot-encoding representations. The contributions of our work are twofold: we, first, propose an efficient way to achieve satisfactory classification performance using only embedding representations and, we further improve it by applying a naturally parallelizable fusion mechanism between the distributed representations and one-hot-encoding representations.

Several methods have been proposed for obtaining distributed word-level representations. We cite for instance: [18, 17, 3, 15, 24]. Extensions of those methods have been proposed to cope with larger portions of text such as sentences or paragraphs: [30, 14, 25]. However, generalizing from words to larger text spans can be inefficient, since for every unseen span new passes over a neural network are required. Hence, we focus on methods that given a dictionary of word representations apply composition functions [19, 2] to produce document representations.

Although distributed embeddings have been applied in a plethora of tasks from analogies evaluation [16] to extractive summarization [9], their application on large scale text classification has not been investigated. What is more, most of the work in this direction deals with short text spans, such as sentences or tweets, and the size of the investigated problems with regard to the number of classes is small. Recently, for instance, [11, 12, 26] investigated the problem of short text similarity with applications to classification with a limited number of classes, like binary sentiment analysis [20] and ternary sentiment analysis [21]. More frequently, word embeddings for text classification are used for initializing architectures such as convolutional and recurrent networks. The works of [10] and [13] for instance, are in this line but, again, they limit their study at sentence-length spans. The latter is, also, due to hardware limitations of the GPUs used: their limited memory capacity in conjunction with the vocabulary size of large scale classification problems require many data transfer operations which results in significant overhead. In this work we place ourselves at the large scale setting (number of classes in the order of 10^4) where compositional methods based on neural networks are difficult to be applied.

The remainder of this paper is organised as follows: in

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Neu-IR '16 SIGIR Workshop on Neural Information Retrieval July 17–21, 2016, Pisa, Italy

© 2016 ACM. ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

	Instances	Vocabulary	Avg. labels per instance	Avg. Doc. Size (words)
Train Data	12.5M	868,219	-	215
PubMed ₁₀₀₀	225,000	289,789	1.42	239
PubMed ₅₀₀₀	225,000	302,802	3.23	233
PubMed ₁₀₀₀₀	225,000	303,637	6.64	230

Table 1: Description of the data used to obtain the distributed representations (Train Data) and for classification purposes.

Section 2 we propose and evaluate composition functions for obtaining document from word representations, and in Section 3 we discuss their integration with one-hot-encoding schemes. Finally, Section 4 concludes with remarks to our future work.

2. DISTRIBUTED REPRESENTATIONS AS CLASSIFICATION REPRESENTATIONS

We explore three functions to compose document representations: *min*, *average* and *max* [3, 25] which have been used as simple yet effective methods for compositionality learning in vector-based semantics [19], to obtain a document’s representation. We use the output of each composition function as document features and we evaluate both their classification performance as well as the performance of their concatenation:

$$z_{conc}(doc) = [z_{avg}(doc), z_{min}(doc), z_{max}(doc)]$$

where $z(doc)$ is the representation of a document and $z_x(doc)$ is the result of applying the composition $x \in \{avg, max, min\}$ element-wise, in the distributed vectors of the words of the documents. For instance, assuming we want the representation of “harsh winter”, the composition function requires the vector representations of the words “harsh” and “winter”. Then, applying the *max* function will produce a vector of the same dimensionality with the original representations where each element will have the maximum value of the corresponding elements of the original word representations. In this process, we assume access to a vocabulary of distributed representations, where each word is associated with a D -dimensional vector. This vocabulary, which we assume readily available, may have been generated beforehand.

Table 1 shows the data we use throughout the paper. They are abstracts of biomedical texts from PubMed released by the BioASQ challenge organisers [28] as well as texts from Wikipedia [22]. To obtain the dictionary of word embeddings we used the skipgram model of `word2vec` tool [17]. We have kept the tool’s default parameters for skipgram apart from the number of iterations that we have set to 15. For training the representations, we used 10M PubMed abstracts and we added 2.5M Wikipedia documents for better generalization (“Train Data” of Table 1). In the pre-processing steps we applied lower-casing, we space-padded the punctuation symbols and we filtered the words with less than 5 occurrences. For classification purposes we used the remaining PubMed abstracts from the released data. We created three classification datasets: “PubMed _{x} ” with $x \in \{1, 000, 5, 000, 10, 000\}$ being the x most common classes in the data. We performed a stratified split in train-test (200K-25K instances) parts. In the experiments hereafter, we use SVMs with linear kernel which have been widely used

	Document length (in words)				
	<100	100-200	200-300	300-400	>400
D=50	0.312	0.268	0.266	0.282	0.307
D=200	0.365	0.339	0.337	0.345	0.359
D=400	0.427	0.400	0.396	0.404	0.409

Table 2: The impact of the document length on the classification performance (micro F_1 measure) when using *avg* representations and the embeddings dimension is $D \in \{50, 200, 400\}$. The representations perform better on smaller documents, which is in line with the outcome that the *avg* representations do not capture enough information for large documents.

for text classification. The λ value that controls the importance of the regularization term in the optimization problem was selected using 5-fold cross validation on the training data of each experiment and for each representation. The classification problem is multi-label: each instance is associated with several classes as shown in Table 1. We cope with the multi-label problem using a binary relevance approach [29]. For SVMs, tackling the multi-label problem with binary relevance results in predicting for each instance every label with positive distance from the separating hyperplanes of the one-vs-rest binary problems. If the classifier does not return any label for an instance, the most common label of the training data is assigned. For our implementations, we have used Python’s Scikit-Learn [23].

We now present results for classification experiments on the PubMed_{1,000} and PubMed_{10,000} datasets when the described composition functions are used. Figure 1 shows the scores for the F_1 measure obtained on the test data, when varying the size of the training data. From the Figure, first note that the *avg* function performs better compared to both *min* and *max*. The latter two, perform equally but they are not competitive. However, the best results are consistently obtained with the concatenation (*conc*) of the outputs of the three composition functions. Interestingly, adding the *min* and *max* representations creates a richer representation that benefits the performance. To this direction, note the steep increase in the performance of *conc* representations with the availability of training data: being richer, those representations have bigger discriminative power. Depending on the dataset and the dimension of the representations, the achieved improvements using *conc* vary from ~ 3 -5 F_1 points which is important for such problems. We believe that the *avg* function does not retain enough information for large documents, given that they consist on average of more than 200 words (Table 1). To this end, Table 2 reports the micro F_1 measure for the PubMed_{10,000} dataset, with respect to the document length in words. Note that the best performance is achieved for smaller documents independently of the embedding dimension.

Another observation from Figure 1 and Table 2 concerns the effect of the dimension of word representations in the classification performance. Representations of bigger dimensions benefit performance. In fact, increasing the dimension from 50 to 400, improves the F_1 measure for *conc* for PubMed_{1,000} (resp. PubMed_{10,000}) by ~ 7 (resp. ~ 13) points. Summarizing, we highlight here two of the advantages of the proposed approach: (i) although simple, our

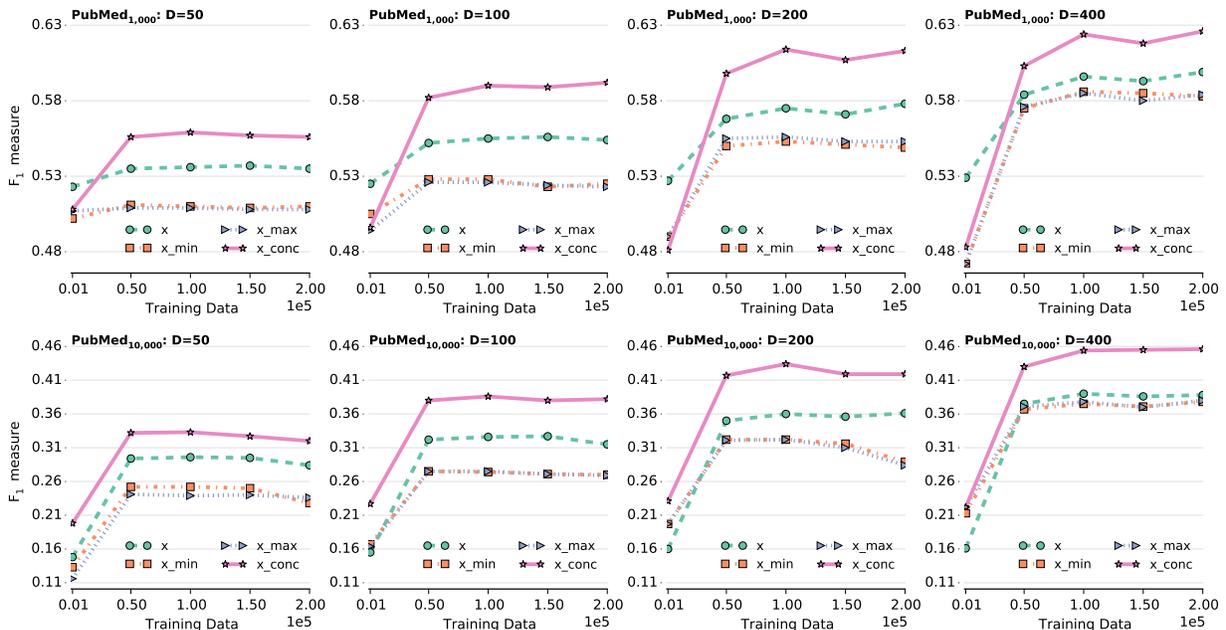


Figure 1: Classification performance of the different composition functions on the PubMed₁₀₀₀ and PubMed₁₀₀₀₀ datasets with representation dimensions $D \in \{50, 100, 200, 400\}$.

composition functions have yielded significant performance improvements and, (ii) their application on large datasets is naturally parallelizable, hence, they can be easily applied on real, large scale problems.

3. COMBINATION OF DISTRIBUTED AND ONE-HOT-ENCODING SCHEMES

In the previous section we have shown that the concatenated representations obtained by the output of the composition functions achieved the best classification performance. We, now, compare them with the traditional one-hot-encoding representations. We focus on two ways of representing text: (i) by using *tf-idf* representations of unigrams, and (ii) by employing a *hash* function [5, 27]. For the former, to generate the *tf-idf* representations we used sub-linear term frequency counts multiplied by their respective smoothed inverse document frequency, i.e., for a term w we have $(1 + \log(tf_w)) * (idf_w + 1)$. For the latter, the hash function, given a text string, transforms it on a numerical value in a pre-specified space, that is used as the index to generate a vector representation. Increasing the output dimension of the hash functions reduces the probability of collisions, i.e. different words mapped on the same vector indices, but also increases the output vector size. We investigate this trade-off in Figure 2: we present the effect of the size of the hash representations with respect to classification performance for PubMed_{5,000} and PubMed_{10,000} which have the biggest vocabulary size. We also report exact timings of each scenario, executed on 10 cores of an Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz. From the figure, note that after 70K features the classification performance does not improve when increasing the size of the feature space. On the other hand, increasing the representation’s dimension, the training time also increases. For reference, *tf-idf* represen-

tations in the same computational setting need 602 sec. and 1203 sec. for PubMed_{5,000} and PubMed_{10,000} respectively. As a result, we set the hash dimension to 70,000 features. Note the significant dimensionality reduction achieved, given the vocabulary sizes of our problems reported in Table 1.

Table 3 details the scores achieved with regard to the representations used. We first discuss the performance when single representations are used: *x_conc*, *tf-idf* and *hash*. Notice that *tf-idf* performs better than both *x_conc* and *hash*, with the latter achieving the lowest performance. The performance of the three representations on PubMed_{1,000} is comparable, but in the bigger classification problems *tf-idf* and performs considerably better. We, thus, consider the *tf-idf* representations as our baseline model, and we examine how the models with concatenated representations behave compared to it.

We investigate now whether the fusion of distributed document representations with the one-hot-encoding representations benefits the classification performance. The last two lines of Table 3 present the performance of the fusion, by concatenation, of the embedding representations with $D \in \{100, 200, 400\}$ with *hash* and *tf-idf*. In the experiments, both *hash* and *tf-idf* consistently achieve better performance when combined with the distributed representations. For instance, for PubMed_{10,000} and $D = 400$ the *tf-idf* (resp. *hash*) representations improve in absolute numbers by 2 (resp. 3.5) F_1 points. We have performed two-sided student’s t-tests ($p < 0.01$) to compare whether the improvements obtained for each classification problem are statistically significant compared to using *tf-idf* representations. Those results, indicated by ([†]) in Table 3, reveal that the concatenation of *tf-idf* with distributed representations improves the classification performance in a statistically significant way. In addition to the important improvements, note than in the fused representations, the effect of

	PubMed _{1,000}			PubMed _{5,000}			PubMed _{10,000}		
	D=100	D=200	D=400	D=100	D=200	D=400	D=100	D=200	D=400
<i>hash</i>	0.63			0.427			0.456		
<i>tf-idf</i>	0.65			0.469			0.492		
<i>x_conc</i>	0.592	0.614	0.626	0.362	0.41	0.436	0.386	0.434	0.454
<i>hash+x_conc</i>	0.651	0.654	0.646	0.464	0.476	0.473	0.488	0.495	0.491
<i>tfidf+x_conc</i>	0.66[†]	0.66[†]	0.656	0.484 [†]	0.486 [†]	0.487[†]	0.507 [†]	0.507 [†]	0.512[†]

Table 3: Classification performance of the different representations. The upper part of the table presents one-hot-encoding methods, while the bottom part methods that depend on the dimensionality of the distributed representations. We report the best performance obtained when the size N of the training data $N \in \{1, 50, 100, 150, 200\} \times 10^3$. The best achieved performance per classification problem is shown in bold. We have performed statistical significance tests to test if improvements obtained compared to *tf-idf* representations are statistically important, which is noted by putting a dagger (\dagger) to the accuracy scores.

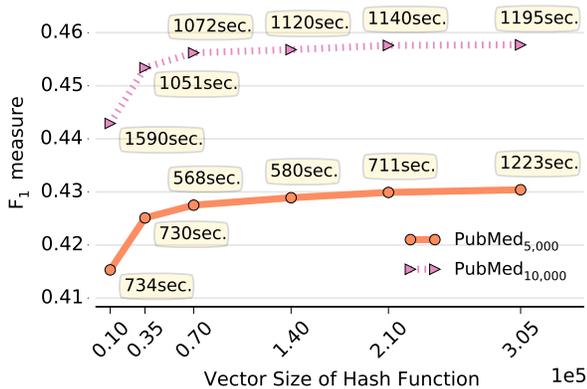


Figure 2: The effect of the vector size generated by the hash function in terms of classification performance and time efficiency for PubMed_{5,000} and PubMed_{10,000}.

D diminishes. For instance in PubMed_{1,000}, one can obtain the optimal performance using embedding dimensions with $D < 400$ and similar observations can be made in the rest of the datasets.

4. CONCLUSION AND DISCUSSION

In this work we have restricted ourselves in representations learned in the word level. This is advantageous in terms of speed since the dictionary of representations can be generated offline. Then, applying composition functions is naturally parallelizable and fast for prediction. However, this poses the challenge of having robust composition functions, which if carefully selected, can result in performance gains such as those reported above. Also, similarly to the bag-of-words paradigm, it does not take into account the word order and the words' grouping in coherent segments like sentences or phrases.

In this line, it would be interesting to further investigate how more complex embeddings such as paragraph vectors [14] perform. Even if such approaches are computationally expensive in the document level, previous research has shown their effectiveness on the sentence level. Hence, a direct extension of this work is to test the investigated composition functions with sentence level representations. In terms of applications, those sentence representations can be

directly used to evaluate the effectiveness of the embeddings and the composition functions. Importantly, their can be evaluated simultaneously in different levels of text granularity from sentences to documents in the framework of passage retrieval and document classification from instance.

Another interesting line of research concerns the memory efficiency of such dense representations. Recent research efforts [1, 8] have investigated ways of compressing the learned (or composed) representations using either linear (e.g. PCA) or non-linear (e.g. auto-encoders) approaches to decrease the memory requirements or the dimension of the representations. Such approaches, apart from having a positive effect on the memory footprint, also have a positive effect on the required computational requirements for training.

In this work we have evaluated different composition functions for obtaining document-level representations using distributed embeddings of words. Summarizing our findings, adding the concatenated vector of word-level skip-gram derived features to *tf-idf* unigrams performs better than *tf-idf* features alone. Also, the result seems to be more pronounced as the representations' cardinality increases and as the output label space increases. Given the obtained improvements, we have also outlined promising future research directions.

5. REFERENCES

- [1] G. Balikas and M.-R. Amini. Multi-label, multi-class classification using polylingual embeddings. In *Advances in Information Retrieval*, pages 723–728. Springer, 2016.
- [2] W. Blacoe and M. Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics, 2012.
- [3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.
- [4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [5] G. Forman and E. Kirshenbaum. Extremely fast text feature extraction for classification and indexing. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages

- 1221–1230. ACM, 2008.
- [6] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.
- [7] T. Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.
- [8] J. Jurgovsky, M. Granitzer, and C. Seifert. Evaluating memory efficiency and robustness of word embeddings. In *Advances in Information Retrieval*, pages 200–211. Springer, 2016.
- [9] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 31–39, 2014.
- [10] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.
- [11] T. Kenter and M. de Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM, 2015.
- [12] J. Kim, F. Rousseau, and M. Vazirgiannis. Convolutional sentence kernel from word embeddings for short text categorization. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. In *EMNLP*, volume 15, 2015.
- [13] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
- [14] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.
- [15] O. Levy and Y. Goldberg. Dependency-based word embeddings. In *ACL (2)*, pages 302–308, 2014.
- [16] O. Levy, Y. Goldberg, and I. Ramat-Gan. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180, 2014.
- [17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [19] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.
- [20] P. Nakov, Z. Kozareva, A. Ritter, S. Rosenthal, V. Stoyanov, and T. Wilson. Semeval-2013 task 2: Sentiment analysis in twitter. 2013.
- [21] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.
- [22] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015.
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [24] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.
- [25] R. Socher, E. H. Huang, J. Pennington, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.
- [26] Y. Song and D. Roth. Unsupervised sparse vector densification for short text similarity. *Proc. North Am. Chapter Assoc. Computat. Linguistics*, pages 1275–1280, 2015.
- [27] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.
- [28] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. R. Alvers, D. Weissenborn, A. Krithara, S. Petridis, D. Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1, 2015.
- [29] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Dept. of Informatics, Aristotle University of Thessaloniki, Greece*, 2006.
- [30] H. Zhao, Z. Lu, and P. Poupard. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*, 2015.