BMC Bioinformatics

RESEARCH ARTICLE Open Access

Fast and scalable neural embedding models for biomedical sentence classification

Asan Agibetov, Kathrin Blagec, Hong Xu and Matthias Samwald*

Abstract

Background: Biomedical literature is expanding rapidly, and tools that help locate information of interest are needed. To this end, a multitude of different approaches for classifying sentences in biomedical publications according to their coarse semantic and rhetoric categories (e.g., Background, Methods, Results, Conclusions) have been devised, with recent state-of-the-art results reported for a complex deep learning model. Recent evidence showed that shallow and wide neural models such as fastText can provide results that are competitive or superior to complex deep learning models while requiring drastically lower training times and having better scalability. We analyze the efficacy of the fastText model in the classification of biomedical sentences in the PubMed 200k RCT benchmark, and introduce a simple pre-processing step that enables the application of fastText on sentence sequences. Furthermore, we explore the utility of two unsupervised pre-training approaches in scenarios where labeled training data are limited.

Results: Our fastText-based methodology yields a state-of-the-art F1 score of .917 on the PubMed 200k benchmark when sentence ordering is taken into account, with a training time of only 73 s on standard hardware. Applying fastText on single sentences, without taking sentence ordering into account, yielded an F1 score of .852 (training time 13 s). Unsupervised pre-training of N-gram vectors greatly improved the results for small training set sizes, with an increase of F1 score of .21 to .74 when trained on only 1000 randomly picked sentences without taking sentence ordering into account.

Conclusions: Because of it's ease of use and performance, fastText should be among the first choices of tools when tackling biomedical text classification problems with large corpora. Unsupervised pre-training of N-gram vectors on domain-specific corpora also makes it possible to apply fastText when labeled training data are limited.

Keywords: Natural language processing, Text classification, Neural networks, Word vector models, FastText, Scientific abstracts

Background

Biomedical literature is vast and rapidly expanding. With over 27 million articles currently in PubMed, it is increasingly difficult for researchers and healthcare professionals to efficiently search, extract and synthesize knowledge from diverse publications. Technological solutions that help users locate text snippets of interest in a quickly and highly targeted manner are needed. To this end, a multitude of different approaches for classifying sentences in PubMed abstracts according to their coarse semantic and rhetoric categories (e.g., Introduction/Background, Methods, Results, Conclusions) has been devised. Many different methodological approaches to this task have been described in literature, including naive Bayes [1–4], support vector machines [2, 3, 5], Hidden Markov models [6], Conditional Random fields (CRFs) [7–9], and advanced, semi-automatic engineering of features [10].

Recently, a new state-of-the-art methodology for the task of sequential sentence categorization in PubMed abstracts based on a deep learning model has been reported [11]. The model is based on a sophisticated architecture with bi-directional Long short-term memory (LSTM) layers applied to characters and word tokens, taking sentence ordering in the abstract into account. The authors demonstrate superior results of this deep model on the established NICTA-PIBOSO corpus [9, 11], as well as the

*Correspondence: matthias.samwald@meduniwien.ac.at
Section for Artificial Intelligence and Decision Support, Medical University of Vienna, Währinger Strasse 25A, OG1, 1090 Vienna, Austria

newly created, larger PubMed 200k RCT benchmark dataset [12].

Training deep neural networks on large text data is often not trivial, since they require careful hyperparameter optimization to provide good results, require the use of graphics processor units (GPUs) for performant training, and often take a long time to train. Recent evidence showed that shallow and wide neural models such as the fastText model [13] based on token embeddings can provide results that are competitive with complex deep learning models while requiring drastically lower training times and having better scalability [13, 14] without necessitating the utilization of GPUs.

In this work, we analyze the applicability of the fastText model on the classification of biomedical sentences in the PubMed 200k RCT benchmark. Specifically, we demonstrate how simple corpus preprocessing can be used to train fastText on sentence sequences instead of singular sentences, and how such an approach can yield state-of-the-art results while retaining very short training times. Furthermore, we explore approaches of semi-supervised learning, where models are pre-trained through unsupervised training on predicting word contexts or sentence reconstruction tasks, and demonstrate that unsupervised pre-training greatly improves classification quality when the labeled training data are limited.

Methods

Datasets

PubMed 200k RCT is a new dataset derived from PubMed that is intended as a benchmark for sequential sentence classification. It is made up of two subsets, PubMed 200k and PubMed 20k. PubMed 200k contains approximately 200,000 abstracts of randomized controlled trials (RCTs), with a total of 2,2 million sentences. PubMed 20k is a smaller version of the PubMed 200k dataset, containing only 20,000 abstracts. Each sentence in the datasets is labeled with one of five labels ('objective', 'background', 'method', 'result' or 'conclusion'). The datasets are divided into predefined training, validation and test data splits. Details about the construction of the datasets and dataset statistics can be found in [11].

To investigate if additional and more diverse training data could further improve classification results, we also created an extended corpus, where the training data of the PubMed 200k dataset were augmented with additional structured abstracts derived from journals with a medical focus indexed by PubMed. The training split of this extended corpus contains 872,000 abstracts (compared to only 190,000 abstracts in the training split of PubMed 200k). Validation and test data for the extended corpus remained the same as for the PubMed 200k dataset.

All corpora were lower-cased and punctuation was separated from words through added whitespace. Statistics of all datasets are summarized in Table 1.

Neural embedding models for n-gram embeddings

Recently introduced neural embedding word vector models are based on the so-called matrix factor models (sometimes also referred to as bi-linear models). The models are flexible as they can be easily adapted to both supervised and unsupervised learning, and can also be extended to n-gram embedding problems by simply creating additional embeddings for encountered n-grams. Embedding vectors are learned through a shallow neural network with a single hidden layer. These embedding vectors are learned implicitly, and collected in the weight matrix of the hidden layer (Fig. 1).

To simplify the notation, we present below the most general formalized form of these models, adapted from [15]:

$$\arg\min_{\mathbf{U},\mathbf{V}} \sum_{S \in C} f_S(\mathbf{U}\mathbf{V}\iota_S). \tag{1}$$

The goal of this optimization problem is to find the parameter matrices $\mathbf{U} \in \mathbb{R}^{k\times h}, \mathbf{V} \in \mathbb{R}^{h\times|\mathcal{V}|}$ ($\mathcal{V}$ - vocabulary, set of all words or n-grams) that minimize (1). Learned embedding vectors will have dimension h and will be stored as columns in the matrix $\mathbf{V}$. S are either the fixed-length context windows in the corpus $\mathcal{C}$ in the continuous bag-of-words (CBOW) model [16, 17], or entire sentences in the sent2vec model. $\iota_S \in 0, 1^{|\mathcal{V}|}$ (binary indicator vector) is the bag-of-words encoding of S. In the case of supervised learning $k \ll |\mathcal{V}|$ is the number of class labels, analogously $k = |\mathcal{V}|$ in the unsupervised case. The cost functions $f_s : \mathbb{R}^k \mapsto \mathbb{R}$ only depend on a single row of its input matrix (as a result of $\mathbf{UV}$), and will have different forms depending on the learning task (we will detail this aspect below).

Supervised versus unsupervised training

In the following sections we describe the application of fully supervised learning as well as a mix of unsupervised learning followed by supervised learning. In the fully supervised training setup, a completely uninitialized embedding model was trained to predict labels

Table 1 $|\mathcal{V}|$ denotes vocabulary size. For train, validation and test datasets, the number of abstracts followed and the number of sentences (in parentheses) are shown

Dataset	$\lvert\mathcal{V}\rvert$	Train	Validation	Test
PubMed 20k	68k	15k (180k)	2.5k (30k)	2.5k (30k)
PubMed 200k	331k	190k (2.2M)	2.5k (29k)	2.5k (29k)
Extended corpus	451k	872k (10,3M)	2.5k (29k)	2.5k (29k)

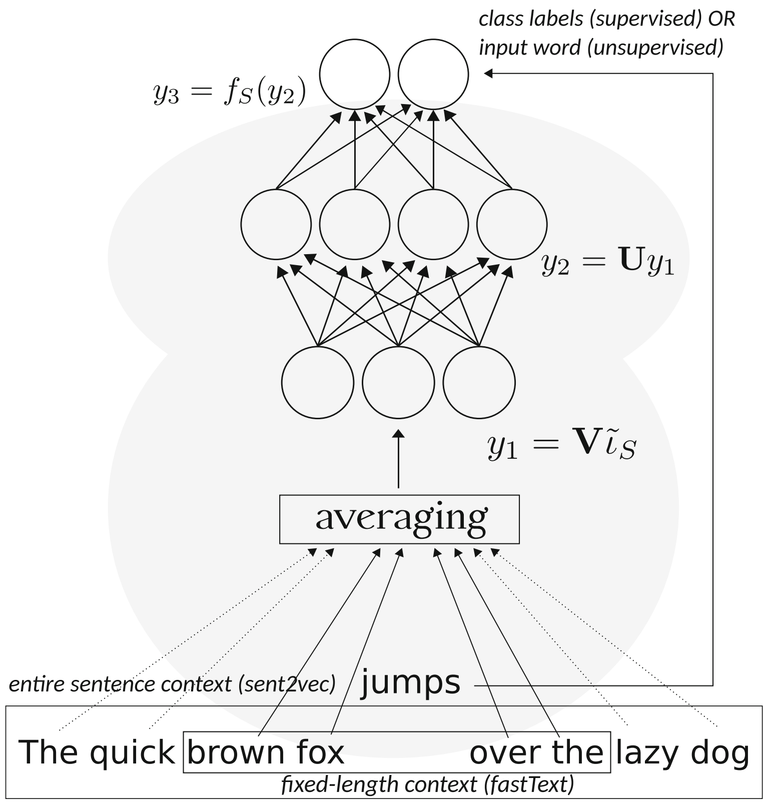
class labels (supervised) OR
input word (unsupervised)
$y_3 = f_S(y_2)$
$y_2 = \mathbf{U}y_1$
$y_1 = \mathbf{V}\tilde{\iota}_S$
averaging
entire sentence context (sent2vec)
jumps
The quick brown fox over the lazy dog
fixed-length context (fastText)

Fig. 1 Schematic representation of the neural embedding model for sentences (supervised and unsupervised) consisting of two embedding layers and a final softmax layer over k classes (for the supervised case). In the unsupervised case $k = |\mathcal{V}|$ and the softmax outputs the probability of the target word (over all vocabulary, as in C-BOW model) given its context: fixed-length context for fastText and entire sentence context for sent2vec. Independently of the training mode (e.g., supervised vs unsupervised) word embeddings are stored as columns in the weight matrix V of the first embedding layer. Note that in the unsupervised case the rows of the weight matrix U of the second embedding layer represent the embeddings for the "negative" words; these embeddings however are not used for the downstream machine learning tasks. In all instances the averaging of embeddings of constituent tokens ($\hat{\iota}_S$) is performed by fastText (sent2vec implementation is based on fastText)

and the resulting model was evaluated. In mixed setups, an unsupervised pre-training step was used to generate models that would then be used as the basis for supervised training in a second step. In unsupervised learning, the model was trained to predict the internal structure of the text, without requiring explicit labels (i.e., 'without supervision'). Such unsupervised pre-training can induce useful representations of the content of sentences so that downstream supervised classification can potentially succeed with fewer training examples.

Supervised n-gram model of fastText

We used the fastText natural language processing library for sentence classification of biomedical text [13]. The methodology for sentence classification relies on the supervised n-gram classification model from fastText, where each sentence embedding is learned iteratively, based on the n-grams that appear in it. To this end each sentence is represented as a normalized bag of n-grams that capture the local word order. The fastText model can be seen as a shallow neural network that derives its capabilities by scaling up the number of learnable vector embeddings of n-gram features that are fed into the network. By adapting straightforwardly (1), we can represent this supervised classification model as a minimization of the negative log-likelihood over the class labels problem, as shown below:

$$\underset{\mathbf{U},\mathbf{V}}{\arg\min} - \frac{1}{N}\sum_{S\in C} y_S \log(f_S(\mathbf{U}\mathbf{V}\tilde{\iota}_S)). \qquad (2)$$

We note that y_S is the label of the fixed-length context S, $k \ll |\mathcal{V}|$ is the number of class labels, and $\tilde{\iota}_S$ is the normalized bag of features encoding of S (i.e., $\sum_{x\in\tilde{\iota}_S} x = 1$). Despite the simplicity of the model, it was shown to be competitive or superior to many deep neural architectures in text classification tasks [13], but also other tasks such as knowledge-base completion [14], with training times and resource requirements that are of often superior by orders of magnitude.

The fastText library was downloaded from GitHub (November 14, 2017) and compiled. For each experiment, an exhaustive hyperparameter grid search was conducted, hyperparameters considered were dimensionality of the n-gram embedding (10, 20 and 50 dimensions), word N-gram sizes (1, 2, 3 or 4 words), and number of training epochs (between 1 and 8 training epochs for 20k datasets and between 1 and 4 training epochs for larger datasets). All other hyperparameters were left at their default settings. All experiments were run on a machine with the Ubuntu 16.04 operating system, Intel Core i7-6700 4x3.40 Ghz processor and 32 GB RAM. Experiments were run in Jupyter notebooks [18] under Python 3.6. The scikit-learn package [19] was used for statistical analyses. Jupyter notebooks used for the experiments are available on GitHub [20] and include detailed information on hyperparamter sweeps and performance for each hyperparameter setting.

We investigated classification quality based on F1 scores weighted by the support of each label (i.e., how frequent each label occurred in the dataset). The scikit-learn package [19] was used for statistical analysis.

Since fastText is based on a relatively simple bag-of-N-grams model, it cannot utilize data on sentence sequences as-is. To utilize the sequential nature of sentences in PubMed abstracts, we devised a simple pre-processing step that gives fastText information about the position of a sentence in the abstract, as well as the content of preceding and following sentences in the abstract through additional tokens. These additional tokens are added to the sentence that is to be classified, and the standard bag-of-N-grams model of fastText is then trained on this enhanced sentence representation. Even though this increases the vocabulary of N-grams and the number of N-grams used to classify each sentence, fastText still remains highly performant. In pre-processing, the sentence representation was augmented by adding numeric sentence position information, as well as representations of the two preceding and trailing sentences. Tokens in these context sentences are altered by adding prefixes (e.g., `'-1_'` for the directly preceding sentence so that sentence sequence information is preserved in the fastText model). As an abstract example, the following represents a sequence of five sentences, with 'aaa', 'bbb' exemplifying tokens in each sentence:

```
Label:BACKGROUND   aaa bbb ccc
Label:BACKGROUND   ddd eee fff
Label:OBJECTIVE    ggg hhh iii
Label:METHODS      jjj kkk lll
Label:RESULT       mmm nnn ooo
```

The preprocessing algorithm turned the third sentence (with the 'objective' label) into the following representation with additional tokens for training fastText (added tokens representing numeric sentence position and context sentences):

```
Label:OBJECTIVE  ggg hhh iii
                 sentence_3 of_5
                 -2_aaa -2_bbb -2_ccc
                 -1_ddd -1_eee -1_fff
                 1_jjj 1_kkk 1_lll
                 2_mmm 2_nnn 2_ooo
```

We also conducted ablation experiments where we removed parts of the preprocessing algorithm to quantify the benefit of each part of the algorithm.

Unsupervised model for sentence embeddings with sent2vec and fastText

sent2vec is an unsupervised model for learning universal sentence embeddings. It is an extension of the fixed-length word-contexts from CBOW to a larger sentence context of variable length. This extension allows for learning sentence embeddings in an additive manner by minimizing the unsupervised objective loss function (3). Thus, a sentence embedding $\mathbf{v}_S$ is generated by averaging the word (or n-gram) embeddings of its constituents.

$$\mathbf{v}_S := \frac{1}{|R(S)|}\mathbf{V}\iota_{R(S)} = \frac{1}{|R(S)|}\sum_{w\in R(S)} \mathbf{v}_w,$$

where $R(S)$ is the list of n-grams (including unigrams) present in the sentence S. In the process of minimization we learn source $\mathbf{v}_w$ and target $\mathbf{u}_w$ embeddings for each word w in the vocabulary $\mathcal{V}$ as in (1). Similar to CBOW, sent2vec predicts a missing word from the context (which in the case of sent2vec is the entire sentence) via an objective function that models the softmax output approximated by negative sampling. Coupled with the logistic sigmoid function $l(x) = \log\left(\frac{1}{1+e^{-x}}\right)$, the unsupervised training objective function is formulated as follows

$$\arg\min_{\mathbf{U},\mathbf{V}} \sum_{S\in\mathcal{C}}\sum_{w_t\in S}\left(l\left(\mathbf{u}_{\mathbf{w_t}}{}^T\mathbf{v}_{S\setminus w_t}\right) + \sum_{w'\in N_{w_t}} l\left(\mathbf{u}_{\mathbf{w'}}{}^T\mathbf{v}_{S\setminus w_t}\right)\right), \tag{3}$$

where S corresponds to the current sentence and N_{w_t} is the set of words sampled negatively for the word w_t.

This unsupervised model can also be used with fastText, however, the main difference will be in the definition of the context: fixed-length context for fastText and entire sentences (variable length context) for sent2vec. Detailed comparison and the evaluation of the two models can be found in [15].

To simulate settings in which limited training data were available, we created limited training datasets by randomly sampling sentences from the PubMed 200k training corpus. The number of sampled sentences for training was varied from 100 as the lowest to 180 000 as the highest sentence count, the latter being roughly equivalent to the number of sentences in the PubMed 20k training corpus. Three classifier setups were compared:

- *fastText:* The standard, fully supervised fastText algorithm
- *fastText, pre-trained:* A semi-supervised fastText model where N-gram embeddings were pretrained in an unsupervised way on the full PubMed 200k training text corpus (disregarding any labels) before switching to supervised training for sentence classification
- *sent2vec + multi-layer perceptron (MLP):* whole sentence embeddings were trained in an unsupervised way on the full PubMed 200k training corpus, vector representations of sentences generated by sent2vec were then used to train a multilayer perceptron with a single hidden layer (size 100 neurons) in a supervised way

Single sentences without sentence context or ordering were used for the evaluations of semi-supervised training. Hyperparameter settings for the fastText models were taken from the best-performing model established in the unsupervised task. The test set for each run consisted of 20 000 randomly sampled sentences that did not overlap with the training set sentences. For each training set size, each classifier was run on five random samples of training and test sentence sets, and the median weighted F1 scores on the test sets were calculated.

Code availability

Jupyter notebooks with code for training, testing and statistical analysis procedures, as well as trained models are available on GitHub [20].

Results

Fully supervised training

An overview of the results of our fastText models compared to other published results is shown in Table 2. The fastText model with sentence context and numeric sentence position provides a result for the PubMed 200k benchmark that outperforms the current state-of-the-art model by a small margin (F1 score of .917 vs. F1 score of .916 reported in [11] for the sophisticated bi-ANN deep learning model), while retaining a short training time of 73 s. For the smaller PubMed 20k corpus, fastText results are slightly worse than those of the bi-ANN model (.896 vs. .900), while completing training in only 11 s. Expectedly, the fastText classifier based only on single sentences (without taking information on sentence sequence, sentence context or sentence position into account), yields lower F1 scores (.852 for PubMed 200k and .825 for PubMed 20k).

fastText with sentence context and numeric text position trained on the extended corpus and evaluated on the PubMed 200k dataset achieves an F1 score of .919, showing that utilizing a larger training corpus can further improve classification quality. The extended training set size did not yield an improvement for the single-sentence fastText model.

Ablation experiments on the PubMed 200k benchmark showed that both the numeric sentence position and the addition of context sentences greatly benefitted classification quality, with the removal of context sentences yielding a greater degradation of quality than removal of numeric sentence positions (Table 3).

To further analyse the results of our best fastText classifier on the PubMed 200k dataset, we calculated the confusion matrix for the predictions on the test data split (Table 4). We found that while classification of methods, results and conclusion sentences is almost perfect, objective and background sentences are often mixed up. This problem has also been noted with the predictions of the deep bi-ANN model of [11].

Semi-supervised training

We found that the semi-supervised approaches and the fully unsupervised approach yield equal classification qualities for larger training set sizes of > 50000 training examples, where classification performance approached a ceiling with an F1 score of approximately 0.84 (Fig. 2). However, unsupervised pre-training yields a decisive advantage at smaller training set sizes. When training on a small training set of 1000 sentences, the fully unsupervised model did not yield useful results (weighted F1 of 0.21), while sent2vec+MLP and fastText with pre-trained word vectors yield far superior F1 scores (0.74 and 0.73, respectively). At even smaller training set sizes, fastText with pre-trained N-gram vectors was superior to sent2vec+MLP, while the two methodologies yielded similar results for training set sizes of 1000 sentences and above. Given these results and the greater ease of use of the methodology of using fastText with pre-trained, domain-specific N-gram vectors, this appears to be the methodology of choice.

Table 2 Weighted F1 scores for various models trained on single sentences. Best results for each dataset are printed in bold. For our models, training time is given (for hyperparameter settings yielding the shown score)

Model	PubMed 20k	PubMed 200k	Extended corpus
Logistic regression model (LR) [11][a]	.831	.859 (33,006 s)	-
Forward artificial neural network (ANN) [11][a]	.861	.884	-
Conditional random field (CRF) [11][a]	.895	.915 (4867 s)	-
bi-ANN [1][a]	**.900**	.916	-
fastText single-sentence (ours)	.825 (5 s)	.852 (13 s)	.852 (61 s)
fastText with sentence context and numeric sentence position (ours)	.896 (11 s)	**.917 (73 s)**	**.919 (183 s)**

[a] Result and runtime reported by [11]; the reported runtimes given by authors include both training and testing time while we report only training time. Testing of a trained fastText model took approx. 15 s with the evaluation tool supplied by the fastText library.

Discussion

fastText yielded good results with very low training times for both unsupervised and semi-supervised training. While we introduced a simple pre-proces5sing algorithm for utilizing sentence sequence information, it is noteworthy that fastText functioned well without much additional pre-processing (e.g., rare words and numbers were not removed from the corpora, resulting in a very high number of tokens fed into fastText that were not relevant to the classification task). Results were good across a wide variety of hyperparameter settings, demonstrating that the algorithm is quite robust. The main determinant of classification performance was the number of epochs, where, expectedly, low epoch numbers led to underfitting and higher numbers led to overfitting. Since fastText does not have an in-built methodology for early stopping, it is therefore indispensable to set up external scripts that do this hyperparameter optimization.

fastText requires little data pre-processing, little hyperparameter tuning, does not require a GPU, optional engineering of task-specific pre-processing steps is simple and intuitive, and training of models is very fast. We therefore suggest that fastText should be among the first methodologies to consider in biomedical text classification tasks. Overall, the robustness of results across a wide range of hyperparameter settings makes it possible to achieve good results with less hyperparameter tuning, which can further ease training when compared to more complex deep-learning methods that usually require extensive hyperparameter tuning.

Table 3 Ablation experiments based on PubMed 200k dataset

	Weighted F1 score
Full model	.917
Removed numeric sentence position	.912
Removed sentence context	.904
Removed both sentence context and numeric sentence position (single sentence model)	.852

The finding that the bag-of-N-grams model of fastText achieved similar classification quality as more detailed representations of sentence structure (e.g. precise word sequences as captured by recurrent neural networks) suggests that these detailed representations are not required for classification tasks. The bag-of-N-grams model captures enough of the "rough content" of a specific sentence, which seems to be sufficient for most classification tasks. Providing contextual information, such as the content of preceding and following sentences and the position of the sentence within the abstract improve classification quality, but can also also be captured through a bag-of-N-grams representation through the pre-processing algorithm we introduced.

How is fastText able to tackle difficult classification tasks with such short CPU training time? While other more complex neural network architectures capture word combinations and n-gram patterns through a cascade of recurrent or convolutional layers, fastText relies solely on scaling up the "width" of the shallow neural network, and on the distributional hypothesis of semantics of the words and n-grams within a given context. The neural embedding model includes two embedding layers, and the weight matrices of these embedding layers are the parameters of

Table 4 Confusion matrix for test results for the PubMed 200k dataset, yielded by the fastText model with sentence context and numeric sentence position

True label	Predicted label				
	Objective	Background	Methods	Results	Conclusions
Objective	**1704 (72%)**	573	98	0	2
Background	471	**2051 (77%)**	91	8	42
Methods	25	36	**9375 (96%)**	296	18
Results	3	3	395	**9744 (95%)**	131
Conclusions	1	23	13	203	**4186 (95%)**

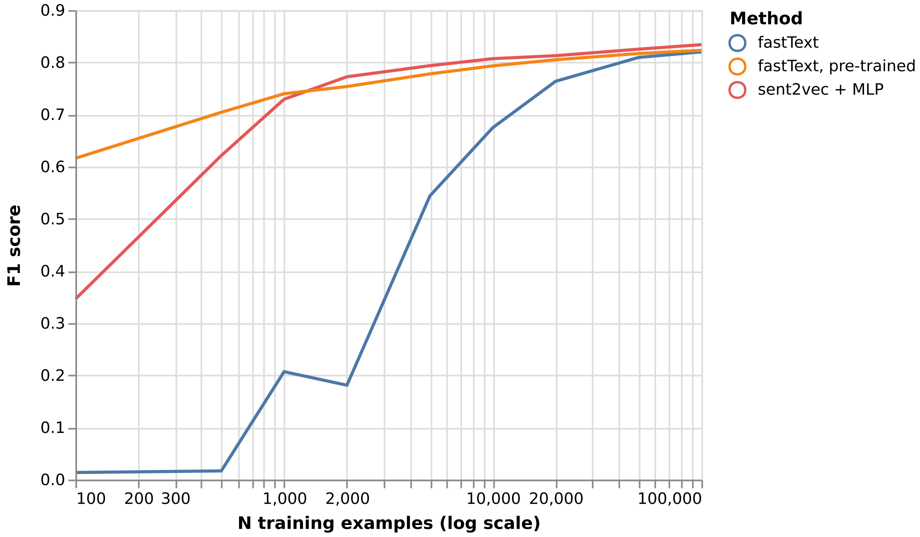

Fig. 2 Weighted F1 score of test set predictions for different training set sizes

the model. The parameters are updated when the neural network is trained through the gradient descent-based optimization methods. At the end of training the columns of the weight matrix of the first embedding layer represent the final learned word embeddings (latent vector representations). The implementation of this tool is optimized to fast updates of the model parameters (i.e., embeddings of words and n-grams), in such a way that it scales very well for a very large number of tokens ("rows" in the weight matrix of the first embedding layer). While other models scale exponentially as we increase the number of tokens to capture the semantics of sentence embeddings, fastText scales linearly. Finally, provided that abstract texts use different n-gram patterns, for the different parts of the abstracts (e.g., conclusions, results), the classification task boils down to capturing the most salient features to discern these n-gram patterns. fastText is able to capture those features by increasing the width instead of the depth of its neural network architecture, which is why it is able to deal with the classification task on the subparts of the biomedical abstracts so quickly.

The ability of the fastText model to scale to very large vocabularies and n-grams can be used to represent more complex structures by simply representing them as additional entities that can be embedded. In the context of this work, we represented words in context sentences as separate entities, which multiplied the number of entities that are embedded, but was nonetheless easily handled by the algorithm.

Limitations

A limitation of the fastText algorithm is that is not easily applicable to multi-label prediction (i.e., settings where a varying number of labels apply to one input text, instead of precisely one label per input text). fastText generates a single probability distribution over all labels with a softmax function (i.e. probability of all labels adding up to 1), which is not ideal if multiple labels can be correct for the same input text. While workarounds for this problem are available, it remains a limitation in use-cases that require multi-label prediction.

Another potential limitation of this work lies in the PubMed 200k RCT benchmark dataset. Both the models of [11] and our models have difficulty discerning sentences from the background and objective classes, and a sizable fraction of the difference between perfect F1 scores and observed F1 scores is caused by this difficulty. Reviewing a sample of abstracts in the dataset suggests that these classes are used in an inconsistent manner, and successfully discerning these classes might be difficult even for human expert annotators. This could imply that current best results of automated classifiers are already very close to the best possible scores that can be achieved, which would limit the utility of the benchmark dataset. Ideally, a gold-standard score based on the performance of expert human curators should be established for this benchmark.

Future work

The presented methodology should be evaluated with other sentence classification use-cases and benchmarks. Further research should also be devoted to the exploration of semi-supervised approaches that combined unsupervised pre-training on large text corpora with supervised training on smaller corpora. Of special interest in this regard might be methods that are based on ensembles of different unsupervised sentence representations, such as an ensemble of the sent2vec model trained on single sentences with other models that work on sentence sequences, such as the deep Skip-Thoughts model [21]. The classifiers developed in this work could also be integrated into larger natural language processing pipelines for biomedical information extraction. For example, selecting only sentences that are classified as conclusion sentences might provide a better signal-to-noise ratio than using full abstracts for term co-occurrence analysis and other text extraction approaches. In terms of software for end-users (i.e., medical professionals and biomedical researchers) we plan to integrate the classifiers created in this work into a new version of the FindMeEvidence search system [22]. The goal of this search system is to provide users with a quick overview of the main findings of biomedical PubMed research articles. The classifier will be used to provide a condensed overview of the key findings of articles that matched a user query. It has also recently been demonstrated that fastText can provide competitive results at low training times when applied for link prediction in knowledge graphs [14], a domain that is fundamentally different from text classification. In future research we will further investigate if fastText can be successfully applied to more such types of data through preprocessing tricks similar to the ones we demonstrated in this paper.

Conclusion

We demonstrated that through utilizing a simple preprocessing algorithm, the fastText model can provide state-of-the-art results in biomedical sentence classification at low computational cost. We characterized semi-supervised approaches based on neural embeddings that enable good classification results with a lower number of training examples compared to a fully supervised approach. We suggest that highly performant, shallow neural embedding models such as fastText should be among the first methodologies to be considered when classification needs to be made on data that can be represented as bags of tokens. We demonstrated that more structured data can be utilized through preprocessing. Future work should investigate the potential of this approach for a wide variety of data that go beyond simple text, such as structured knowledge graphs.

Abbreviations

ANN: Artificial neural network; CBOW: Continuous bag-of-words; CRF: Conditional random field; GPU: Graphics processor unit; LSTM: Long short-term memory; LR: Logistic regression; MLP: Multi-layer perceptron; RCT: Randomized controlled trial

Acknowledgements

We want to thank the development teams behind fastText and sent2vec for making these software packages openly available.

Funding

A part of the research leading to these results has received funding from the European Community's Horizon 2020 Programme under grant agreement No. 668353 (U-PGx).

Availability of data and materials

Datasets generated and/or analysed during the current study, Jupyter notebooks and trained models are available through Github at https://github.com/matthias-samwald/Fast-and-scalable-neural-embedding-models-for-biomedical-sentence-classification

Authors' contributions

MS devised the study, carried out experiments and conducted data analysis. AA carried out and conducted data analysis. HX conducted data preparation. MS, AA, KB and HX participated in authoring the manuscript. All authors read and approved the final manuscript.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 4 May 2018 Accepted: 16 November 2018
Published online: 22 December 2018

References

1. Ruch P, Boyer C, Chichester C, Tbahriti I, Geissbühler A, Fabry P, Gobeill J, Pillet V, Rebholz-Schuhmann D, Lovis C, Veuthey A-L. Using argumentation to extract key sentences from biomedical abstracts. Int J Med Inform. 2007;76(2-3):195–200.
2. Guo Y, Korhonen A, Liakata M, Karolinska IS, Sun L, Stenius U. Identifying the information structure of scientific abstracts: An investigation of three different schemes. In: Proceedings of the 2010 Workshop on Biomedical Natural Language Processing, BioNLP '10. Stroudsburg: Association for Computational Linguistics; 2010. p. 99–107.
3. Guo Y, Korhonen A, Silins I, Stenius U. Weakly supervised learning of information structure of scientific abstracts–is it accurate enough to benefit real-world tasks in biomedicine? Bioinformatics. 2011;27(22): 3179–85.
4. Huang K-C, Chiang I-J, Xiao F, Liao C-C, Liu CC-H, Wong J-M. PICO element detection in medical text without metadata: are first sentences enough? J Biomed Inform. 2013;46(5):940–6.
5. Yamamoto Y, Takagi T. A sentence classification system for multi biomedical literature summarization. In: 21st International Conference on Data Engineering Workshops (ICDEW'05). Washington, DC: IEEE; 2005. p. 1163.
6. Lin J, Karakos D, Demner-Fushman D, Khudanpur S. Generative content models for structural analysis of medical abstracts. In: Proceedings of the Workshop on Linking Natural Language Processing and Biology: Towards Deeper Biological Literature Analysis, BioNLP '06. Stroudsburg: Association for Computational Linguistics; 2006. p. 65–72.
7. Hirohata K, Okazaki N, Ananiadou S, Ishizuka M. Identifying sections in scientific abstracts using conditional random fields. In: Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-I; 2008.
8. Lin RTK, Dai H-J, Bow Y-Y, Chiu JL-T, Tsai RT-H. Using conditional random fields for result identification in biomedical abstracts. Integr Comput-Aided Eng. 2009;16(4):339–52.
9. Kim SN, Martinez D, Cavedon L, Yencken L. Automatic classification of sentences to support evidence based medicine. BMC Bioinformatics. 2011;12(Suppl 2):5.
10. Nam S, Jeong S, Kim S-K, Kim H-G, Ngo V, Zong N. Structuralizing biomedical abstracts with discriminative linguistic features. Comput Biol Med. 2016;79:276–85.

11. Dernoncourt F, Lee JY, Szolovits P. Neural networks for joint sentence classification in medical paper abstracts. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. Valencia: Association for Computational Linguistics; 2017. p. 694–700.
12. Dernoncourt F, Lee JY. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing: Volume 2: Short Papers. Taipei: Asian Federation of Natural Language Processing; 2017. p. 308–313.
13. Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers. Valencia: Association for Computational Linguistics; 2017. p. 427–431.
14. Joulin A, Grave E, Bojanowski P, Nickel M, Mikolov T. Fast linear model for knowledge graph embeddings. arXiv:1710.10881 [stat.ML]. 2017.
15. Pagliardini M, Gupta P, Jaggi M. Unsupervised learning of sentence embeddings using compositional n-gram features. arXiv:1703.02507 [cs]. 2017.
16. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Bag of tricks for efficient text classification. Red Hook: Curran Associates Inc.; 2013. p. 3111–3119.
17. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781 [cs.CL]. 2013.
18. Project Jupyter |Home. https://jupyter.org/Last. Accessed 3 May 2018.
19. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine learning in python. J Mach Learn Res. 2011;12:2825–2830.
20. GitHub repository. https://github.com/matthias-samwald/Fast-and-scalable-neural-embedding-models-for-biomedical-sentence-classification/. Accessed 3 May 2018.
21. Kiros R, Zhu Y, Salakhutdinov RR, Zemel R, Urtasun R, Torralba A, Fidler S. Skip-thought vectors. In: Advances in Neural Information Processing Systems 28. Red Hook: Curran Associates, Inc.; 2015. p. 3294–3302.
22. Samwald M, Hanbury A. An open-source, mobile-friendly search engine for public medical knowledge. Stud Health Technol Inform. 2014;205: 358–62.