UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS ASSIGNMENT No. 1418

Classification of Large-Scale Biological Annotations Using Word Embeddings Derived from Corpora of Biomedical Research Literature

Adriano Baćac

Zagreb, June 2017.

Replace with original assignment page

Assignment text:

Document classification tasks are commonly addressed using a simple bag-of-words representation, which produces very sparse data sets and ignores word semantics. Word embeddings can provide rich contextual information for words. Recent analyses suggest that they may increase predictive accuracy on certain text classification tasks. Since embeddings are typically trained on broad, general-purpose text libraries, they might not be appropriate for specialized corpora. In this work custom word embeddings for scientific literature in the biomedical domain, and additionally more subject-specific subsets, will be trained that can result in representation that better encodes semantic meaning. Novel embedding models will next be used to represent words in biomedical texts, and as an input to recurrent neural networks. These models will be further used to train classifiers for discriminating large-scale biological annotations, such as phenotypes and gene functions.

I wish to thank my mentor Mile Šikić for support and guidance during all my years being a student.

I would also like to thank Fran Supek and Maria Brbić from the Ruđer Bošković Institute in Zagreb for numerous suggestions and advice that helped shape this work.

Special thanks go to my friends and family members that sacrificed their gaming nights so that I could use their GPUs.

Table of Contents

Introduction

In a natural language processing problem, such as document classification, it is common to encode words using one-hot vectors. Each word is represented by a vector the size of the vocabulary V, with all values equal to 0, except for a single 1 on the position that encodes the word. This approach results in wide and sparse word representations. Using word embeddings provides more information, as words similar by context have similar embeddings, while in the one-hot encoding the distance between all words is the same regardless of their similarity. For example, “ocean” and “sea” are similar words and would have similar embeddings, but with one-hot vectors this information is not made explicit.

In this work, we address the task of microbial phenotypes prediction from scientific papers. Since these papers consist of scientific terms and expressions, word embeddings trained on general-purpose libraries, such as tweets or news articles, may not be appropriate. One question we aimed to answer is whether it is better to use a larger, more general corpus, or a smaller, but more specific corpus for learning word embeddings. For this reason, custom word embeddings have been trained on publicly available scientific literature and its more subject-specific subsets related to biology and one particular branch thereof (here, microbiology). To compare which embedding is more suited for the problem at hand several methods for determining document embeddings from word embeddings have been tested. A simple, yet popular choice, is to aggregate word embeddings into a single document embedding. However, the problem with this approach is that it ignores word order. To capture this information, the hidden state of a single layer LSTM was used to represent a document.

The first chapter gives a quick overview of used word embeddings. The second chapter introduces recurrent neural networks. The third chapter explains how the corpora for training embeddings were prepared and the way the embedding size was determined. In the fourth chapter, the phenotypic trait classification dataset is presented alongside a baseline model. The fifth chapter introduces two ways to represent document embeddings, by aggregating word embeddings and by using a recurrent neural network. In the sixth chapter, the proposed models are compared to a baseline bag-of-words classifier. The seventh chapter explores if there is a connection between model performance and the corpus the embedding was trained on. The eight chapter explores if there is complementarity between the baseline and the proposed models. The ninth, and final chapter, brings the conclusion.

1. Word embeddings

Recent word embedding models [1] [2] represent words as dense N-dimensional vectors that capture linguistic regularities without using any external annotations. Not only are similar words grouped together, which is an improvement over one-hot vectors, but the embedding model also encodes complex relationships, as shown in Figure 1-1.

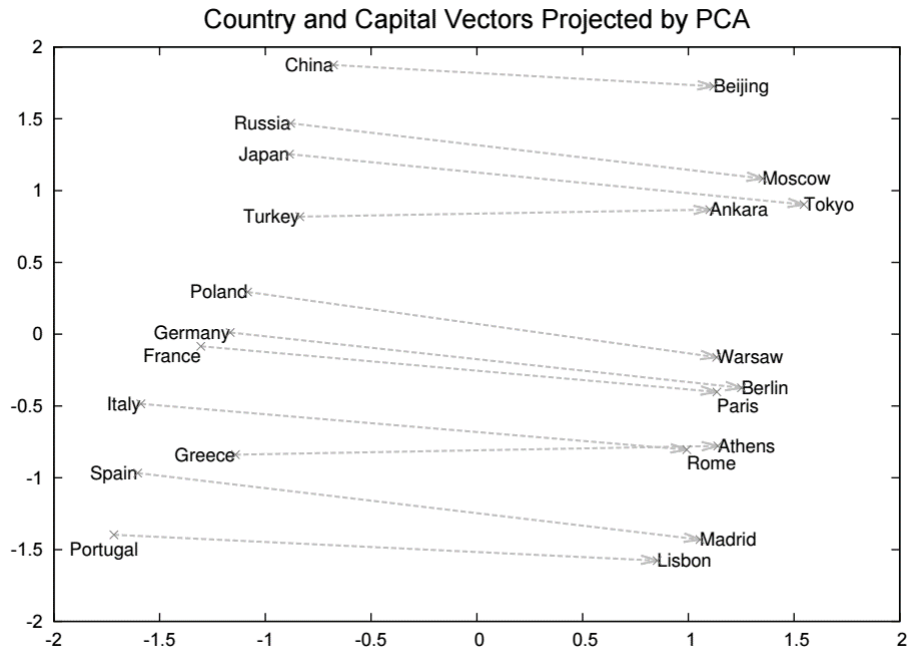

Figure 1-1 from [1] example of an embedding model learning the relationship between countries and their capital city

One of the most often examples used when talking about word embeddings is that the expression $embedding(\text{king}) - embedding(\text{man}) + embedding(female)$ results in a vector closest to $embedding(queen)$.

1.1. Word2vec

Word2vec is one of the most popular approaches to word embeddings offering high-quality word embeddings at a low computational cost [3] [1] [4].

It is a "shallow" model, having only a single hidden layer. The number of neurons in the input layer equals the number of words in the vocabulary. The hidden layer size is determined by the desired embedding vector size, and the output layer is the same size as the

input layer. Assuming we have V words in the vocabulary and desire an embedding size of N, input to hidden matrix W_I would have shape $V \times N$ and the output matrix W_O would have the inverted shape of $N \times V$. Input is encoded as a one-hot vector, where all the values are 0 except for one, encoding the word. Because of this, the hidden layer values will always be equal to one of the rows from W_I, the embedding of the input word, effectively working as a lookup table. Word2vec uses *softmax* to transform output values to have a sum of 1, so that they may be interpreted as probabilities, as the training is done by predicting connected words.

For example, let us say we want to learn the relationship between words *"kangaroo"* and *"jump"*. In that case when we use $onehot(\text{"kangaroo"})$ as input and expect a high probability for *"jump"*, our target word, in the output. Error vector is computed by subtracting the probability vector output from the one-hot vector of target word.

All weights are trained using stochastic gradient descent and backpropagation.

1.1.1. Continuous Bag of Words (CBOW)

The method described in the previous chapter is for learning relationships between a pair of words, in our example *"kangaroo"* and *"jump"*. Instead of using a single word to predict its pair, CBOW [3] uses the context of a word that is words in a neighborhood of the target (output) word. Context is taken without regard to word order, which is where the name bag-of-words comes from. For each target word, C words before and after it are used.

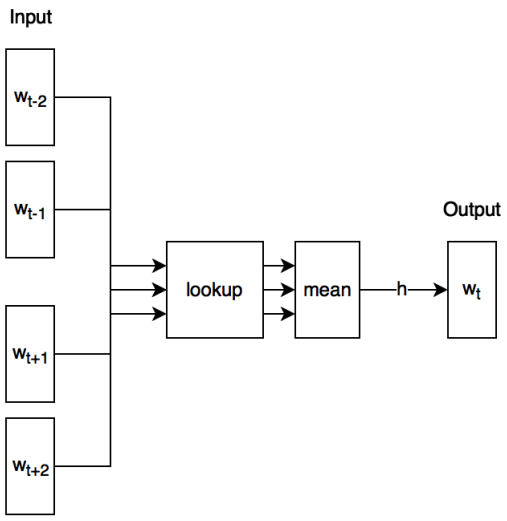

Figure 1-2: Continuous Bag of Words with context $C = 2$

Previously defined model is changed so that multiple inputs, each size of V, are projected using the same projection matrix W_I and then averaged, making the value of the hidden layer the average embedding of words defining the context.

$$h = \frac{1}{2C} W_I \sum_{-C \le c \le C, c \neq 0} w_{t+c}$$
$$w_t = W_O^T h \tag{1}$$

The rest of the method is unchanged.

1.1.2. Skip-Gram Model

Another approach to training word2vec is called Skip-Gram [3] [4]. A single input word is used to predict multiple target words (the exact opposite of CBOW). For each target word, a context is defined as the neighborhood of that word. Skip-gram tries to predict each context word from its the target word, effectively repeating the pairwise training for each context word. The training objective is to maximize the log probability

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-C \le c \le C, c \neq 0} \log p(w_{t+c} | w_t) \tag{2}$$

where $p(w_{t+c}|w_t)$ is defined with softmax, the output of neural network. Several methods have been developed to improve the computational cost of training and the quality of the word embedding, such as Hierarchical Softmax [1], Negative Sampling [1] [4] and Subsampling of Frequent Words [1].

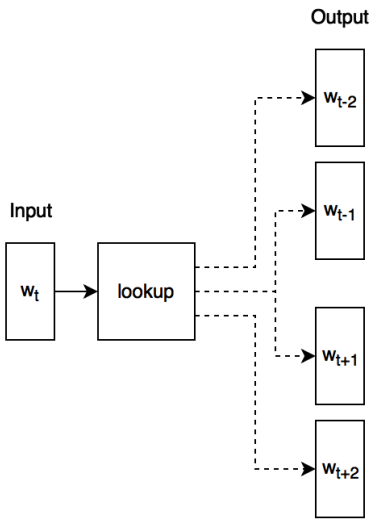

Figure 1-3: Skip-Gram model with context $C = 2$

1.2. GloVe

While Word2vec relies on a local context of a window to learn word embeddings, GloVe [2] directly uses the global word to word co-occurrence (name derived from "***Glo****bal* ***Vec****tors"*). Word co-occurrence is stored in the matrix X, where X_{ij} is the number of times word j appears in the context of word i. When learning word co-occurrence from the corpus with V distinct words a fixed number of words around the target word are taken as context. Then, for each word k in the context, the co-occurrence matrix gets updated relative to the distance of word k to the target word i in the context window.

$$X_{ik} = X_{ik} + \frac{1}{distance(i,k)} \tag{3}$$

This makes it so the closest words have the greatest impact in the later stages of training. The probability that word j will be found in the context of word i can be expressed as:

$$P_{ij} = \frac{X_{ij}}{\sum_{k=0}^{V} X_{ik}} \tag{4}$$

The main idea of this embedding method is that for two unrelated word i and j, and a word k related to i but unrelated to j, the ratio $\frac{P_{ik}}{P_{jk}}$ is high, as k appears in the context of i more often than in the context of j. From this idea, a constraint is defined for a pair of word vectors v_i and v_j [2]:

$$w_i^T w_j + b_i + b_j = \log(X_{ij}) \tag{5}$$

For the training of GloVe word embeddings, a weighted least squares error was proposed in [2]. By using a conventional least square error all word co-occurrence would be weighted equally and very frequent words, such as *the*, would affect the training the most.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \left(w_i^T w_j + b_i + b_j - log(X_{ij})\right)^2 \tag{6}$$

The weighting function f is defined by the authors as:

$$f = \begin{cases} \left(\frac{X_{ij}}{XMAX}\right)^{\alpha} & if\ X_{ij} > XMAX \\ 1 & otherwise \end{cases} \tag{7}$$

In the original paper, they used $XMAX = 100$ and $\alpha = \frac{3}{4}$, as those values have empirically been shown to work well.

2. Recurrent neural networks

Traditional feed-forward neural networks can have multiple vector inputs, as was shown with CBOW previously, but they accept only a fixed number of inputs, and do not take the order of the vectors into account. To be able to process varying length input vectors and capture the order information, a recurrent neural network (RNN) is used. RNNs have an additional recurrent connection in the hidden layer and can be viewed as a feed-forward network that remembers its hidden layer from the previous iteration and uses it to calculate new hidden layer values. This allows the output to be based not only on the current input but all past inputs.

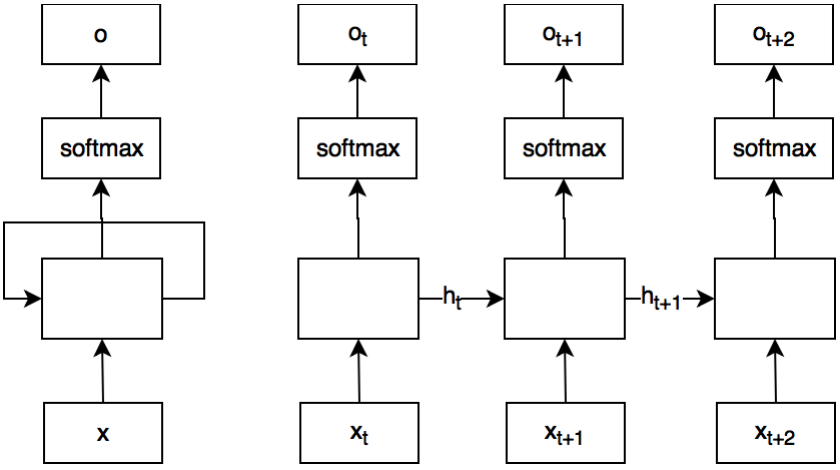

Figure 2-1: structure of RNN, before and after unrolling

A recurrent node has two inputs, the one fed to the network x_t and the hidden state of the previous iteration h_{t-1}. The hidden state is used to transfer previous information and is defined as

$$h_t = f(W_{hh}h_{t-1} + W_{ih}x_t + b_h) \tag{8}$$

where b_h is the hidden layer bias, W_{hh} is the hidden-to-hidden weights matrix and W_{ih} input-to-hidden weights matrix. Function f is a nonlinear, differentiable function, usually $tanh$. The output of the network is defined as

$$o_t = softmax(W_{ho}h_t + b_o) \tag{9}$$

with W_{ho} being the hidden-to-output matrix and b_o the output bias. The training goal is to minimize the loss function, commonly chosen to be cross-entropy loss. With y as the true label for iteration t and o and the output, cross-entropy loss is given by

$$L(y, o) = -\sum_i y_i \log o_i \tag{10}$$

Recurrent neural networks are trained using backpropagation through time (BPTT) which is the same as tradition backpropagation on an unraveled network. In theory, RNN could learn long-term connections, but because the gradient of the loss function decays exponentially with time, in practice it has a problem with capturing long-term connections.

2.1. Long Short-Term Memory

Long Short-Term Memory (LSTM) [5] [6] introduces more complex hidden state units consisting of memory cells and multiplicative update gates. The idea behind the architecture is to allow information to travel unchanged from past events, enabling the learning of long-term connections.

The information contained in the cell state is changed using three gate layers:

- Forget gate f_t
- Input gate i_t
- Output gate o_t

Each gate depends on the previous hidden state h_{t-1} and the current input x_t and has a specific purpose in controlling the information from and to the memory.

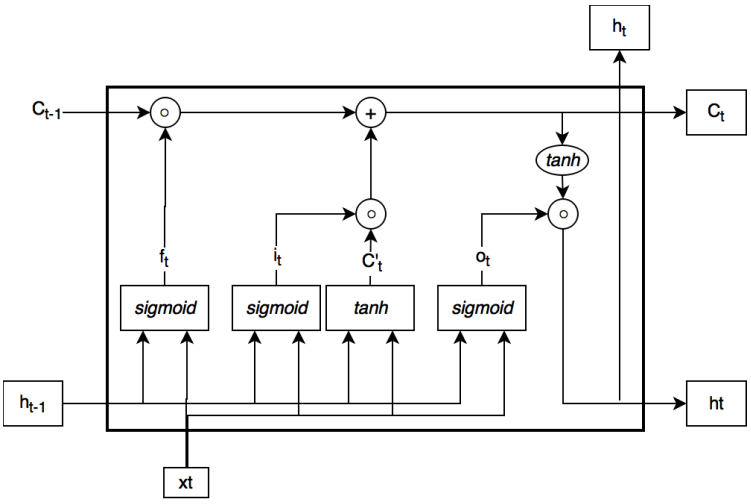

Figure 2-2 from [6] Structure of LSTM hidden unit neuron

The forget gate f_t determines what information should be removed from the memory cell C_t which is done with element-wise multiplication of the previous memory cell C_{t-1}. This produces a memory state with some values diminished, which will be annotated as $C_{t-1}^{(f)}$.

$$f_t = sigmoid\big(W_f\langle h_{t-1}|x_t\rangle + b_f\big) \tag{11}$$

$$C_{t-1}^{(f)} = f_t \circ C_{t-1} \tag{12}$$

New potential memory state C_t' is regulated with the input gate i_t so that the memory state $C_{t-1}^{(f)}$ is updated with a selected part of the information $C_t^{(i)}$.

$$i_t = sigmoid(W_i\langle h_{t-1}|x_t\rangle + b_i) \tag{13}$$

$$C_t' = \tanh(W_c\langle h_{t-1}|x_t\rangle + b_c) \tag{14}$$

$$C_t^{(i)} = i_t \circ C'_t \tag{15}$$

After removing part of the former information and adding new information $C_t^{(i)}$, value of the new memory state C_t is finally

$$C_t = C_{t-1}^{(f)} + C_t^{(i)} \tag{16}$$

Output gate determines which information from the memory cell should be used in the hidden state h_t. This is to prevent the influence of unwanted stored information.

$$o_t = sigmoid(W_o\langle h_{t-1}|x_t\rangle + b_o) \tag{17}$$

$$h_t = o_t \circ \tanh(C_t) \tag{18}$$

Unlike RNN, where information from past iterations travels by matrix multiplication, causing exploding, or vanishing gradients, in LSTM it travels linearly, allowing the network to learn long-term connections.

3. Training word embeddings

Both Word2vec [3] and GloVe [2] have the capacity to learn word embeddings with encoded semantic meaning from unlabeled data. For the task of classifying phenotypic traits, both Word2vec and GloVe were trained on the publicly available "biomedical and life sciences journal literature" repository PubMed Central (PMC) [7], obtained from their FTP site [8].

If not specifically mentioned, all code was written in Python [9], as it has all the required packages and bindings available, not only for the preprocessing but also for training Word2vec [10] and GloVe [11] embeddings.

3.1. Corpora preparation

The entire PMC corpus is separated alphabetically into four files and available in two formats. The original XML format, containing not only the article text but also all information about the article, such as formatting, chapters names, and tables. The other available option was a text file with the article text already extracted.

Although the already preprocessed text files were available, the abstract was missing, and there was no way to distinguish when the paper ended, and references began. For this reason, the text was extracted from the XML format.

Since Word2vec uses word context to create word-pairs which are used in training and GloVe uses it to generate the word co-occurrence matrix X_{ij}, all tables and the references were removed from the XML file. In tables, words are mostly used as headers and the context is determined by table cells, making those words unfit for training word embeddings. If the table does consist of words, their context is multi-dimensional and is hard to capture so, to not create false word context, tables were removed altogether. References were removed as author and paper names are numerous and the structure is predefined by the referencing standard, which is not relevant for the problem.

All other text was extracted from the XML format using the `beautifulsoup4` package in Python.

```
def nxml2txt(xml):
    soup = BeautifulSoup(xml, features="xml")
    body = soup.find('body')
    for table in body.find_all("table"):
        table.extract()
    for xref in body.find_all("xref"):
        xref.extract()
    pagetxt = [t.strip() for t in body.findAll(text=True)]
    return " ".join(pagetxt)
```

Code 1 Function for text extraction from XML format

The context of a word is interrupted with the ending of the sentence. To make certain that word context was restricted by the sentence, the text was split into sentences using the `sent_tokenize` function found in the `nltk` package [12]. For each sentence, all symbols were replaced with white spaces, except for dash "-" which was removed as it is mostly used to connect two relevant words, such as "Alpha-Lipoic Acid". All numbers have also been removed, and Unicode characters have been replaced with closest matching ASCII characters; 'ü' was replaced with 'u', 'α' with 'a' and so on. Finally, all English stop words (*the*, *and*, *through*, etc.), as defined in the `nltk` package, were removed, as they hold no relevant information for the problem of document classification.

Remaining words were stemmed using the `nltk` implementation of the Porter Stemming algorithm [13] because the volume of the training corpus was likely not enough to learn different tenses of the same word or relationships, such as plurality. The resulting word stem might not be a real word (*temptations* → *temptat*) as the algorithm uses predefined suffix rules, but it will appear in the same context as the original word, and that is sufficient for training word embeddings.

3.2. Determining subset corpora

More subject-specific subsets related to biology and microbiology were selected using MeSH (**Me**dical **S**ubject **H**eadings) categories [14]. Python package `biopython` [15] was used to for automated querying of PubMed Central (PMC) articles by MeSH terms. The query returned a list of PMC ids, which could be mapped to PMC articles. For comparing word embedding by specificity, several corpora were used:

- All – entire PMC article dataset – 1.5M articles
- Middle – subset of PMC articles related to biology – 374K articles
- Specific – subset of PMC articles related to microbiology – 76K articles

Each query was in the format:

```
open access[filter] AND („<term1>"[MeSH] OR „<term2>"[MeSH] OR ...)
```

A subset of approximately 76K PubMed Central articles from the field of microbiology, further referred as "Specific corpus", have been selected with the following MeSH terms:

- Archaea [B02]
- Bacteria [B03]
- Microbiological Phenomena [G06]
- Microbiology [H01.158.273.540]
- Bacterial Structures [A20]

Its superset, further referred as "Middle corpus", consisted of 374K articles and encompassed articles from the field of biology. Three additional MeSH terms were used:

- Biological Phenomena [G16]
- Biological Science Disciplines [H01.158]
- Cells [A11]

3.3. Determining embedding vector size

To determine embedding vector size, word embeddings quality was tested using top 20 ranked words in 113 manually curated groups obtained using non-negative matrix factorization (NMF) obtained from [16]. These words are co-occurring terms found in multiple microbiological corpora. Some words, such as *beer*, *lactic* and *spoilage*, were grouped together as lactic acids are responsible for spoiling beers, a connection that would not be present in corpora not related to biology. Although these words are not directly related, they do appear in similar articles, and they should be grouped closer together than random words.

The distance of word embeddings was calculated between word pairs within the group and between grouped words and random words. Mann-Whitney U-Statistic was calculated on the two distance distributions and transformed into an AUC score using the following formula:

$$auc = \frac{U}{N_1 N_2} \tag{19}$$

where U is the value of the U-statistic and N_1 and N_2 are the sizes of distributions.

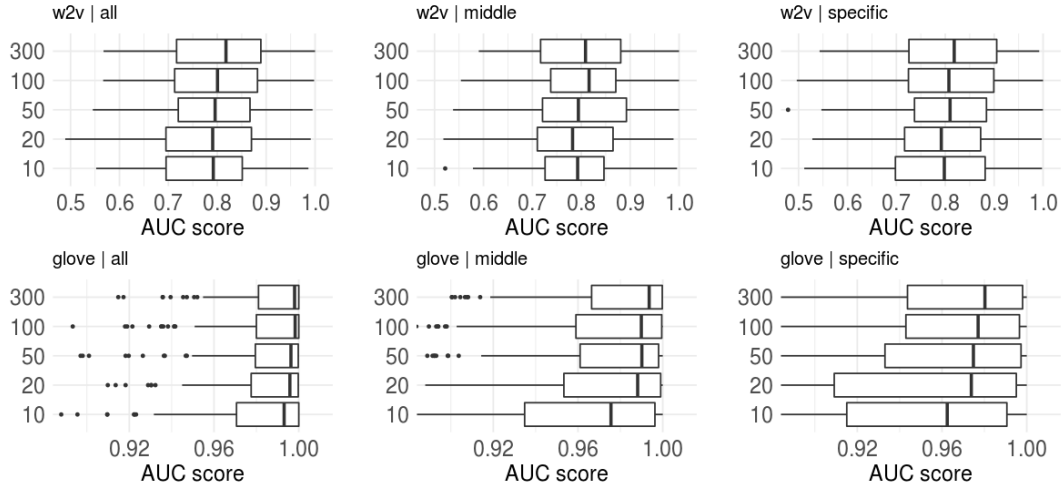

Figure 3-1 performance of different embedding sizes on the task of grouping similar topic terms

This test was run with the main corpus and its larger subsets, *Middle* and *Specific*, on different embedding sizes $e \in \{10, 20, 50, 100, 300\}$. GloVe performed significantly better than Word2vec, as can be seen on Figure 3-1 (note the different x-axis scale for Word2vec and GloVe). Furthermore, larger embedding sizes generally achieve better results. To reduce the parameter space that needed to be searched in further testing, it was decided to consider only embedding sizes $e \in \{100, 300\}$.

4. Phenotypic trait classification dataset

An existing dataset of labeled articles [16] was used. It consists of texts from six textual resources:

- Wikipedia
- Hamap
- MicrobeWiki
- PubMed Central publications
- PubMed abstracts
- mixed collection of smaller resources

Each text document maps to a taxonomy id (tax id) via its filename, formatted as `<name>__(<tax id>).txt`, where each tax id is associated with a single species. Set of phenotypes was gathered from the ProTraits database, which collected them from various sources [16], most notable for this work being the merged phenotypic traits from NCBI and BacMap. They are widely used and have the largest amount of tax ids associated with them.

Text processing was done in the same manner as in Section3.1, without the extraction from XML and splitting into sentences. Because phenotypic traits were matched with tax ids and not documents, all files associated with the same tax id were concatenated for each source.

4.1. Baseline model

In a bag-of-words representation, each document is encoded into a vector of length V, where V is the number of unique words in the corpus. Values are simply the number of times a word appears in the document. An improvement over this approach is to weight each word w in a document d by its frequency and rarity:

$$tfidf(w,d) = tf(w,d) * idf(w) \tag{20}$$

Frequency of the word in the document $tf(w,d)$ is multiplied by the inverse percentage of documents containing that word $idf(w)$, also known as inverse-document-frequency (tf-idf).

All documents were encoded using a bag-of-words approach with *tf-idf* weighting. A binary linear support vector machine (SVM) classifier, from the `sklearn` [17] package, was

trained for each *NCBI+BacMap* phenotypic trait and text corpus. Phenotypic traits with less than 10 positive and negative labels were deemed as untrainable and removed from the dataset. The model score was defined as the average AUC score on 5 stratified folds. Random guessing would have AUC score of 0.5. The regularization parameter $C \in \{2^{-15}, 2^{-14}, \ldots, 2^{4}, 2^{5}\}$ was optimized using stratified 5-fold cross-validation (on the remaining 4/5 folds of the first split), keeping the value with the best AUC score. Figure 4-1 shows the AUC scores of the baseline BoW model.

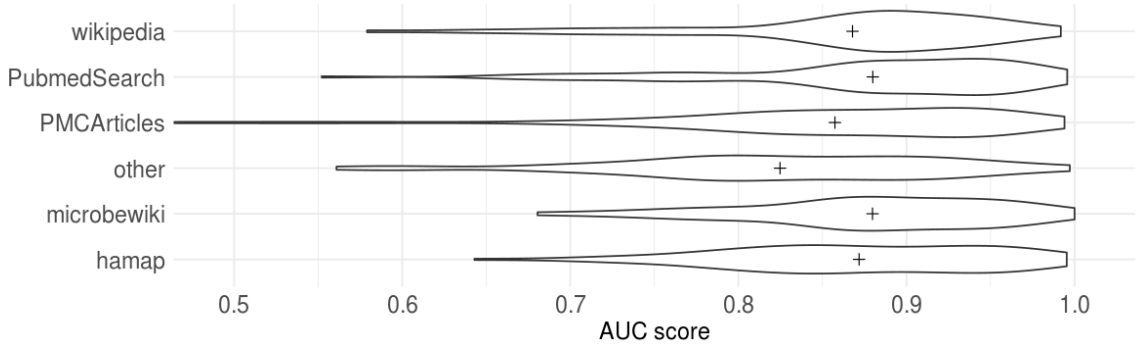

Figure 4-1 AUC scores of the baseline BoW model with marked medians

As it can be seen from Figure 4-1, *other*, the only dataset comprised of multiple sources, is significantly less accurate (with $pvalue < 0.01$ using a paired t-test) than all other dataset except for *PMCArticles*.

5. Document embedding

Both Word2vec and GloVe are methods for word embeddings, but for the problem of document classification, a single vector encoding a document is needed. We explored three methods, two based on aggregating word embeddings and one based on recurrent neural networks.

5.1. Aggregation of word embeddings

Word embeddings are generated in a way that tries to preserve the word meanings in N-dimensional space. The simplest way to get a document embedding from word embeddings is to sum all the word embeddings into a single N-dimensional vector. While this approach does not take word order into account, it does consider similar words, as they should have similar embeddings.

5.1.1. Aggregation function

Instead of summing all word vectors, several functions were tested to try to determine a possibly better function for combining information from multiple word embeddings. This was done by using Word2vec and GloVe embeddings, trained on the entire PubMed Central corpus with embedding size $e = 100$, to classify 60 representative phenotype traits. For each document, a single embedding was calculated from all its word embeddings using one of the tested functions: *.05 percentile*, *.95 percentile*, *min*, *max*, *sum*, as well as *min*, *max* and *sum* applied to normalized word embeddings and a concatenation of *min*, *max* and *sum* (further referred as *MinMaxSum*), both on regular and L2 normalized word embeddings.

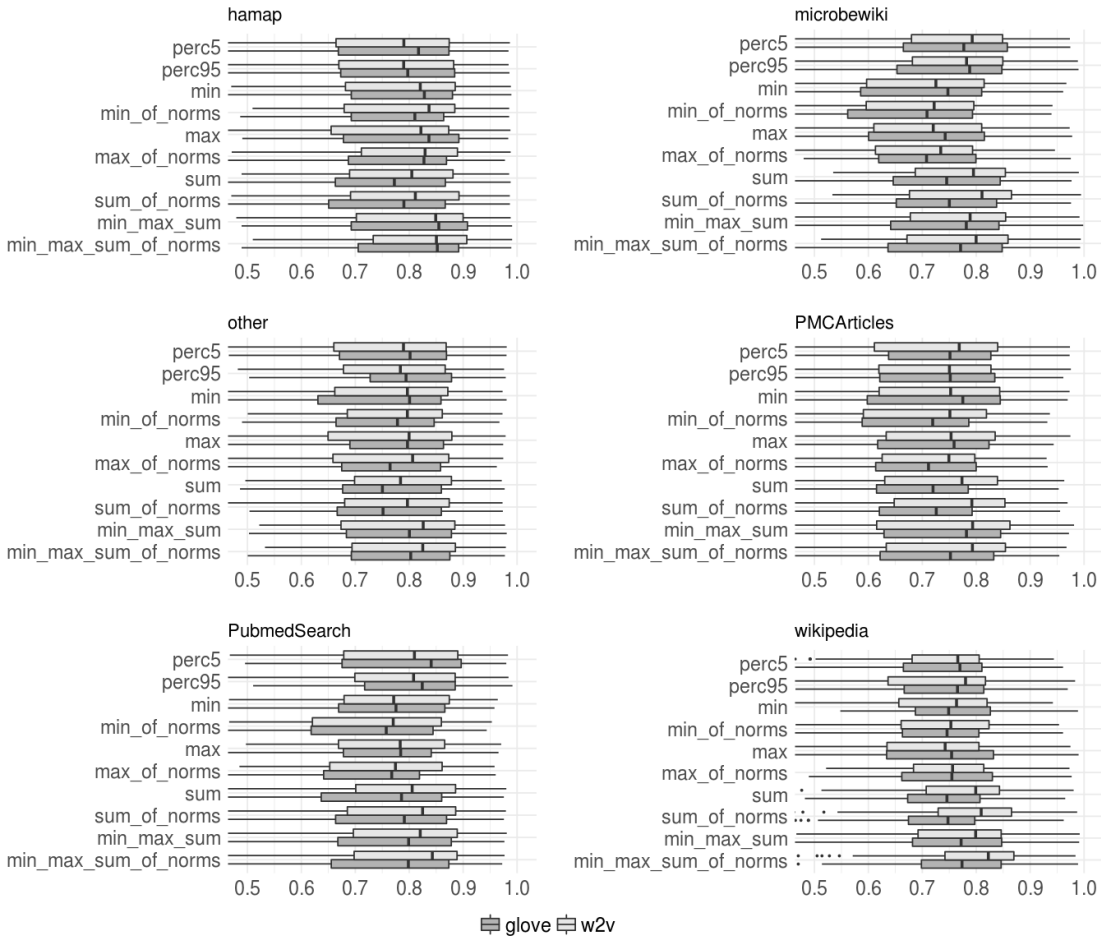

Figure 5-1 AUC scores while classifying 60 phenotypic traits using different functions with embeddings trained on the entire PMC corpus

All tested functions had similar scores, but *MinMaxSum* showed better results than others. Using normalized vectors with this function improved the AUC score of Word2vec embeddings in two corpora, *PubmedSearch* and *Wikipedia*, but had a negative effect on GloVe embeddings in the *PMCArticles* corpus. Despite this, *MinMaxSum* applied on normalized vectors was chosen as the aggregating function.

5.1.2. Weighting word embeddings

Frequent words are given vectors with large L2 norms by most embedding methods, which is a problem while aggregating word embeddings. Frequent words (such as *the*, *and* or *they*) do not have high information value but have the greatest impact. This is true even while using normalized embeddings, as they greatly outnumber other words. One way to overcome this issue was to remove English stop-words, but as the corpora were comprised of scientific texts, frequent scientific words remained.

Word embeddings were weighted using *smooth inverse frequency* (SIF) [18]:

$$sif(w) = \frac{a}{a + p(w)} \tag{21}$$

where a is a small constant (10^{-3} was used) and $p(w)$ is the word frequency estimated from the corpus. For the sake of comparing corpora specificity, word frequencies were estimated from the same corpus used to train the word embeddings.

Embedding of document D was then calculated as:

$$d' = \frac{1}{|D|} \sum_{w \in D} sif(w)\ emb(w) \tag{22}$$

Following [18], the projection of the document embeddings on their first principal component u was removed:

$$d = d' - u\, u^T\, d' \tag{23}$$

The reasoning behind removing the projection on the first principal component was that vectors have huge components along semantically meaningless directions as a by-product of embedding training.

5.2. Document embedding using LSTM

The memory block in LSTM can be used to store information passed from the previous word. The hidden state of an LSTM, after passing all the words from the document, was considered as the documents embedding and used to classify microbial phenotypic traits. In this way, not only were dense word embeddings used, but also the order of words in the document was considered.

5.2.1. Architecture

The model consisted of an input layer, one LSTM hidden layer, and output layer. The network was trained on the binary classification task differentiating documents describing species according to phenotype trait presence or absence. A different model was trained for each phenotypic trait.

All word embeddings connected to one document were considered a single input and sequentially fed to the network. The final hidden state (document embedding d) was then passed to the output layer with softmax activation, as shown on Figure 5-2.

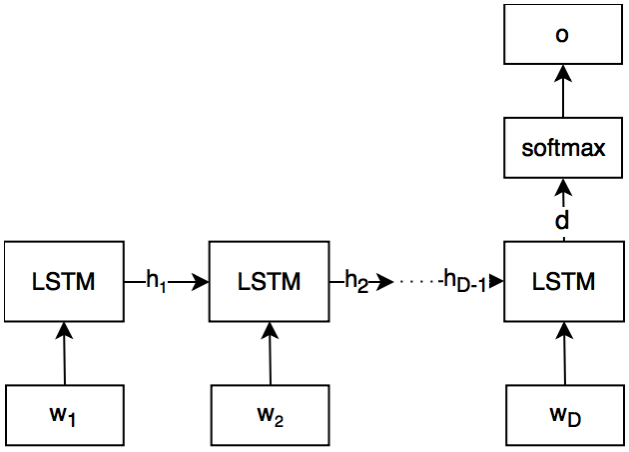

Figure 5-2 Architecture used to predict phenotypic traits using word embeddings

Dropout was only applied on non-recursive connections [19]. Training was done using *RMSprop* [20], a gradient descent optimizer that utilizes the moving average of squared gradients g for each weight

$$r_t = \rho r_{t-1} + (1-\rho) g * g \tag{24}$$

The moving average r and gradient g are then used to update parameters θ

$$\theta_t = \theta_{t-1} - \frac{\varepsilon}{\sqrt{\delta + r}} * g \tag{25}$$

where ε is the learning rate and δ the decay rate. As the task was binary classification, binary cross entropy was used as the loss function L:

$$L(\boldsymbol{o}, \boldsymbol{y}) = -\frac{1}{N}\sum_{i=1}^{N} (y_i * \log o_i + (1-y_i) * \log(1-o_i)) \tag{26}$$

The number of epochs was set to 100, with early stopping if the loss function, applied to the validation set, showed no improvement in 6 epochs.

5.2.2. Hyperparameters

As grid-searching LSTMs is time-consuming, only a few hyperparameters were tested. While experimenting on the phenotypic trait "*TemperatureRange=thermophilic"*, parameters: dropout $\in \{0.5, 0.7\}$, LSTM hidden layer size $H \in \{100, 300\}$ and learning rate $\varepsilon \in \{0.00075, 0.001, 0.0025\}$ have shown good results. The best parameters for a given embedding and corpus specificity were determined by the best average AUC test score on stratified 5-fold split using three phenotypic traits that were chosen based on different ratio of positive examples, as well as different quality results using the baseline model:

- TemperatureRange=thermophilic
    - 16.87% positive examples and baseline AUC score of 0.97
- shape=bacilli
    - 71.07% positive examples and baseline AUC score of 0.85
- metabolism=cellulosedegrader
    - 9.49% positive examples and baseline AUC score of 0.99

All models were defined using the *Keras* [21] framework with *Tensorflow* [22] backend. Because of the way CUDA GPU parallelization works, lists of word embeddings had to be zero-padded to the length of the largest document, which did not affect the result but meant that having one large text in the batch increased the memory requirements drastically.

Using a Nvidia GTX 1060, it took approximately 3 days to find the best hyperparameters for a given embedding and dataset. This was much faster when the model had poor predictive performance, because of early stopping after 6 epochs without improvement of the loss function applied on the validation set.

Best results in hyperparameter optimization for *hamap*, *MicrobeWiki*, and *PMCArticles* datasets are shown in Table 1, where P_1 = *TemperatureRange=thermophilic*, P_2 = *shape=bacilli* and P_3 = *metabolism=cellulosedegrader*.

Table 1 AUC scores for three phenotypic traits using the best hyperparameters obtained for each embedding separately

		Train			**Test**			
embedding	**dataset**	P_1	P_2	P_3	P_1	P_2	P_3	$\frac{1}{3}\sum_{i=1}^{3} P_{i,test}$
Word2vec entire PMC	*hamap*	0.99	0.91	0.99	0.95	0.59	0.91	0.82
Word2vec *Middle*	*hamap*	0.99	0.82	0.99	0.95	0.54	0.94	0.81
Word2vec *Specific*	*hamap*	0.99	0.90	0.99	0.95	0.61	0.92	0.83
GloVe entire PMC	*hamap*	0.90	0.79	0.91	0.80	0.62	0.76	0.73
GloVe *Middle*	*hamap*	0.86	0.74	0.88	0.73	0.60	0.74	0.69
Glove *Specific*	*hamap*	0.87	0.74	0.84	0.76	0.62	0.66	0.68
Word2vec entire PMC	*MicrobeWiki*	0.99	0.98	1	0.84	0.48	0.32	0.54
Word2vec *Middle*	*MicrobeWiki*	0.99	0.98	1	0.87	0.48	0.35	0.57
Word2vec *Specific*	*MicrobeWiki*	0.99	0.99	1	0.77	0.54	0.53	0.61
Word2vec entire PMC	*PMCArticles*	0.98	0.91	0.97	0.76	0.58	0.53	0.62
Word2vec *Middle*	*PMCArticles*	0.96	0.87	0.97	0.77	0.58	0.55	0.63
Word2vec *Specific*	*PMCArticles*	0.96	0.86	0.97	0.75	0.57	0.58	0.63

6. Performance and comparison with baseline

As the *NCBI+BacMap* phenotypic traits had the largest number of labeled documents, they were used to compare different capabilities and characteristics of our three models. Phenotypic traits with less than 10 positive and negative labels have been deemed as untrainable and were not considered further.

The baseline BoW + *SVM* model described in 4.1 was compared to two aggregation methods (*MinMaxSum* of norms and Weighted embedding approach) with RF on all phenotype document datasets using embedding size $e \in \{100, 300\}$. The LSTM model was compared to the baseline only using the *hamap* dataset, because of very high computational costs, as well as poor results on the three phenotypic traits used for finding optimal hyperparameters when using other datasets (Table 1).

6.1. Aggregation methods

Since the approach using aggregation functions generated a document embedding, and not the final prediction, both a support vector machine (SVM) and Random Forest (RF) [23] were trained to predict phenotypic traits from that embedding.

It was shown that RF with 500 decision trees behaved very similar to a SVM model (considering the best model among a linear SVM with $C \in \{2^{-5}, 2^{-4}, \ldots, 2^{14}, 2^{15}\}$ and a Radial Basis Function (RBF) SVM with regularization parameter $C \in \{2^{-5}, 2^{-4}, \ldots, 2^{14}, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-14}, \ldots, 2^{4}, 2^{5}\}$). All parameters were optimized using stratified 5-fold cross-validation, keeping the value with the best AUC score. To minimize computational time, RF was used in all classification tasks.

6.1.1. MinMaxSum of norms

14 out of 18 comparisons (in both Word2vec and GloVe) showed significantly ($pvalue < 0.01$ using a paired t-test) higher predictive accuracy while using embedding size of 300 over 100, as can be seen in Figure 6-1 and Figure 6-2. The baseline BoW model was significantly ($pvalue < 10^{-4}$ using a paired t-test) better than any model based on the aggregation function *MinMaxSum* of norms, regardless of the embedding.

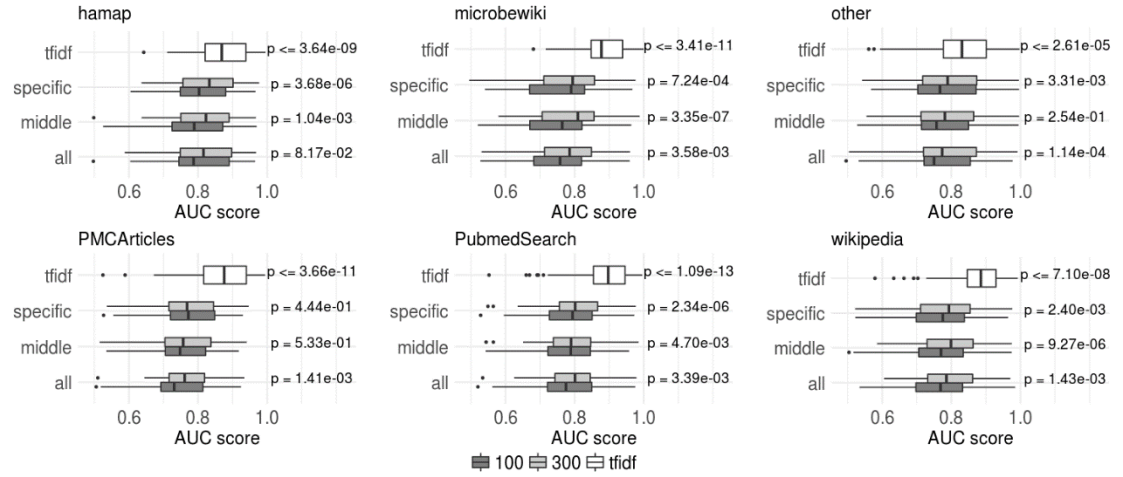

Figure 6-1 AUC scores for *MinMaxSum* of norms using Word2vec embedding and RF classifier, p-value (using a paired t-test) when comparing models with different embedding sizes is next to the paired boxplot, the baseline p-value (topmost one) is the maximal p-value (using a paired t-test) when comparing the baseline Bow model with all other models

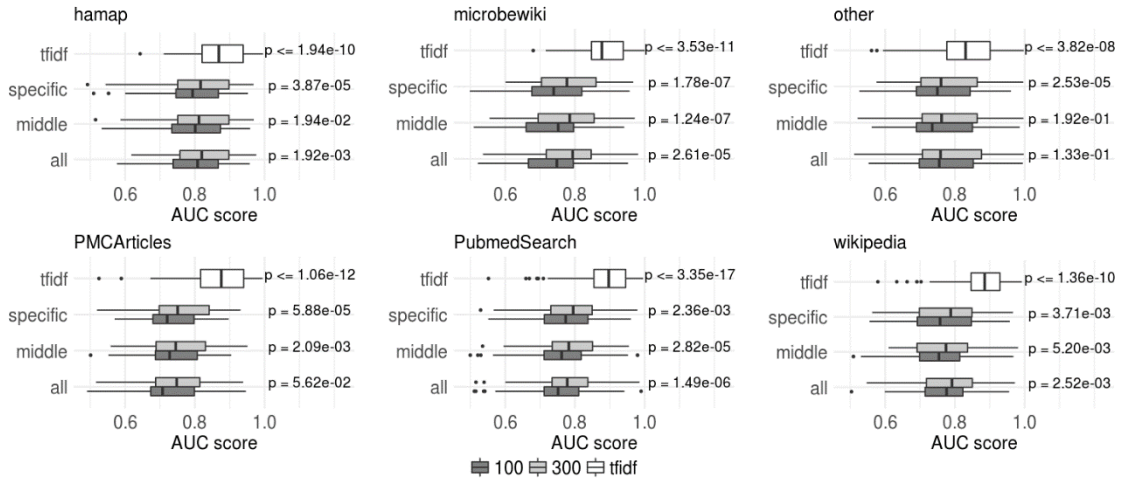

Figure 6-2 AUC scores for *MinMaxSum* of norms using GloVe embedding and RF classifier, in the same format as Figure 6-1

6.1.2. Weighted embeddings

Using embedding size 100 or 300 mostly did not exhibit a significant difference (4 of 18 comparisons) with Word2vec (Figure 6-3). On the other hand, GloVe embedding with size 300 was significantly ($pvalue < 0.01$ using a paired t-test) better then with size 100 in 15 of 18 comparisons (Figure 6-4). The baseline model was significantly ($pvalue < 10^{-11}$ using a paired t-test) better than any model based on the proposed weighted aggregation approach.

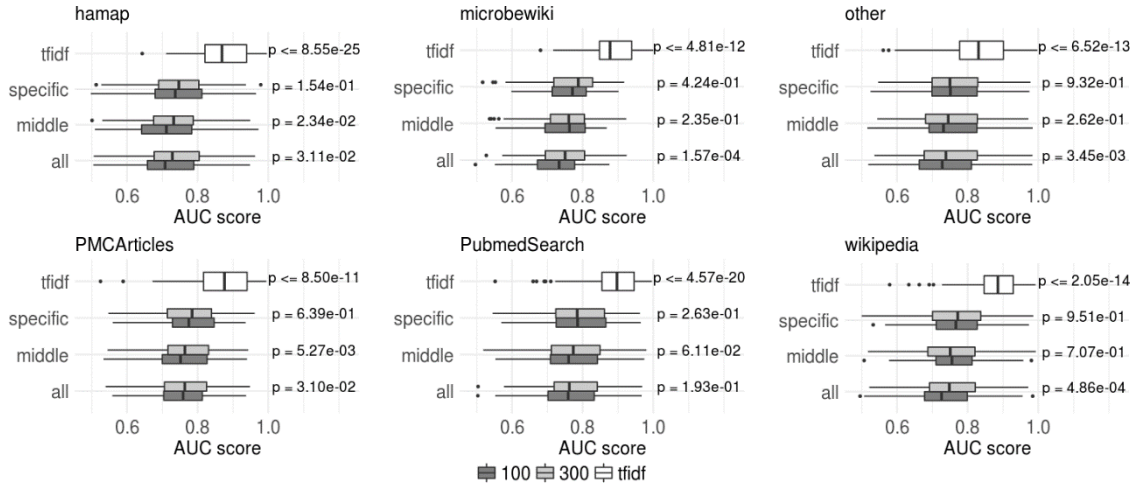

Figure 6-3 AUC scores for the weighted aggregation using Word2vec embedding and RF classifier, in the same format as Figure 6-1

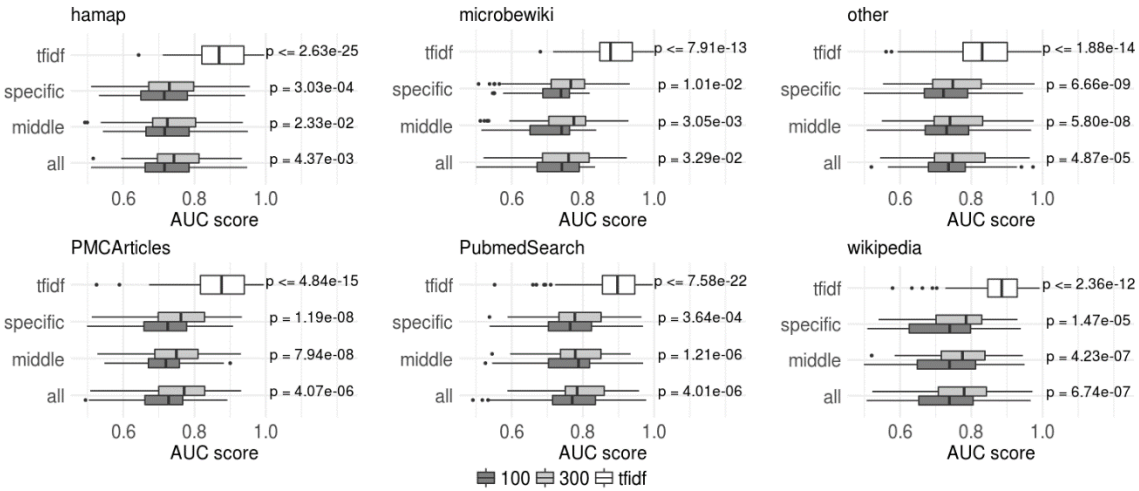

Figure 6-4 AUC scores for the weighted aggregation using GloVe embedding and RF classifier, in the same format as Figure 6-1

6.2. LSTM method

LSTM model had a classifier built into the architecture, so there was no need to use RF or SVM. As a separate model needed to be trained for each phenotypic trait, the training process was time-consuming. Using a Nvidia GTX 1060, it took approximately 7 days to get AUC scores for *NCBI+BacMap* phenotypic traits.

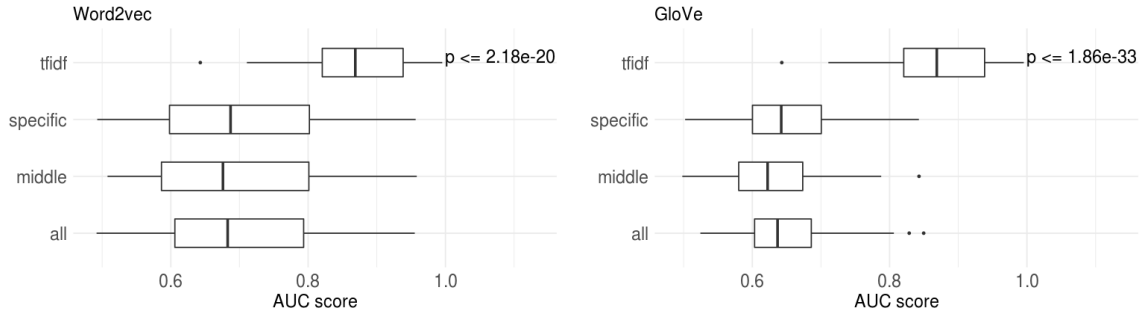

Figure 6-5 AUC scores for LSTM on the *hamap* dataset

Although this model takes both similar words and word order into account, it performed worse than all other models. GloVe embeddings showed consistently lower accuracy than Word2vec (average AUC=0.63 and 0.69 for Glove and W2V, respectively).

Whether it was because the documents were too long, or the training set too small (around 1000 documents per phenotypic trait), the LSTM model tended to overfit.

7. Influence of the specificity of corpora

Is it better to use a larger, more general corpus, or a smaller, but more specific corpus for learning word embeddings? Although the proposed models were not better than the BoW baseline, the influence of the size and specificity of the embedding corpora could still be analyzed.

To check if the AUC scores showed a significant trend with using different specificity corpora (entire PMC, *Middle*, *Specific*) to train the embeddings, *permTREND* function from the R [24] package *perm* [25] was used. In short, this method encoded the embedding corpora: *Specific corpus* became 1, *Middle corpus* 2, and the entire PMC corpus 3. Pearson correlation between corpora index and AUC scores was calculated. From this, the p-value was found empirically, through the randomization of corpora indexes. Indexes were permuted multiple times, and the new correlation coefficient was calculated; the p-value would be low if the initial correlation coefficient was unlikely to be observed in randomized data.

7.1. Corpus specificity test on NCBI+BacMap phenotypic traits

The same phenotypic traits used to compare the proposed models with the baseline in the previous chapter were used to test if there is a correlation between embedding corpus size and model quality.

7.1.1. MinMaxSum of norms

The observed trend using *MinMaxSum* of norms was not significant using Word2vec (Figure 7-1) or GloVe (Figure 7-2) on any dataset. It is important to note that even though the *Specific* corpus consists of 4.9% of the entire PMC articles, it had very similar (or even slightly better) results.

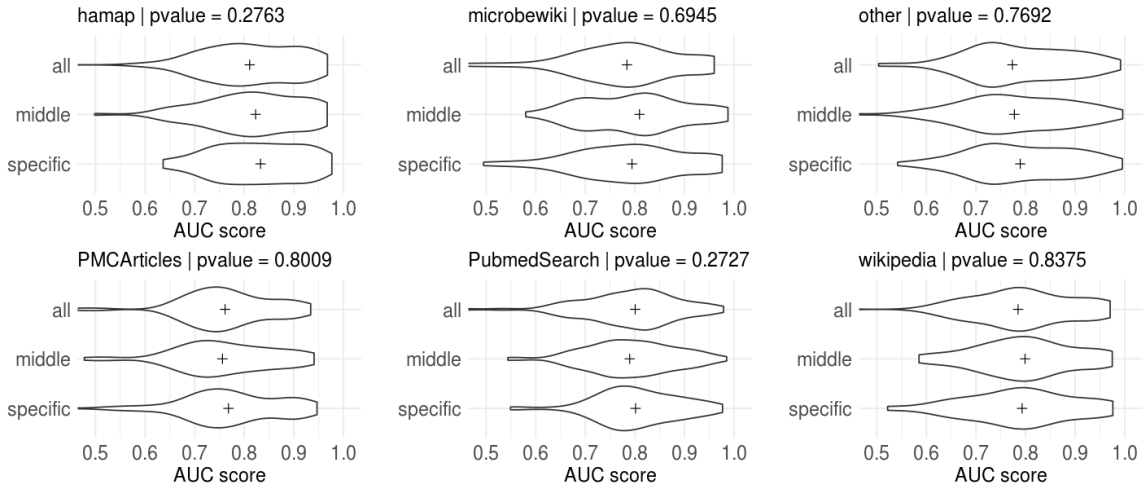

Figure 7-1 AUC scores for *MinMaxSum* of norms using Word2vec embedding with size 300 and RF classifier; p-values are from the permTREND function

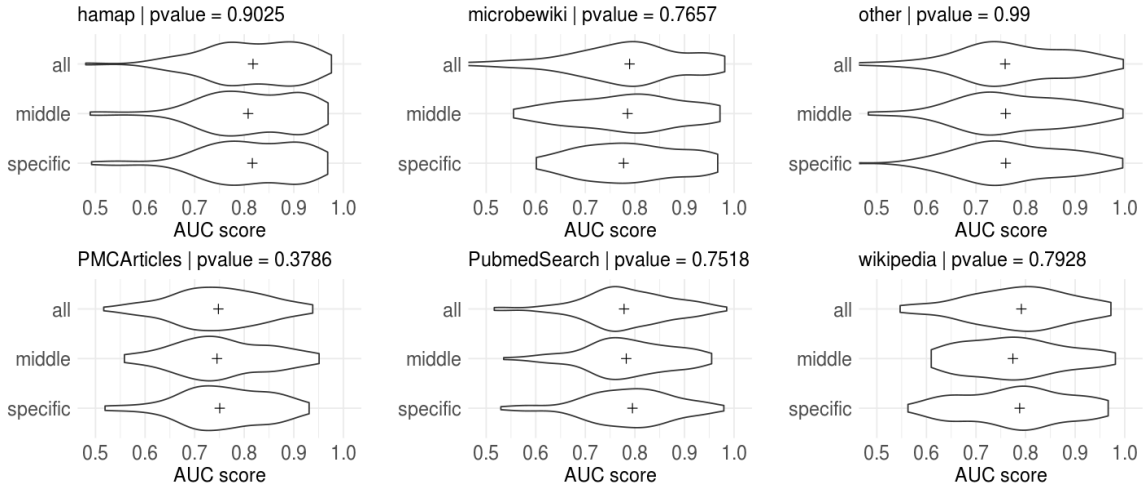

Figure 7-2 AUC scores for *MinMaxSum* of norms using GloVe embedding with size 300 and RF classifier; p-values are from the permTREND function

7.1.2. Weighted embeddings

Weighting word embeddings while using Word2vec embedding showed a visible (not significant) trend in favor of smaller, specific datasets, as shown in Figure 7-3. One might argue that this was because word probabilities were also estimated from the same corpus the embeddings were trained on, but using GloVe embeddings did not reveal the same trend (Figure 7-4).

When comparing Word2vec trained on the entire PMC corpus and on the *Specific* corpus, 4 of 6 datasets showed significantly ($pvalue < 10^{-3}$ using a paired t-test) better predictive accuracy with the *Specific corpus*.

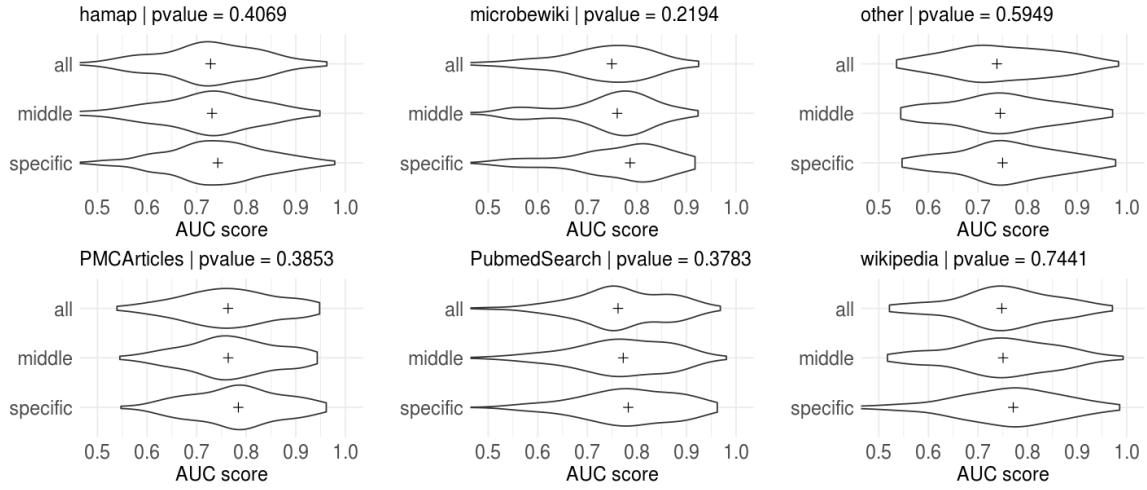

Figure 7-3 AUC scores for the weighted aggregation using Word2vec embedding with size 300 and RF classifier; p-values are from the permTREND function

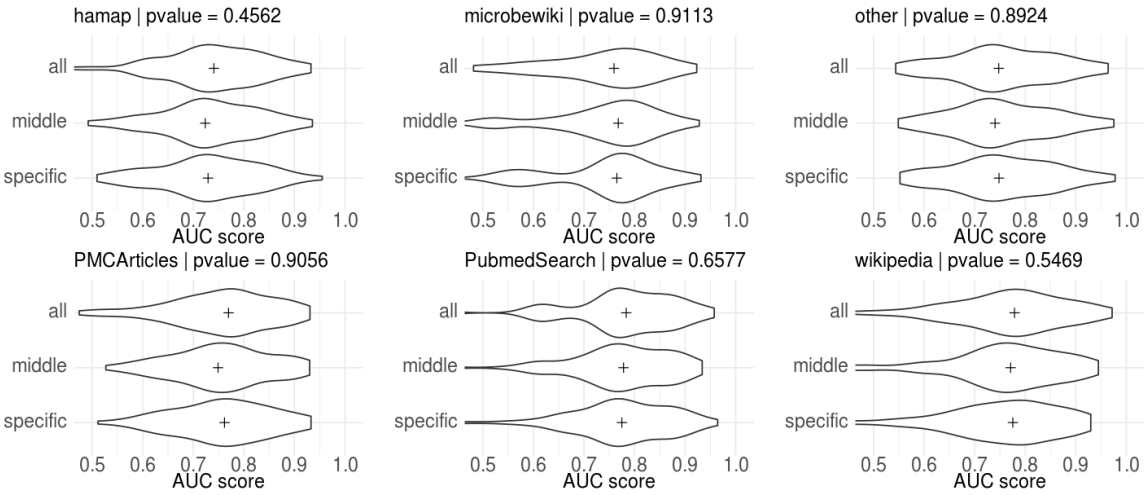

Figure 7-4 AUC scores for the weighted aggregation using GloVe embedding with size 300 and RF classifier; p-values are from the permTREND function

7.1.3. Simulation experiments to determine the influence of corpus size and specificity on predictive accuracy

While the trend was not significant, the fact that *Specific* embeddings showed mostly same (or even better) results than the entire PMC corpus invited further testing. We investigated the predictive power of the word embeddings if they had all been generated from corpora of the same size, but different subject specificity. To test this, the entire corpus (1.5M) and its *Middle* subset (374K) were randomly sampled to the size of the *Specific* subset (76K).

The same trend shown while using weighted Word2vec embeddings (Figure 7-3) was even more pronounced (Figure 7-7) with some datasets now showing a significant ($pvalue < 0.05$) trend. The same trend could now also be seen while using *MinMaxSum* of norms (Figure 7-5). GloVe embeddings showed no visible trend related to corpora specificity when using *MinMaxSum* of norms (Figure 7-6).

With all corpora the same size, using weighted embeddings showed a trend towards higher predictive accuracy both Word2vec (Figure 7-7) and GloVe (Figure 7-8). This was most likely because the *Specific* corpora had a better approximation of word probability than other, subsampled, corpora.

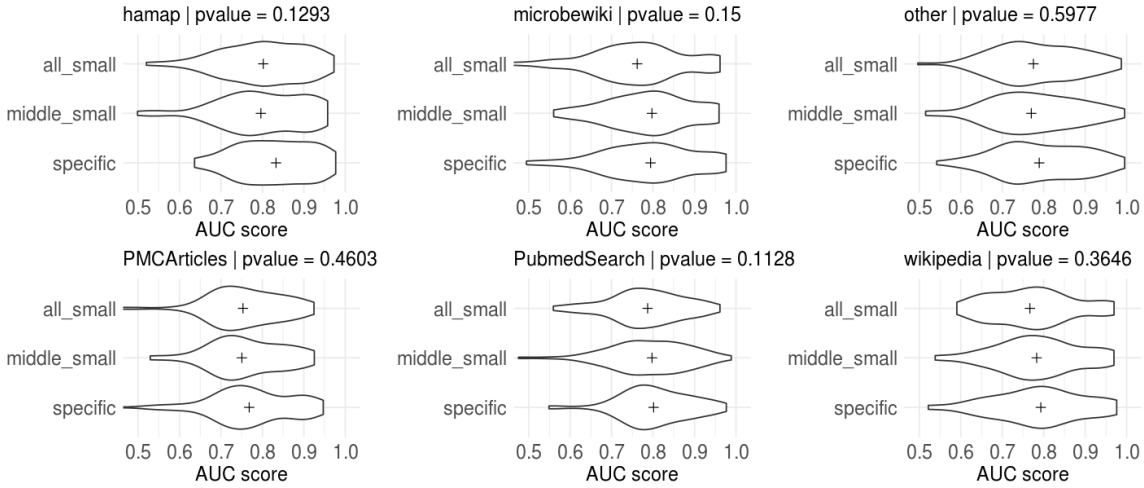

Figure 7-5 AUC scores for *MinMaxSum* of norms using Word2vec embedding with size 300 trained on the same sized corpora; p-values are from the permTREND function

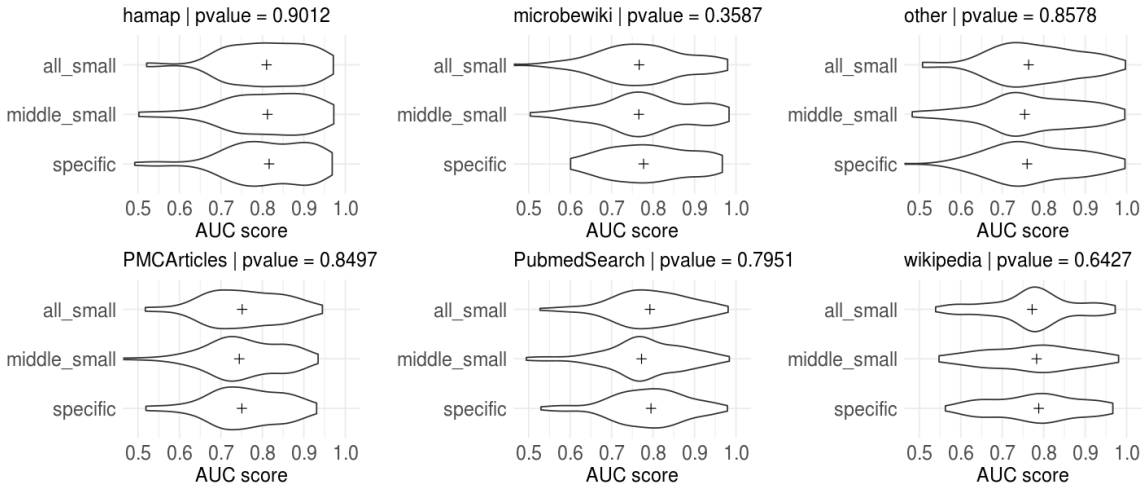

Figure 7-6 AUC scores for *MinMaxSum* of norms using GloVe embedding with size 300 trained on the same sized corpora; p-values are from the permTREND function

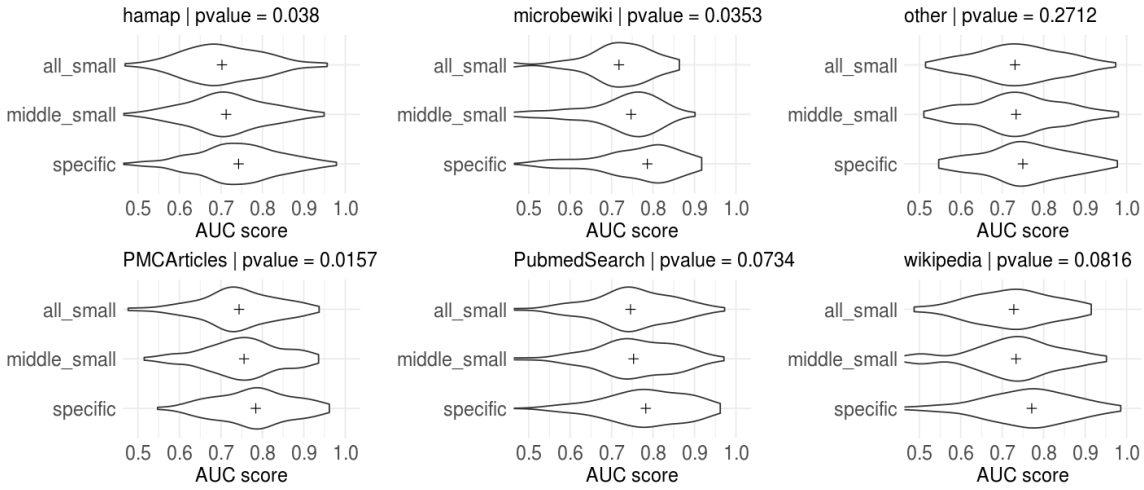

Figure 7-7 AUC scores for the weighted aggregation using Word2vec embedding with size 300 trained on the same sized corpora; p-values are from the permTREND function

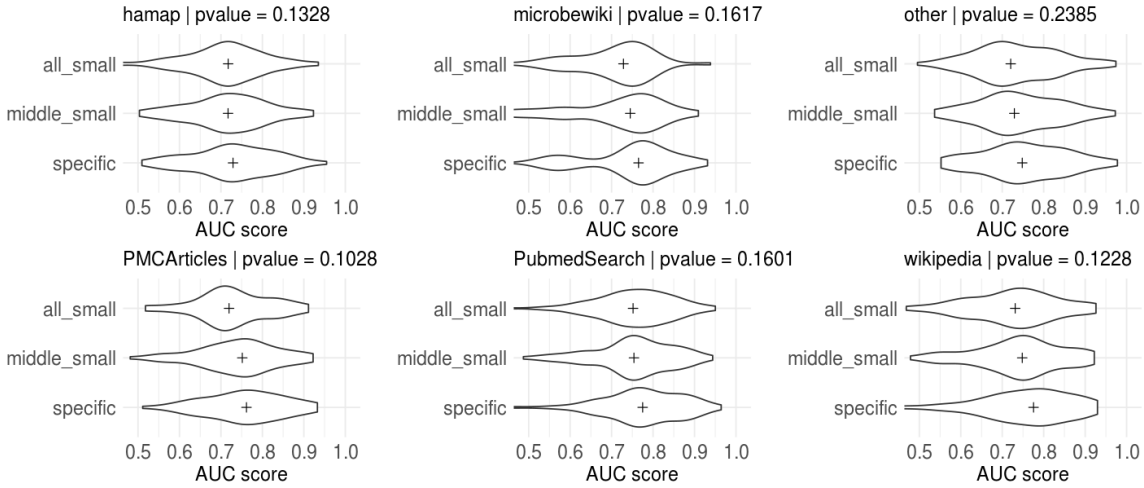

Figure 7-8 AUC scores for the weighted aggregation using GloVe embedding with size 300 trained the same sized corpora; p-values are from the permTREND function

7.2. Corpus specificity test on highly specific phenotypic traits

In addition to corpora related to scientific fields of biology and microbiology, four corpora related to highly specific phenotypes (further referred as "Very Specific [A, B, C, D] corpus") were selected using the following MeSH terms:

a. Sporulation (9.4K documents)
    - Spores [B05.775], [A11.870]
    - Endospore-Forming Bacteria [B03.300]
b. Pathogenic in mammals (33.3K documents)
    - Host-Pathogen Interactions [G06.590.470]
    - Bacterial Infections [C01.252]
    - Blood-Borne Pathogens [B03.165]
c. Nitrogen fixation (2.3K documents)
    - Symbiosis [G06.590.580.800], [G16.100.900]
    - Endophytes [B05.237]
    - Rhizosphere [G16.500.275.157.625], [G16.500.853], [N06.230.124.437]
    - Nitrogen Fixation [G06.590.620], [G06.590.110.610], [G06.099.112.610]
    - Nitrogen Cycle [G16.500.240.465]
d. Cell Respiration (744 documents)
    - Bacteria, Aerobic [B03.120]
    - Cell Respiration [G04.299.305]
    - Anaerobiosis [G03.495.146]
    - Aerobiosis [G03.495.112]

These corpora consisted of few documents and targeted a specific set of phenotypic traits. *Very Specific A* was targeted at a single phenotypic trait (*sporulation*), *Very Specific B*, the largest very specific corpus, was targeted at 9 phenotype traits, *Very Specific C* was targeted at 5 and *Very Specific D* at 2 phenotypic traits.

We wanted to see if embeddings trained on very specific corpora would perform better on the 17 very specific phenotypic traits than the three larger corpora and if the trend favoring specific corpora, visible while using the weighted aggregation approach, would persist.

7.2.1. Results

Since the weighted aggregation approach with Word2vec embedding showed a trend favoring specific corpora with NCBI+BacMap phenotype traits, we further tested this approach with very specific phenotypic traits.

Very Specific models were less accurate than *Specific*, meaning that the minimum corpus size needed to incorporate semantics into embeddings, and to approximate word frequencies, might have been reached. The specificity trend was quite visible on the *MicrobeWiki* dataset while using Word2vec (Figure 7-9).

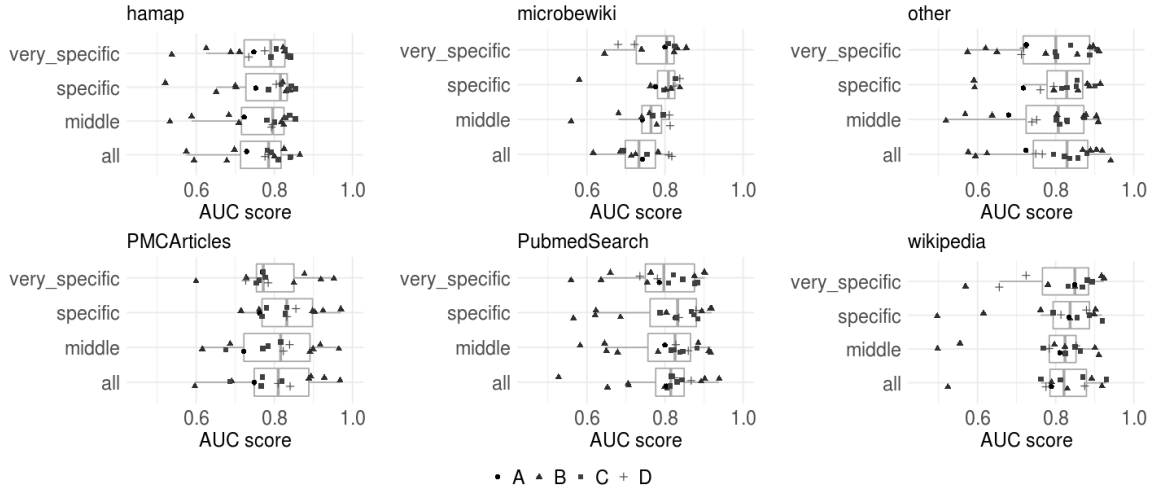

Figure 7-9 AUC scores for the weighted aggregation using Word2vec embedding with size 300, the appropriate *Very Specific* model was used for each label

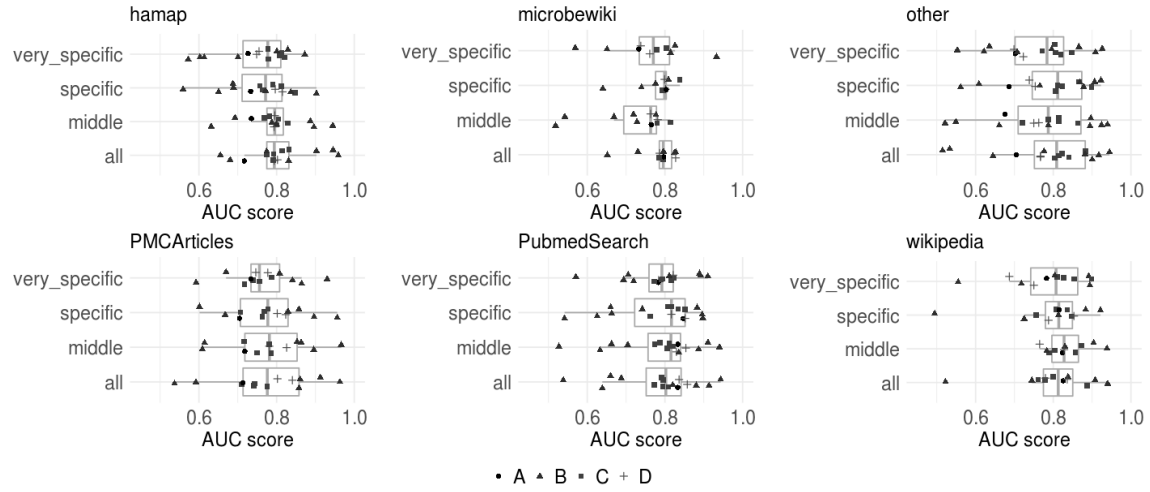

Figure 7-10 AUC scores for the weighted aggregation using GloVe embedding with size 300, the appropriate *Very Specific* model was used for each label

8. Model complementarity

Even if a model was generally less accurate, it could still be confident in some predictions where the more accurate model was not. Given that different methods were used to get the final document representation, it was possible that models work better on different documents.

To measure which documents the model was certain about, its confidence level (output) was converted to a precision score by thresholding the precision-recall curves obtained via cross-validation. Simply put, each confidence score given by the model was used as a threshold (instead of the typical 0.5 threshold), from which the precision was calculated as:

$$precision = \frac{TP}{TP + FP} \tag{27}$$

This was done for the minority class, most often being '+' (presence of a phenotypic trait). Predictions with $precision \geq 0.8$ were considered as confident predictions.

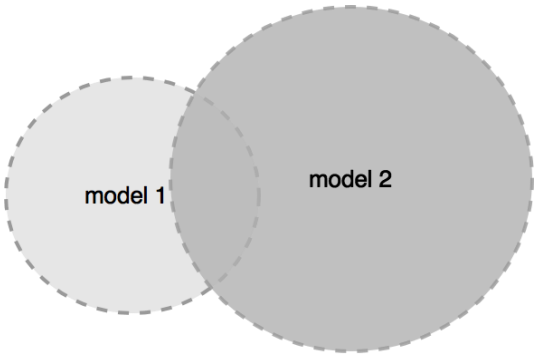

Figure 8-1 Example of distribution of confident predictions while using two models

The percentage the predictions improved when using a model alongside the baseline was calculated as $\frac{added}{before}$, following the notation from Figure 8-2.

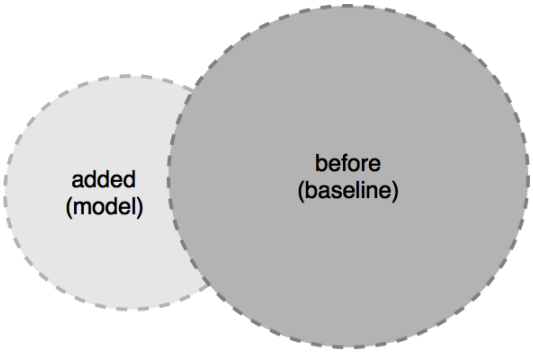

Figure 8-2 Improvement for classifying documents when using a model alongside the baseline

8.1. MinMaxSum of norms

No conclusion could be made whether it is better to use a subject-specific corpus alongside the baseline model; both Word2vec (Figure 8-3) and GloVe (Figure 8-4) showed varying results depending on the dataset. Approximately 42% of phenotype traits showed an improvement, which averaged around 10%.

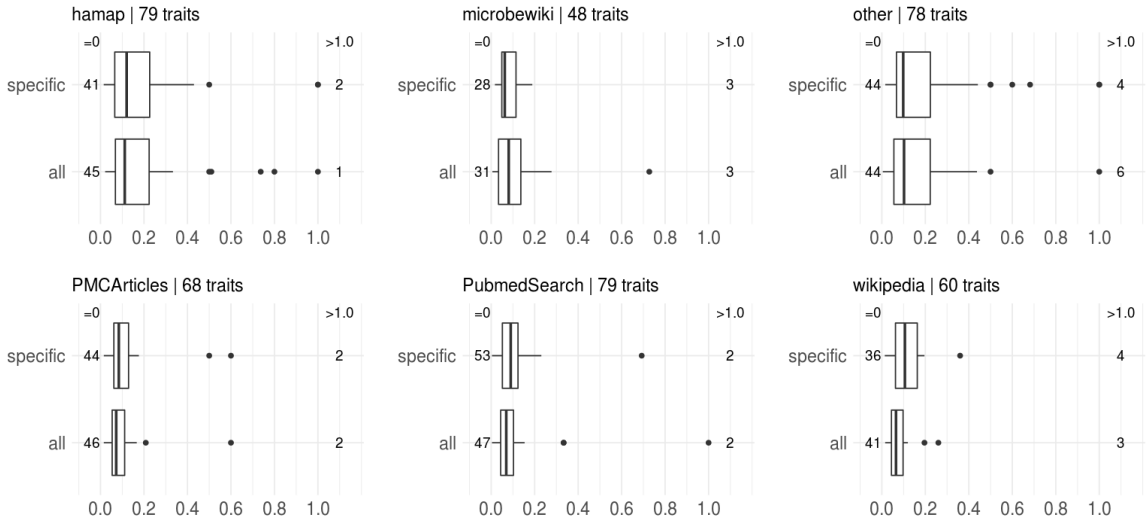

Figure 8-3 Percentage of improvement for *MinMaxSum* of norms using Word2vec embedding with size 300 and RF classifier; extreme values (0 and >1) are counted on each side

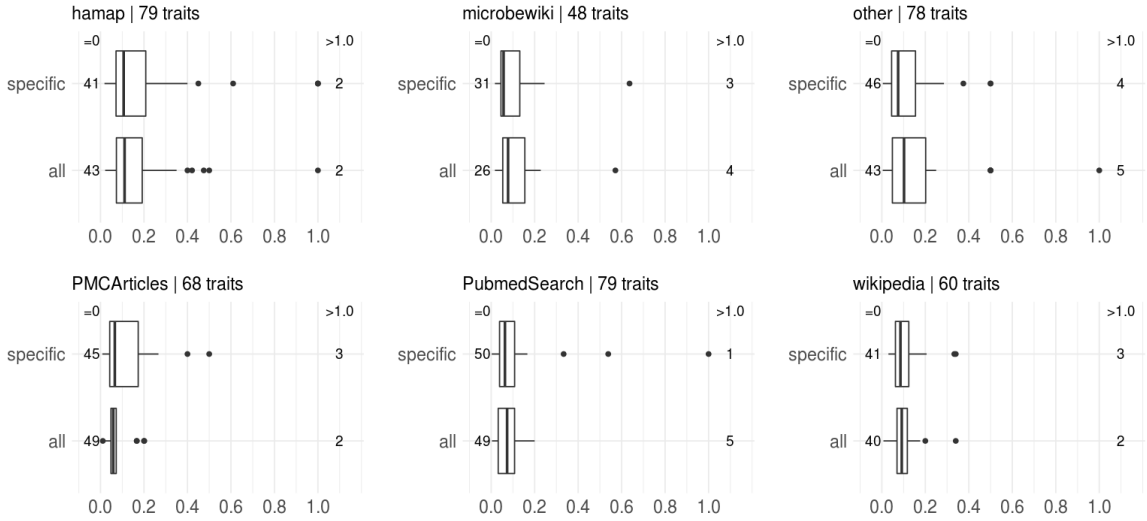

Figure 8-4 Percentage of improvement for *MinMaxSum* of norms using GloVe embedding with size 300 and RF classifier; extreme values (0 and >1) are counted on each side

8.2. Weighted embeddings

Word2vec embedding had same or higher percentage improvement of high-confidence annotations while using alongside the baseline model than Glove. Using a more subject specific corpus showed better results with Word2vec (Figure 8-5), but not with GloVe (Figure 8-6). Using weighted embeddings approach alongside the baseline model showing better results than using *MinMaxSum* of norms.

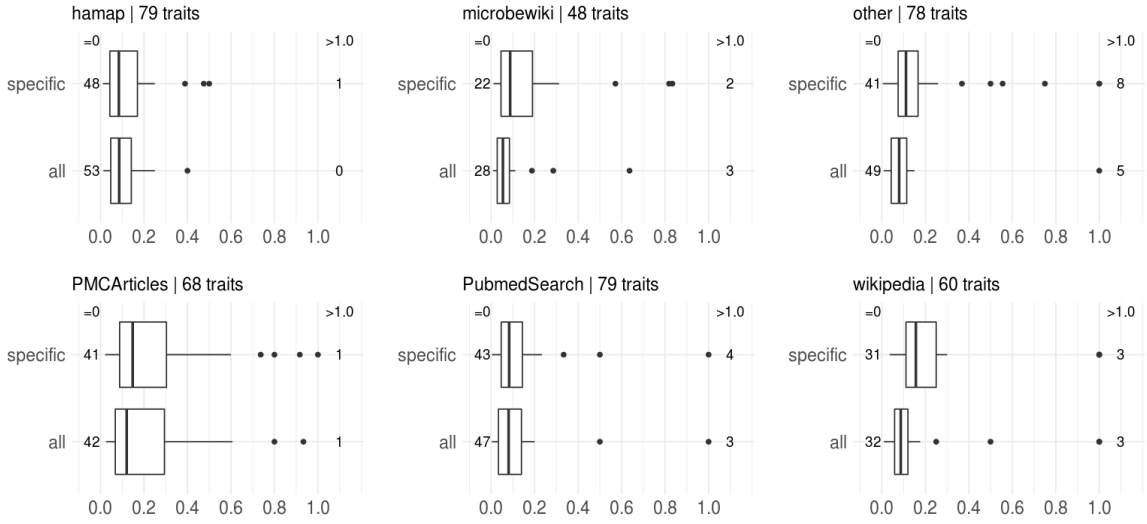

Figure 8-5 Percentage of improvement for weighted aggregation using Word2vec embedding with size 300 and RF classifier; extreme values (0 and >1) are counted on each side

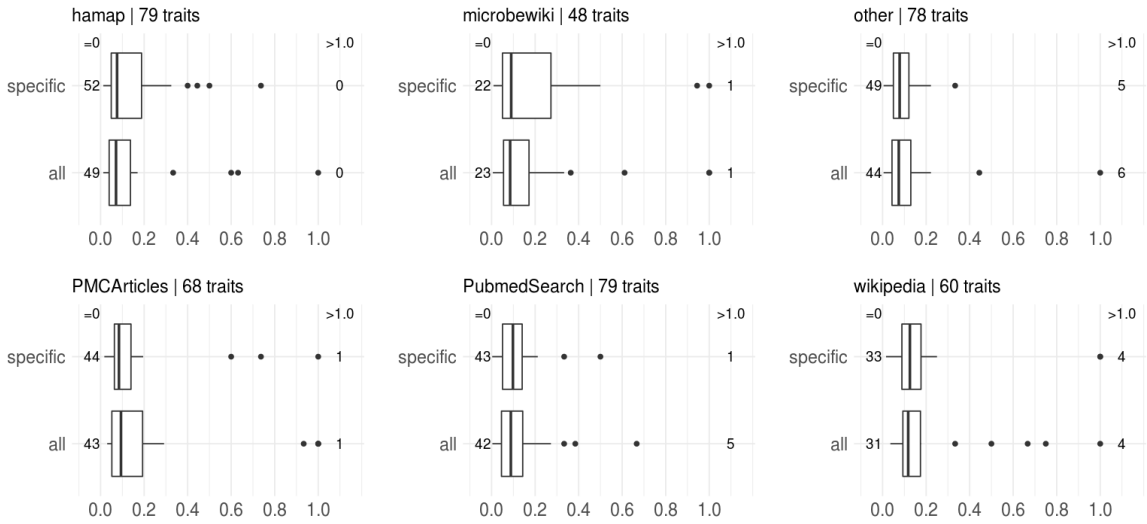

Figure 8-6 Percentage of improvement for weighted aggregation using GloVe embedding with size 300 and RF classifier; extreme values (0 and >1) are counted on each side

9. Conclusion

Using a simple bag-of-words with *tf-idf* weighting outperformed all proposed models based on word embeddings, even though the initial idea was that using embeddings should result in overall more accurate models.

The GloVe embedding did not show a substantial net difference between using the entire corpus of 1.5M documents and its subject specific subset making up only 5% of the documents. Word2vec embedding had shown slightly better results while using a subject-specific corpus, which could be attributed to the fact that the global word co-occurrence was not available during training, as opposed to GloVe. This was consistent throughout all our tests and especially expressed when using the weighted embedding approach (even significant in some datasets when the corpora were reduced to the same size). Using highly subject-specific corpora showed a deterioration in performance, which could be caused by small corpora size.

Even though the weighted embedding approach had an overall worse performance than *MinMaxSum* of norms, using it alongside the baseline model showed a greater increase in the correct classification of the minority class. This means that it could correctly classify documents that the baseline BoW model was not confident about, and that it could be used alongside the existing model to increase performance.

The approach using LSTMs hidden layer to represent document embeddings was prone to overfitting. Long-term connections could be a problem when having a small dataset, as the model could learn a combination of word embeddings that appear only in the training set.

Using word embeddings in classifying documents is a promising approach, but the challenges ahead include how to choose a corpus to train the embedding and how to represent documents. Further improvements could possibly be made in the training of word embeddings, as both Word2vec and GloVe were trained using default parameters, such as context size or number or epochs. These parameters were chosen by their respective authors because they showed good results, but given the specificity of biomedical corpora, such parameters may not be optimal.

Bibliography

[1] Mikolov, Tomas; Chen, Kai; Corrado, Greg; Dean, Jeffrey;, "Efficient Estimation of Word Representations in Vector Space," arXiv preprint arXiv:1301.3781v3, 2013.

[2] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation," *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532-1543, 2014.

[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *Advances in neural information processing systems*, pp. 3111-3119, 2013.

[4] Y. Goldberg and O. Levy, "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method," arXiv preprint arXiv:1402.3722v1, 2014.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, pp. 1735-1780, 1997.

[6] C. Olah, "colah's blog, Understanding LSTM Networks," 27 August 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs. [Accessed 5 May 2017].

[7] "PubMed Central," [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/. [Accessed 3 March 2017].

[8] "PubMed Central FTP site," [Online]. Available: ftp://ftp.ncbi.nlm.nih.gov/pub/pmc. [Accessed 3 March 2017].

[9] Python Core Team, "Python: A dynamic, open source programming," Python Software Foundation, 2017. [Online]. Available: https://www.python.org/.

[10] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta, ELRA, 2010, pp. 45-50.

[11] maciejkula, "GloVe Python," GitHub repository, 2017. [Online]. Available: https://github.com/maciejkula/glove-python.

[12] S. Bird, E. Klein and E. Loper, Natural language processing with Python: analysing text with the natural language toolkit, O'Reilly Media, Inc., 2009.

[13] F. M. Porter, "An algorithm for suffix stripping," *Program 14.3,* pp. 130-137, 1980.

[14] "MeSH (Medical Subject Headings)," National Center for Biotechnology Information, [Online]. Available: https://www.ncbi.nlm.nih.gov/mesh.

[15] P. J. A. Cock , T. Antao , J. T. Chang , B. A. Chapman , C. J. Cox , A. Dalke , I. Friedberg , T. Hamelryck , F. Kauff , B. Wilczynski and M. J. L. de Hoon, "Biopython: freely available Python tools for computational molecular biology and bioinformatics," 2009. [Online]. Available: https://doi.org/10.1093/bioinformatics/btp163.

[16] M. Brbić, M. Piškorec, V. Vidulin, A. Kriško, T. Šmuc and F. Supek, "The landscape of microbial phenotypic traits and associated genes," *Nucleic Acids Research 44.21,* pp. 10074-10090, 2016.

[17] 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 É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research vol. 12*, p. 2825−2830, 2011.

[18] S. Arora, Y. Liang and T. Ma, "A Simple but Tough-to-Beat Baseline for Sentence Embeddings," *ICLR 2017 conference submission,* 2016.

[19] W. Zaremba, I. Sutskever and O. Vinyals, "Recurrent Neural Network Regularization," *arXiv:1409.2329,* 2015.

[20] G. Hinton, N. Srivastava and K. Swersky, "rmsprop: Divide the gradient by a running average," *Neural Networks for Machine Learning,* p. slide 26.

[21] F. Chollet and others, "Keras," GitHub repository, 2015. [Online]. Available: https://github.com/fchollet/keras.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015.

[23] L. Breiman, "Random Forests," *Machine Learning* , vol. 45, no. 1, pp. 5-32, 2001.

[24] R. C. Team, "R: A Language and Environment for Statistical Computing," R Foundation for Statistical Computing, Vienna, 2015.

[25] M. P. Fay and A. P. Shaw, "Exact and Asymptotic Weighted Logrank Tests for Interval Censored Data: The R Package," *Journal of Statistical Software*, vol. 36, no. 2, pp. 1-34, 2010.

Klasifikacija bioloških anotacija na velikoj skali koristeći reprezentacije riječi izvedene iz dokumenata biomedicinske znanstvene literature

Sažetak

Naučeni su vlastiti Word2vec i GloVe modeli prikaza riječi za znanstvenu literaturu u području biomedicine, kao i tri klasifikacijske metode za diskriminaciju fenotipova, dvije temeljene na agregaciji vektorskog prikaza riječi, i jedna na rekurentnim neuronskim mrežama. Prikazi riječi su trenirani na velikom korpusu znanstvenih članaka i njegovim tematski specifičnim podskupovima. Rezultati klasifikacije su testirani na 6 izvora dokumenata. Pokazano je da Word2vec postiže bolje rezultate kada je treniran na tematski specifičnom podskupu koji je sačinjen od 4.9% od ukupnih članaka nego kada je treniran na cijelom korpusu. Korištenje rekurentnih neuronskih mreža imalo je problema s prenaučenosti, moguće zbog predugačkih dokumenata ili premalog skupa za učenje. Iako predloženi modeli nisu bili bolji od stroja potpornih vektora koristeći prikaz vreće riječi, pokazano je da korištenje agregacijskih metoda uz bazni model povećava količinu ispravne klasifikacije manjinske klase kod nekih fenotipova za oko 10%.

Ključne riječi: vektorski prikaz riječi, Word2vec, GloVe, RNN, LSTM, klasifikacija fenotipova, specifičnost korpusa

Classification of Large-Scale Biological Annotations Using Word Embeddings Derived from Corpora of Biomedical Research Literature

Abstract

Custom Word2vec and GloVe embeddings for scientific literature in the biomedical domain were trained, as well as three classification methods for discriminating phenotype traits, two of which were based on aggregating word embeddings and one on recurrent neural networks. Word embeddings were trained on a large corpus of scientific articles and its more subject-specific subsets. Classification performance was tested on 6 document sources. It was shown that Word2vec achieves better performance when trained on a subject-specific subset corpus comprised of 4.9% articles, than when trained on the entire corpus. Using recurrent neural networks had an overfitting problem, possibly because the documents were too long or the training set too small. Although the proposed models did not outperform support vector machine using bag-of-words, it was shown that using the aggregation methods alongside the baseline model increases the amount of correctly classified minority class in some phenotype traits by around 10%.

Keywords: word embedding, Word2vec, GloVe, RNN, LSTM, phenotype classification, corpus specificity