Bag-of-Embeddings for Text Classification

Peng Jin,[1] **Yue Zhang,**[2] **Xingyuan Chen,**[1*] **Yunqing Xia**[3]
[1] School of Computer Science, Leshan Normal University, Leshan China, 614000
[2] Singapore University of Technology and Design, Singapore 487372
[3] Search Technology Center, Microsoft, Beijing China, 100087

Abstract

Words are central to text classification. It has been shown that simple Naive Bayes models with word and bigram features can give highly competitive accuracies when compared to more sophisticated models with part-of-speech, syntax and semantic features. Embeddings offer distributional features about words. We study a conceptually simple classification model by exploiting multi-prototype word embeddings based on text classes. The key assumption is that words exhibit different distributional characteristics under different text classes. Based on this assumption, we train multi-prototype distributional word representations for different text classes. Given a new document, its text class is predicted by maximizing the probabilities of embedding vectors of its words under the class. In two standard classification benchmark datasets, one is balance and the other is imbalance, our model outperforms state-of-the-art systems, on both accuracy and macro-average F-1 score.

1 Introduction

Text classification is important for a wide range of web applications, such as web search [Chekuri *et al.*, 1997], opinion mining [Vo and Zhang, 2015], and event detection [Kumaran and Allan, 2004]. Dominant approaches in the literature treat text classification as a standard classification problem, using supervised or semi-supervised machine learning methods. A key research topic is the design of effective feature representations.

Words are central to text classification. Bag-of-words models [Harris, 1954], such as Naive Bayes [Mccallum and Nigam, 1998] can give highly competitive baselines compared with much more sophisticated models with complex feature representations. The main intuition behind is that there is typically a salient set of words that signal each document class. For example, in news the words "coach", "sport", "basketball" occur relatively frequently in sports, and the words "chipset", "compiler" and "Linux" are relatively unique for information technology.

*Corresponding author: Xingyuan Chen (1045258214@qq.com)

More sources of information have been explored for text classification, including parts of speech [Lewis, 1995], syntax structures [Post and Bergsma, 2013; Tetsuji *et al.*, 2010] and semantic compositionality [Moilanen and Pulman, 2007]. However, such features have demonstrated limited gains over bag-of-words features [Wang and Manning, 2012]. One useful feature beyond bag-of-words is bag-of-ngrams. Wang and Manning [2012] show that bigram features are particularly useful for sentiment classification. Bigrams offer a certain degree of compositionality while being relatively less sparse compared with larger n-gram features. For example, the bigram "abnormal return" strongly indicates finance, although both "abnormal" and "return" can be common across difference classes. Similar examples include "world cup" and "large bank", where bi-grams indicate text classes, but the words do not.

One intuitive reason behind the strength of bigrams is that they resolve the ambiguity of polysemous words. In the above examples the words "return", "cup", and "bank" have different meanings under different document classes, and the correct identification of their word sense under a ngram context is useful for identifying the document class. For example, when the word "bank" exists under a context with words such as "card" and "busy", it strongly indicates the "finance" sense. This fact suggests a simple extension to bag-of-word features by incorporating **context** and **word sense** information. We propose a natural extension to the skip-gram word embedding model [Mikolov *et al.*, 2013] to this end.

Word embeddings are low-dimensional, dense vector representation of words, first proposed in neural language models [Bengio *et al.*, 2003]. Traditionally, embeddings are trained during the training of a neural language model as a part of the model parameters. Mikolov *et al.* [2013] define specific objective functions for efficient training of word embeddings, by simplifying the original training objective of a neural language model. The skip-gram model trains word embeddings by maximizing the probabilities of words given their context windows. Two sets of embeddings are defined for the same word as a target word and a context word, respectively. The probability of a target word is estimated by the cosine similarities between the target embedding and the content embeddings of its context words. This offers a way to estimate word probability via the embedding probability that readily integrates context information.

To further integrate word sense information, we make a simple extension by training multi-prototype target word embeddings, with one distinct vector trained for a word under each class. The context vectors of words remain the same across different classes. Here by associating word senses with document classes, we make the assumption that each word exhibits one sense in each document class, and the sense of a word differs across different classes. This assumption is highly coarse-grained and does not correspond to the definition of word senses in linguistics. However, it empirically works effectively, and we find that the definition of *sense* here can capture subtle differences between word meanings across different document classes.

Under the above assumptions, the probability of a class given a document can be calculated from the probabilities of the embeddings of each word under this class. Since the probability of each word embeddings is calculated separately, we call this model a bag-of-embeddings model. Training requires text corpora with class labels, some of them are obtained from naturally labeled [Go *et al.*, 2009] and some are hand-labeled [Lewis, 1995]. The bag-of-embeddings model is conceptually simple, with the only parameters being word embeddings. We show that maximum-likelihood training for document classification is consistent with the skip-gram objective for training multi-prototype word embeddings.

Experiments on two standard document classification benchmark data show that our model achieve higher accuracies and macro-F1 scores compared than state-of-the-art models. Our method achieves the best reported accuracies for both the balanced and the imbalanced datasets. The source code of this paper is released at *https://github.com/hiccxy/Bag-of-embedding-for-text-classification*.

2 Related Work

Text classification has traditionally been solved as a standard classification task, using supervised learning approaches such as Naive Bayes [Mccallum and Nigam, 1998], logistic regression [Nigam *et al.*, 1999], and support vector machines [Joachims, 1998]. Word features, and in particular bag-of-words features, have been used for classification. There have been research efforts to incorporate more complex features into text classification models. Lewis [1995] uses parts-of-speech and phrase information for text categorization; Tetsuji *et al.* [2010] use dependency tree features for sentiment classification; Post and Bergsma [2013] integrate syntax into text classification models by explicit features and implicit kernels; Moilanen and Pulman [2007] model semantic compositionality for sentiment classification. Wang and Manning [2012] show that bag-of-words and bigram features can give competitive accuracies compared with such more complex features.

Distributed word representations [Bengio *et al.*, 2003; Collobert *et al.*, 2011; Mikolov *et al.*, 2013] have been typically used as additional features in discrete models for semi-supervised learning, or as inputs to neural network models. For text classification, word embeddings have mostly been used as inputs to neural models such as recursive tensor networks [Socher *et al.*, 2011], dynamic pooling network [Kalchbrenner *et al.*, 2014] and deep convolutional neural network [Santos and Gatti, 2014]. There has also been work on directly learning distributed vector representations of paragraphs and sentences [Le and Mikolov, 2014], which has been shown as useful as the above mentioned neural network models for text classification. Such neural network result in vector representation of text data. Along this line, Tang *et al.* [2014] is the closet in spirit to our work. They learn text embeddings specifically for end tasks, such as classification. We also learn word embeddings specifically for text classification. However, rather than learning one vector for each word, which Tang *et al.* [2014] do, we learn multi-prototype embeddings, with certain words having multiple vector forms according to the text class. In addition, we have a very simple document model, with the only parameters being word vectors. The simplicity demonstrates the effectiveness of incorporating text class information into distributed word representation.

Our work is also related to prior work on multi-prototype word embeddings [Reisinger and Mooney, 2010; Huang *et al.*, 2012; Tian *et al.*, 2014]. Previous methods define word prototypes according to word senses from ontologies, or induce word senses automatically. They share a more linguistic focus on word similarities. In contrast, our objective is to improve text classification performance, and hence we train multi-prototype embeddings based on text classes.

3 Method

3.1 The Skip-gram Model

Our bag-of-embeddings model extends the skip-gram model [Mikolov *et al.*, 2013], which is a simplification of neural language models for efficient training of word embeddings. The skip-gram model works by maximizing the probabilities of words being predicted by their context words.

In particular, two sets of embeddings are defined for each word w, when w is used as the output target word and as the input context word, respectively. We use $\vec{v}(w)$ and $\vec{u}(w)$ to denote **target (output)** embedding vector and **context (input)** embedding vector of w, respectively.

Given a target word w and a context w' of w, the probability of $\vec{v}(w)$ given $\vec{u}(w')$ is defined based on the cosine similarity between $\vec{v}(w)$ and $\vec{u}(w')$, as follows,

$$P(\vec{v}(w)|\vec{u}(w')) = \frac{e^{\vec{v}(w)\cdot\vec{u}(w')}}{\sum_{w''\in V} e^{\vec{v}(w'')\cdot\vec{u}(w')}},$$

where V is the vocabulary.

The model can be regarded as a crude approximation of a language model estimating the probability of a sentence $s = w_1 w_2 ... w_{|s|}$ by

$$P(s) = \prod_{i=1}^{|s|} P(\vec{v}(w_i)|t(w_i, s)),$$

where $t(w_i, s)$ is the context of w_i in s, which typical includes T words before w_i and T words after w_i. $P(\vec{v}(w_i)|t(w_i, s))$ is estimated by predicting $\vec{v}(w_i)$ using each word w' in $t(w_i, s)$.

$$P(\vec{v}(w_i)|t(w_i,s)) = \prod_{w' \in t(w_i,s)} P(\vec{v}(w_i)|\vec{u}(w')) = \prod_{w' \in t(w_i,s)} \frac{e^{\vec{v}(w_i)\cdot\vec{u}(w')}}{\sum_{w'' \in V} e^{\vec{v}(w'')\cdot\vec{u}(w')}} \quad (1)$$

This approximation does not give a highly accurate language model, but can be used to train word embeddings efficiently. The resulting embeddings can be used as input to train more sophisticated neural language model or to solve other NLP tasks [Mikolov *et al.*, 2013].

Training is achieved by maximizing the likelihood of raw text using stochastic gradient descent, which is equivalent to maximizing $logP(\vec{v}(w_i)|\vec{u}(w'))$ for all w_i and w' in a corpus. Because gradients to the probability in Equation (1) requires summary over the vocabulary, the skip-gram model uses a rough approximation to the noise contrastive estimation (NCE) method (Gutmann and Hyvarinen, 2010), which instead maximizes

$$log\sigma(\vec{v}(w_i)\cdot\vec{u}(w')) + \sum_{j=1}^{l} E_{w_j \sim P_n(w)}[log\sigma(-\vec{v}(w_j)\cdot\vec{u}(w'))] \quad (2)$$

Here σ is the sigmoid activation function, $P_n(w)$ is a distribution for negative samples, which is typically the unigram distribution raised to the $3/4$th power (Mikolov et al., 2013), and l is the number of negative samples, set to a small number below 20. Equation 2 approximates the log probability $P(\vec{v}(w_i)\vec{u}(w'))$ by contrastive estimation between seen words and negative samples from a noise distribution, where $l \to \infty$, the gradient closely approximates the gradient of $P(\vec{v}(w_i)\vec{u}(w'))$.

3.2 The Bag-of-embeddings Model for Text Classification

The goal of text categorization is the classification of a given document $d \in D$ into a fixed number of predefined categories C, where D is the set of documents. Although according to [Joachims, 1998], each document d can be in multiple, exactly one, or no category at all, in this paper we simplify to find the most likely class $c \in C$ for d. We find c by using

$$c = \arg\max_{c \in C} P(c|d)$$

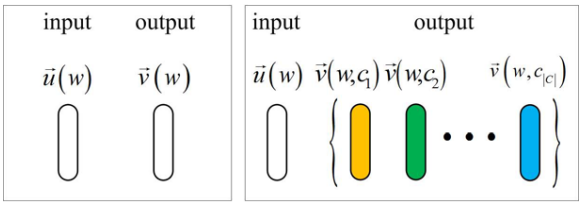
We estimate the class probability $P(c|d)$ by estimating the probability of d's vector form under c, denoted as $\vec{v}(d,c)$. Denoting the words in d as $w_1, w_2, ..., w_{|d|}$, where $w_i \in V$, $\vec{v}(d,c)$ consists of the class-specific **target** embedding sequence of $w_1, w_2, ..., w_{|d|}$.

$$\vec{v}(d,c) = \left[\vec{v}(w_1,c), \vec{v}(w_2,c), ..., \vec{v}(w_{|d|},c)\right]$$

As a result,

$$P(c|d) = P(\vec{v}(d,c)|d) = P(\vec{v}(w_1,c), \vec{v}(w_2,c), ..., \vec{v}(w_{|d|},c)|d)$$

Further by assuming that the class-specific **target** embeddings are conditionally independent i.e. bag-of-embeddings, it follows that



(a) skip-gram model (b) bag-of-embedding model

Figure 1: Illustration of Embeddings.

$$P(c|d) = P(\vec{v}(w_1,c)|d)P(\vec{v}(w_2,c)|d)...P(\vec{v}(w_{|d|},c)|d)$$

Now following the skip-gram model, assume that the probability of $\vec{v}(w_i,c)$ depends only on the context window $t(w_i,d) = w_{i-T}, w_{i-T+1}, ..., w_{i-1}, w_{i+1}, ..., w_{i+T}$ (i.e. $2T$ is the context window size), we have

$$P(\vec{v}(w_i,c)|d) = P(\vec{v}(w_i,c)|t(w_i,d))$$

We use the skip-gram model to estimate this multi-prototype embedding probability. In particular, denoting the **context** embedding of a word w as $\vec{u}(w)$, $P(\vec{v}(w_i,c)|t(w_i,d))$ is estimated by

$$P(\vec{v}(w_i,c)|t(w_i,d)) \approx \prod_{w' \in t(w_i,d)} P(\vec{v}(w_i,c)|\vec{u}(w'))$$

From the above equation, it can be seen that the bag-of-embeddings model is similar to the skip-gram model in the definition of **target** and **context** embeddings. However, it is different in that the target embeddings are class-dependent, and each word can have different target embeddings in different classes. On the other hand, each word has a unique context embedding across classes.

Figure 1 gives an illustration of this difference. Now given $P(\vec{v}(w_i,c)|t(w_i,d))$, the document probability

$$P(c|d) = P(\vec{v}(d,c)|d) \propto \prod_{w_i \in d} \prod_{w' \in t(w_i,d)} e^{\vec{v}(w_i,c)\cdot\vec{u}(w')}$$

Correspondingly,

$$\arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \sum_{w_i \in d} \sum_{w' \in t(w_i,d)} \vec{v}(w_i,c)\cdot\vec{u}(w') \quad (3)$$

The last equation corresponds to the estimation of the log probability, which is more efficient than the estimation of the conditional probability.

The parameters $u(w)$ and $v(w,c_i)$ in Equation 3, which is used for predicting, are trained in a supervised way described in the following subsection.

3.3 Training

Given a set of labeled documents $D = \{(d_1,c_1),(d_2,c_2),...,(d_N,c_N)\}$, the bag-of-embeddings model is trained by maximizing its likelihood function

$$P(D) = \prod_{k=1}^{N} \prod_{w_i \in d_k} \prod_{w' \in t(w_i, d_k)} P(\vec{v}(w_i, c_k) | \vec{u}(w'))$$

By using stochastic gradient descent, maximizing $P(D)$ consists of iteratively maximizing each $P(\vec{v}(w_i, c_k)|\vec{u}(w'))$. This is consistent in form to the objective of the skip-gram model, which maximizes the probability of each word given each context word. The only difference is that the target word vector depends on the label class in the bag-of-embedding model. We follow the skip-gram model and train $P(\vec{v}(w_i, c_k)|\vec{u}(w'))$ using a simplification of NCE by negative sampling, where the objective is

$$log\sigma(\vec{v}(w_i, c) \cdot \vec{u}(w')) + \sum_{j=1}^{l} E_{w_j \sim P_n(w)}[log\sigma(-\vec{v}(w_i, c) \cdot \vec{u}(w_j))]$$

Consistent with the skip-gram model, σ is the sigmoid function, and $P_n(w)$ is a noise distribution of words. Note, however, that in this equation, the context word w' is sampled instead of the target word, because it is relatively more difficult to find a distribution of negative examples that includes the class label. $P_n(w)$ being unigram distribution to the $3/4$th power, and all parameters setting are the same as Mikolov *et al.* [2013] except the iteration number[1]. This is because our corpus is much smaller than theirs. For the parameters, we set $l = 5$, the iteration number to 5, the size of context window to 10 and the dimensions of word vector to 100.

3.4 Correlation with Skip-gram and Bag-of-words

The bag-of-embeddings model is closely related to the skip-gram model and bag-of-words model. Compared with the bag-of-words model for document classification, the bag-of-embeddings model also makes the Naive Bayes assumption, by treating the **target** embedding vector of each word in the document as being independent of each other. On the other hand, it is different from the bag-of-words model in that the model integrates context information. Rather than predicting the text class using unigram alone, it relies on the distributional similarity between each **target** word embedding and the **context** embeddings of words in a context window. With respect to feature space, the bag-of-embeddings model is a much richer model with hundreds of dense parameters to represent each word. Our experiments show that it outperforms a bag-of-words model significantly.

The parameters of the bag-of-embeddings model are target and context embedding vectors. This is closely related to the skip-gram model. The main difference in terms of embeddings is that the bag-of-embeddings model integrates document class information, training multiple-prototype embeddings for each word. The training methods of the skip-gram model and the bag-of-embeddings are shown in Figure 2(a) and 2(b), respectively. Compared with the skip-gram model, the target embeddings of the bag-of-embeddings contain class labels. On the other hand, the context embeddings remain single-prototype, which serves to connect the embeddings space for target word between different classes. One additional difference between the bag-of-embeddings model and the skip-gram model is that the former is designed for document classification, but the latter is specifically designed to train embeddings.

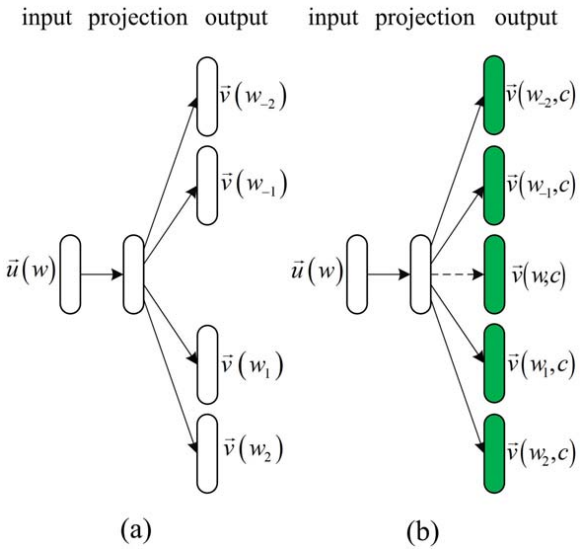


Figure 2: Skip-gram model vs. bag-of-embeddings model.

In our experiments we find that the classification accuracies can be improved by 0.2% to 0.5% by adding w itself into the targets in Figure 2(b), and therefore take this variation in choose the context window $t(w_i, d)$.

4 Experiments

We evaluate our models on two multi-class text classification datasets, one is imbalanced and the other is nearly balanced. The experimental setup is as follows.

4.1 Experiment Setup

Data Sets We choose the twenty newsgroup (20NG)[2] test [Lang, 1995] for multi-class classification. We use the "by-date" data, which consists of 11,314 training instances and 7,532 test instances, which are partitioned (nearly) evenly across the classes. For training, the largest class has 600 instances and the smallest class has 377 instances.

For imbalanced classification, Lewis [1995] introduced the Reuters-21578 corpus[3]. We choose this benchmark dataset as follow Debole and Sebastiani [2005] and perform the R8 evaluation requires that each class should have at least one training and one test example and only the most frequent eight categories (i.e. R8) are evaluated. R8 consists of 5,485 documents for training and 2,189 for testing. For training, the largest class has 2,840 instances and the smallest class has 41

[1]Empirically, the sampling method gives highly competitive accuracy when the iteration number is greater than 10 and the performance is very stable.

[2]http://qwone.com/~jason/20Newsgroups/

[3]http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 1: Overall results.

Model \ Dataset	20NG		R8	
	AC	MF1	AC	MF1
BoW-SVM	79.0	78.3	94.7	85.1
TWE	81.5	80.6	- -	- -
GoW-SVM	- -	- -	95.5	86.4
BoE	**83.1**	**82.7**	**96.5**	**88.6**

instances. The pre-processing is conducted by turning all letters to lowercase, removing all the words which occurs less than five times and replacing all punctuation with space.

Evaluation Metrics We use the standard accuracy (AC), which is defined as the number of correctly predicted documents out of all test documents. Precision, recall and F-1 score are also used. The macro-average F1- score (MF1) takes into account the skewed class label distributions by weighting each class uniformly.

Baseline Models We compare our models with several strong baseline models, including Support Vector Machines, and two state-of-the-art text representation models.

- SVM: We take a bag-of-words baseline using SVM, which outperforms the Naive Bayes model. Follows Wang *et al.* [2013], we use LIBSVM[4] with a linear kernel and optimize the parameter $C \in 1e-4, ..., 1e+4$ by ten-fold cross-validation. Bag-of-words features are used to represent the documents. It is denoted as "BoW-SVM" in Table 1.
- TWE: Liu *et al.* [2015] employ latent topic models to assign topics to each word in the whole corpus, and learn topical word embeddings based on both words and their topics (i.e. each word has different embeddings under different topics). Three models are proposed, and the best one TWE-1, regards each topic as a pseudo word and learns topic embeddings and word embeddings separately. The model outperforms the paragraph vector model of Le and Mikolov [2014].
- GoW: Rousseau *et al.* [2015] propose three graph-of-words approaches to capture the distance dependency between words. A linear SVM is used as classifier and their best model served as our baseline. It is denoted as "GoW-SVM" in Table 1.

4.2 Results

Table 1 summaries the performances of our model and the baseline models, where "BoE" denotes our proposed bag-of-embedding model. Our model outperforms the state-of-the-art models TWE and GoW on the balanced and the imbalanced data sets, respectively. For both data sets, the bag-of-embeddings model gives the best results that we are aware of the in the literature.

Tables 2 and 3 show detailed comparisons between our bag-of-embeddings model and the bag-of-words SVM on the balanced and imbalanced data sets, respectively, where the higher precision or recall is shown in bold. Our model gives higher precision for most of the classes, and higher recall for nearly all the classes. This demonstrates the effectiveness of the bag-of-embeddings model in leveraging context information. The higher recall also shows the advantage of dense features in reducing feature sparsity as compared with discrete bag-of-word features.

[4]https://www.csie.ntu.edu.tw/˜cjlin/libsvm/

Table 2: Class-level results on the balanced dataset.

Class Name	SVM		BoE	
	Pre.	Rec.	Pre.	Rec.
alt.atheism	**67.8**	72.1	60.1	**87.4**
comp.graphics	67.1	73.5	**67.2**	**78.4**
comp.os.ms-windows.misc	**77.1**	66.5	70.3	**74.6**
comp.sys.ibm.pc.hardware	62.8	72.4	**65.6**	**76.5**
comp.sys.mac.hardware	**77.4**	78.2	76.8	**81.8**
comp.windows.x	83.2	73.2	**87.1**	**83.5**
misc.forsale	81.3	**88.2**	**85.9**	84.6
rec.autos	80.7	82.8	**91.8**	**90.2**
rec.motorcycles	92.3	87.9	**94.4**	**94.0**
rec.sport.baseball	89.8	89.2	**96.8**	**92.7**
rec.sport.hockey	93.3	93.7	**97.2**	**96.2**
sci.crypt	92.2	86.1	**93.6**	**91.7**
sci.electronics	70.9	**73.3**	**85.3**	64.9
sci.med	79.3	**81.3**	**95.5**	79.8
sci.space	90.2	**88.3**	**90.5**	86.6
soc.religion.christian	77.3	**87.9**	**89.7**	85.7
talk.politics.guns	71.7	85.7	**77.1**	**90.7**
talk.politics.mideast	91.7	76.9	**95.4**	**88.8**
talk.politics.misc	71.7	56.5	**77.6**	**61.6**
talk.religion.misc	63.2	55.4	**74.1**	**62.1**

Table 3: Class-level results on the imbalanced dataset. "Tr.#" denotes the number of training instances and "Te.#" denotes the number of test instances.

Class Name	Tr.#	Te.#	SVM		BoE	
			Pre.	Rec.	Pre.	Rec.
acq	1,596	696	94.2	97.1	**97.7**	**98.0**
crude	253	121	93.3	92.6	**97.4**	**94.2**
earn	2,840	1,083	98.3	**98.9**	**99.4**	98.6
grain	41	10	**100**	70.0	75.0	**90.0**
interest	190	81	79.2	70.4	**91.2**	**76.5**
money-fx	206	87	75.3	63.2	**81.3**	**89.7**
ship	108	36	**89.3**	69.4	81.8	**75.0**
trade	251	75	**86.4**	93.3	80.2	**97.3**

4.3 Analysis

To better understand the mechanism of disambiguation by the bag-of-embeddings model, we analyze the interactions between target and context vectors, and the correlation between target embedding vectors under different classes. Figure 3 (a) and (b) shows the probabilities of the words "win" and "runs" as calculated using different context word embeddings, respectively. We choose the most popular context words from different classes by following Lacoste-Julien *et al.* [2008].

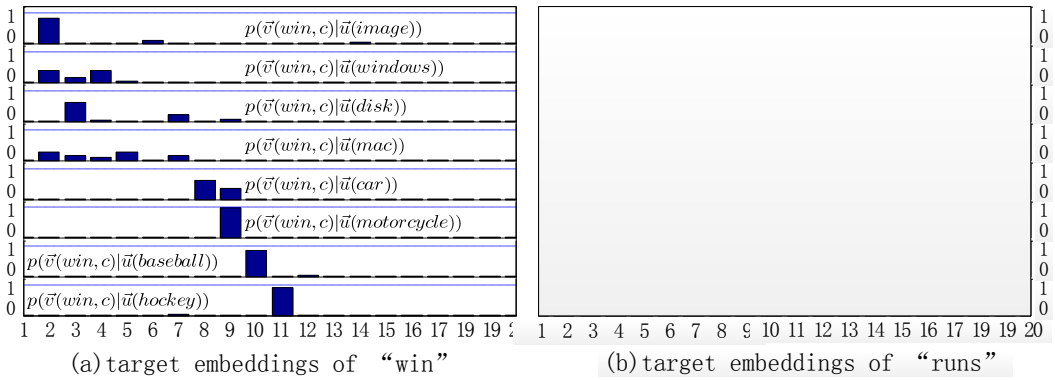


Figure 3: Target embedding probabilities calculated by different context vectors.

Table 4: Nearest neighbors given a target embeddding in a specific class, "tw": target word, "wf": target word frequency, "No": class number.

tw	wf	No	neighbours
win	5	2	transmitting, editor, monochrome, xm, xlib
win	261	3	windows, use, just, dos, program
win	33	4	windows, running, tue, dos, gates
win	7	5	tom, battle, runs, situation, viewed
win	5	8	metro,creator, sell, gen, wagon
win	5	9	chance, taught, single, rider, demonstrating
win	188	10	team, pitching, won, games, game
win	276	11	vs, game, tired, leafs, series
runs	41	2	software, supports, platforms, unix, workstations
runs	14	3	non, applications, windows, dos, use
runs	18	4	installed, set, hard, install
runs	13	5	cpu, wondered, week, dx, breaks
runs	5	8	deeper, theoretically, stronger, lowly, steep
runs	18	9	through, just, bikes, bought, day
runs	234	10	scored, run, game, team, games
runs	13	11	bang, hitter, things, press, hype

The X-axis represents the target embeddings in the 20 classes in Table 2, in the same order.

It can be seen from the Figure 3 that the same words have highly different probabilities across different classes, reflected by the inner product of their class-sensitive target embedding and a context word embedding. In addition, context words have a large influence on the distributions. For example, the word "win" has a large probability under the computer graphics class when the context word is "image", yet large probabilities in the two sports classes when the context word are "baseball" and "hockey", respectively. This demonstrates how word and context information interact in the **bag-of-embeddings** model. In contrast, a **bag-of-word** model relies on words themselves rather than word-context relations for disambiguating document classes.

Within each class, the training of target embedding vectors is essentially the same as the **skip-gram** model, and therefore the embeddings trained for the classification model should exhibit a certain degree of distributed similarities. Table 4 confirms the intuition. For example, under various classes, the closest words to the word "win" include "windows", "dos" and "won", which are synonyms or morphological variations of the word, and "use", "chance" and "game", which are syntactics or semantically related words in a distributed context. In addition, the closest words of the same word differ when the class change, again showing the class sensitivity of the embeddings. For example, under class 3, namely comp.os.ms-windows.misc in Table 2, the word "win" is close to "windows", "dos" and "program". On the other hand, under class 10, rec.sports.baseball, "win" is close to "team", "pitching" and "game", demonstrating class-specific distributional similarity.

5 Conclusion and Future Work

We built a text classifier by using a Naive Bayes model with bag-of-embeddings probabilities. Compared with bag-of-word models, the bag-of-embeddings model exploits contextual information by deriving the probabilities of class-sensitive embedding vectors from their inner product with context words. Our model is conceptually simple, with only parameters being embedding vectors, trained using a variation of the skip-gram method. Experiments on two standard datasets showed that our model outperforms state-of-the-art methods for both balanced and imbalanced data.

Future work includes two directions. First, we would consider leveraging unlabeled data for semi-supervised [Tang *et al.*, 2015]. Second, we would exploit neural documents models, which contains richer parameters spaces by capturing word relations, for potentially better accuracies.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61373056, 61272221 and 61272233).

References

[Bengio *et al.*, 2003] Yoshua Bengio, Rejean Ducharme, Vincent Pascal, and Christian Jauvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[Chekuri *et al.*, 1997] Chandra Chekuri, Michael Goldwasser, Prabhakar Raghavan, and Eli Upfal. Web search using automatic classification. In *Proc. of WWW*, 1997.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, 2011.

[Debole and Sebastiani, 2005] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *J. Am. Soc. Inf. Sci. Tec.*, 56(6):584–596, 2005.

[Go *et al.*, 2009] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision, technical report, dept. of comp. sci., stanford univ. 2009.

[Harris, 1954] Zellig Harris. Distributional structure. *Word*, 10(1):146–162, 1954.

[Huang *et al.*, 2012] Eric Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proc. of ACL*, 2012.

[Joachims, 1998] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proc. of ECML*, 1998.

[Kalchbrenner *et al.*, 2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proc. of ACL*, 2014.

[Kumaran and Allan, 2004] Giridhar Kumaran and James Allan. Text classification and named entities for new event detection. In *Proc. of SIGIR*, 2004.

[Lacoste-Julien *et al.*, 2008] Simon Lacoste-Julien, Fei Sha, and Michael Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Proc. of NIPS*, 2008.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Proc. of ICML*, 1995.

[Le and Mikolov, 2014] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proc. of ICML*, 2014.

[Lewis, 1995] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. of SIGIR*, 1995.

[Liu *et al.*, 2015] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Proc. of AAAI*, 2015.

[Mccallum and Nigam, 1998] Andrew Mccallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proc. of AAAI Workshop on Learning for Text Categorization*, 1998.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. of ICLR Workshop*, 2013.

[Moilanen and Pulman, 2007] Karo Moilanen and Stephen Pulman. Sentiment composition. In *Proc. of RANLP*, 2007.

[Nigam *et al.*, 1999] Kamal Nigam, John Lafferty, and Andrew Mccallum. Using maximum entropy for text classification. In *Proc. of IJCAI Workshop on Machine Learning for Information Filtering*, 1999.

[Post and Bergsma, 2013] Matt Post and Shane Bergsma. Explicit and implicit syntactic features for text classification. In *Proc. of ACL*, 2013.

[Reisinger and Mooney, 2010] Joseph Reisinger and Raymond Mooney. Multi-prototype vector-space models of word meaning. In *Proc. of NAACL*, 2010.

[Rousseau *et al.*, 2015] Francois Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. Text categorization as a graph classification problem. In *Proc. of ACL*, 2015.

[Santos and Gatti, 2014] Cicero Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proc. of COLING*, 2014.

[Socher *et al.*, 2011] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. of EMNLP*, 2011.

[Tang *et al.*, 2014] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proc. of ACL*, 2014.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proc. of KDD*, 2015.

[Tetsuji *et al.*, 2010] Nakagawa Tetsuji, Inui Kentaro, and Kurohashi Sadao. Dependency tree-based sentiment classification using crfs with hidden. In *Proc. of NAACL-HLT*, 2010.

[Tian *et al.*, 2014] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. A probabilistic model for learning multi-prototype word embeddings. In *Proc. of COLING*, 2014.

[Vo and Zhang, 2015] Duytin Vo and Yue Zhang. Target-dependent twitter sentiment classification with rich automatic features. In *Proc. of IJCAI*, 2015.

[Wang and Manning, 2012] Sida Wang and Christopher D. Manning. Baselines and bigrams: Simple, good sentiment and text classification. In *Proc. of ACL*, 2012.

[Wang *et al.*, 2013] Quan Wang, Jun Xu, Hang Li, and Nick Craswell. Regularized latent semantic indexing: A new approac to large-scale topic modeling. *ACM. Trans. Inf. Sys.*, 31(1):5–48, 2013.