Bootstrapping Dialog Systems with Word Embeddings

Gabriel Forgues, Joelle Pineau
School of Computer Science
McGill University
{gforgu, jpineau}@cs.mcgill.ca

Jean-Marie Larchevêque, Réal Tremblay
Nuance Communications, Inc.
{jean-marie.larcheveque,
real.tremblay}@nuance.com

Abstract

One of the main tasks of a dialog system is to assign intents to user utterances, which is a form of text classification. Since intent labels are application-specific, bootstrapping a new dialog system requires collecting and annotating in-domain data. To minimize the need for a long and expensive data collection process, we explore ways to improve the performance of dialog systems with very small amounts of training data. In recent years, word embeddings have been shown to provide valuable features for many different language tasks. We investigate the use of word embeddings in a text classification task with little training data. We find that count and vector features complement each other and their combination yields better results than either type of feature alone. We propose a simple alternative, vector extrema, to replace the usual averaging of a sentence's vectors. We show how taking vector extrema is well suited for text classification and compare it against standard vector baselines in three different applications.

1 Introduction

Dialog systems are gaining popularity and can now be found in cars, televisions, phones and other devices. Two essential components of such systems are: a speech recognition engine to transcribe speech into text and a language model to understand the text's intent. While speech recognition can now often boast impressive accuracy, language understanding is still very difficult in comparison. Here we focus on the text classification task which aims to identify the intent of some short piece of text such as a single utterance. Although dialog systems deployed with large amounts of training data can train complex models to reach high accuracy on this task, their performance hinges on a large domain-specific data collection effort. The most representative in-domain data is collected from real user speech after deploying the dialog system, but data collection can be a lengthy and costly process. In practice, new dialog systems can be seeded with a few artificial samples and then deployed to interact with users. However, the small amount of training data can limit users to shallow dialogs. The conversational data can then be annotated and used to train a better model. Many such iterations might occur before the system reaches high accuracy. A strong start out of the gate can be expected not only to improve early user experience, but also to positively impact dialog quality over the long term. We accordingly focus on improving dialog systems in their early stages when very little training data is available. We consider tasks with very short texts (e.g. single sentences) and many classes, and where texts are semantically rich despite their short length.

2 Word embeddings

In recent years, several algorithms have been proposed to learn word embeddings, also known as word vectors or distributed representations of words [1,2,3]. These embeddings encode words as vectors such that words with similar meanings have similar vector representations. Although large amounts of data are needed to learn effective representations, the data need not be domain-specific or labelled, which makes word vectors well suited as additional features to bootstrap text classification.

2.1 Sentence-level vectors using extrema

Since our goal is not to learn new word embeddings, we assume we have access to vectors which have been pre-trained on some large unlabelled text [4,5]. We must then encode sentences with a variable number of vectors into a fixed-length feature representation. The standard approach sums or averages the sentence's vectors. A simple improvement uses a weighted average where important words carry higher weight. This approach requires some method of computing word weights, such as the commonly-used inverse document frequency (IDF) [6]. While more complex approaches have been proposed [6,7], they typically require additional data in the form of context or extra labels.

Although there are many algorithms to induce word embeddings, the training objectives frequently involve maximizing the similarity of vectors of words occurring in similar contexts while minimizing the similarity with other words. Since the vectors can contain negative as well as positive values, this effectively pulls the vectors of common words towards the zero vector, so that they are not too dissimilar to the entire vocabulary. On the other hand, context-specific words are pushed away from zero, either in the negative or positive direction. Since these rarer words tend to strongly convey intent, we can emphasize them in the sentence's vector by taking the maximum (or minimum) of each dimension $d_i \in D$ from the sentence's set of D-dimensional word vectors. However, since we have no reason to favour either the maximum or minimum, we instead take whichever of the two values is further from zero. We refer to this operation as the vector extrema.

$$\text{extrema}(d_i) = \begin{cases} \max d_i & \text{if } \max d_i \geq |\min d_i| \\ \min d_i & \text{otherwise} \end{cases} \tag{1}$$

2.2 Combining word counts with embeddings

Baroni et al. [8] recently compared context word counts against distributed word representations on tasks such as synonym detection and semantic relatedness between pairs of words, and found that word vectors were overwhelmingly superior. However, their evaluation considered word-level tasks where word embeddings can be directly used as features. The results do not apply to text which must be transformed into a fixed-length representation. For this task, intuition might suggest that word embeddings are especially helpful when there is little training data. In this setting, a vector representation of words gives a basis to relate unseen test words to similar words seen during training. However, as the amount of training data increases, we might expect word counts to be favourable over the averaging of a sentence's word vectors, since the first approach exactly represents each word while the second obscures the presence of individual words. We evaluate whether the two sets of features can be complementary by representing a sentence of words $w_1, ..., w_k$ as a combination of its bag of word counts alongside real-valued features from its word vectors $v_1, ..., v_k$.

BoW	:	$\langle$ bag of word counts $\rangle$
Vec(Avg)	:	$\langle \frac{1}{k}\sum_{i=1}^{k} v_i \rangle$
BoW + Vec(Avg)	:	$\langle$ bag of word counts, $\frac{1}{k}\sum_{i=1}^{k} v_i \rangle$
BoW + Vec(W.Avg)	:	$\langle$ bag of word counts, $\frac{\sum_{i=1}^{k} idf(w_i) v_i}{\sum_{i=1}^{k} idf(w_i)} \rangle$
BoW + Vec(Ext)	:	$\langle$ bag of word counts, extrema$(d_i) \, \forall \, d_i \in D \rangle$

3 Experiments

We evaluate the combination of word count features and word embeddings and compare the vector average and extrema operations. Two dialog datasets from Nuance Communications are used for these experiments: a banking dataset which contains 2,961 unique utterances classified into one of 172 intent labels (deposits, stock quotes, etc.), and a travel dataset of 2,494 utterances and 62 intent

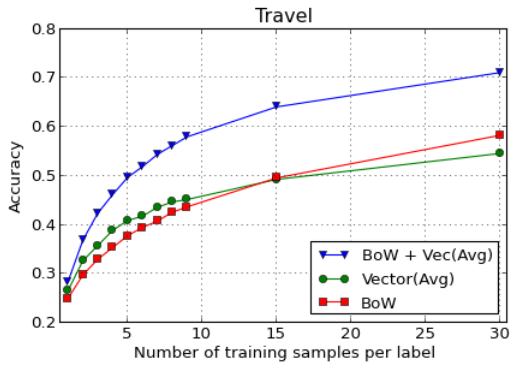

Figure 1: Combination of word count (BoW) and vector average features

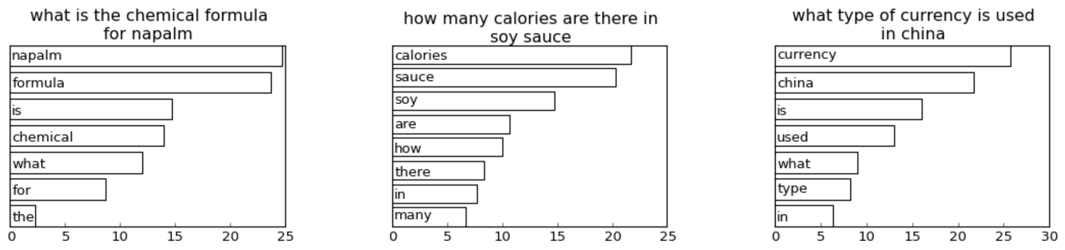

Figure 2: Distribution of extrema words. The x-axis represents the number of extremum values (as a percentage of vector dimension) which each word contributed to the sentence's extrema vector.

labels (book flight, change seats, etc.). We also evaluate the methods on a question classification dataset [9] with 15,452 short questions labelled into 50 classes (person, country, money, etc.).

3.1 Setup

In preliminary experiments, we considered different sets of features to use as baselines such as unigrams, bigrams, part of speech tags and stemming. For the dialog data we evaluated, none of these features provided a significant improvement over unigram word counts alone. We therefore use unigram counts as the baseline feature representation (BoW). We also compared different classifiers, including naive Bayes and logistic regression, but ultimately selected a linear SVM as the best classifier for this task.

We train the classifier in a stratified way to ensure that all classes are seen in training. For some parameter k, we select k samples from each intent label, thereby downsampling the annotated corpus. We run 200 trials for each value of k, where each trial is trained on k random samples from each intent and evaluated on all remaining non-training samples. We report results of accuracy averaged over all trials, with a special focus for $k < 10$ as the smallest number of samples of each class that one might reasonably expect when bootstrapping a new application.

3.2 Preprocessing

We first convert all text to lower-case and split each sentence into tokens for spaces and punctuation. We then remove all single-character words since preliminary experiments showed they were uninformative and their removal led to an increase in accuracy. We also add all label words into the training vocabulary by creating a new artificial sample for each label, where the sample's text consists only of the label's words (e.g. "book flight" for label BOOK_FLIGHT). This ensures the baseline classifier can accurately label the simplest sentences regardless of its training data.

The experiments used the pre-trained word vectors distributed by word2vec [4]. We normalize all vectors by taking the minimum and maximum value of each dimension from the training word vectors, such that the vectors of words seen during training have values which are bounded in [-1,1].

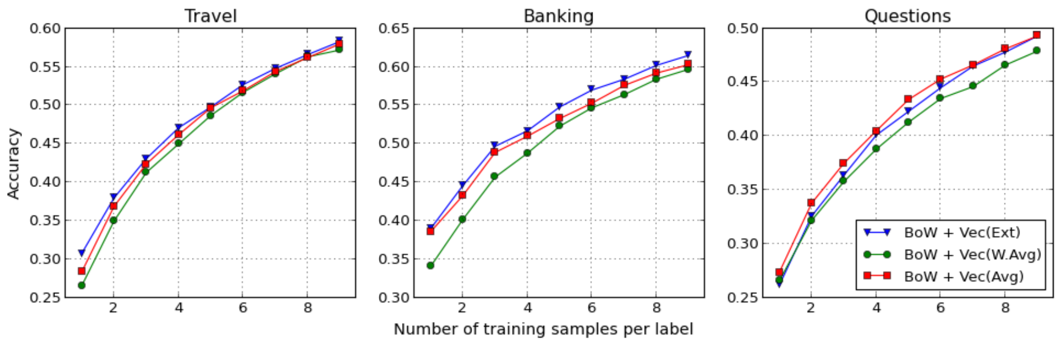

Figure 3: Results comparing vector extrema with IDF-weighted and non-weighted vector averages

4 Results

In Figure 1, we can see a trade-off between word counts (BoW) and vectors. While word vectors are initially superior, this advantage shifts as the amount of training data increases. However, the combination of both features outperforms each set of features individually. While the figure only shows one example domain, the joint use of word counts and vectors was superior to vectors alone for all applications considered, therefore we only report results combining both features hereafter.

The vector extrema operation produced the highest accuracy on both dialog datasets by a small margin and the second highest on the question dataset (Figure 3). The same figure also shows the effect of increasing the amount of training data, with all three domains nearly doubling or tripling in accuracy with less than 10 samples per class. We also analysed the composition of sentence extrema vectors to determine which words contributed the most extreme components. As shown in Figure 2, words which occur in specific contexts (e.g. 'napalm') tend to have the greatest extremum values, while stopwords (e.g. 'the') contribute the least. The results suggest that taking vector extrema effectively induces a global weighing scheme on a sentence's words, where words that are indicative of highly specific contexts in the comprehensive corpus (which was used to learn the word embeddings) have higher weight. This is to be contrasted with the local weights, derived from frequencies in the training set that are used to compute weighted averages.

5 Discussion

While word counts and vectors have often been pitted against each other, our empirical results suggest that they provide features which are complementary, at least to the extent that their combination was advantageous for the three domains we evaluated. However, the combination of real and discrete features is not without issues. The concatenation of word counts with real-valued features prevents the use of a sparse dictionary representation which is typical for large text corpora. While this is not problematic in settings with small amounts of data, it would introduce memory issues as the dataset grows in size. One possible solution would be to discretize the real-valued vectors as new entries in the sparse matrix, but this might significantly reduce the word embeddings' benefit.

The vector extrema operation appears to be a good alternative to a weighted vector average, especially because of its implicit word weights, but its effectiveness seems domain-dependent. It scored lower than a vector average on the question dataset, especially with few training samples, likely because interrogative words were under-represented in sentence extrema (e.g. 'who' is a general word yet it clearly suggests the question relates to a person). The extrema representation will also grow ineffective as the input text's size increases beyond single sentences since the vector must hold components from many words in order to preserve information of the entire sentence.

Acknowledgments

Funding for this work was provided by FRQNT, NSERC and Nuance Communications, Inc..

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137-1155, 2003.

[2] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. *In Proceedings of the 25th International Conference on Machine Learning*, 2008.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *In Proceedings of Workshop at the International Conference on Learning Representations*, 2013.

[4] Word2Vec. [Online]. Available: https://code.google.com/p/word2vec/. [Accessed October 8th 2014]

[5] E. Huang. Word Representations. [Online]. Available: https://ai.stanford.edu/˜ehhuang. [Accessed October 8th 2014]

[6] E. Huang, R. Socher, C. Manning, and A. Ng. Improving word representations via global context and multiple word prototypes. *In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 2012

[7] Q. Le, T. Mikolov. Distributed Representations of Sentences and Documents. *In Proceedings of the 31st International Conference on Machine Learning*, 2014.

[8] M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[9] X. Li and D. Roth. Learning Question Classifiers: The Role of Semantic Information. *In Proceedings of the 19th International Conference on Computational Linguistics*, 2002.