Document Embedding Strategies for Job Title Classification

Yun Zhu, Faizan Javed, Ozgur Ozturk
DataScience R&D, Careerbuilder, Norcross, GA 30092
{Yun.Zhu, Faizan.Javed, Ozgur.Ozturk}@careerbuilder.com

Abstract

Automatic and accurate classification of items enables numerous downstream applications in many domains. These applications can range from faceted browsing of items to product recommendations and big data analytics. In the online recruitment domain, we refer to classifying job ads to a predefined occupation taxonomy as job title classification. A large-scale job title classification system can power various downstream applications such as query expansion, semantic search, job recommendations and labor market analytics. Such classification systems mostly use Bag-of-Words (BOW) model for document representation and consider only the job titles when classifying job ads. However the BOW model lacks the semantic discrimination capability that is needed to accurately classify job ads when they contain multiple aspects of the job such as the job description, job requirements, company overview and other details. In this paper we explore the applicability of recent advances in the word and document embedding space to the problem of job title classification. We investigate several document embedding approaches and propose a novel customized document embedding strategy for job title classification that addresses the multi-aspect job ad issue. Our experimental results show that incorporating document embedding approaches in a job title classification system improves the classification accuracy on entire job ads compared to approaches based on the BOW model.

Introduction

Many e-commerce and web properties have a need to automatically classify millions of items to thousands of categories with a high level of accuracy. Such large-scale item classification systems have many downstream applications such as product recommendations, faceted search, semantic search and big data analytics. In the online recruitment domain, classification of job ads can power applications such as labor market analytics, job recommendations, and semantic search. We refer to classifying job ads (text documents composed of title and description fields) to predefined or custom occupation categories as job title classification. For automatic job title classification, we developed a machine learning-based, semi-supervised, multi-class job title classification system called Carotene, which has: i) a taxonomy discovery component that leverages clustering techniques to discover job titles from data sets to create a custom job title taxonomy, and ii) a two stage classifier, for coarse-grained and fine-grained categories, that uses an SVM-kNN (k-Nearest Neighbors) classifier cascade to classify the input text to the most appropriate job titles, respectively in root and leaf levels of our custom taxonomy. The coarse-grained classifier assigns the title to one of the 23 top level categories, called SOC-majors, after the Standard Occupational Classification (SOC) system (U.S. Bureau of Labor Statistics 2010). Then the fine-grained classifier chooses from normalized job titles only from the given vertical (SOC-major). In creation of the taxonomy, Lingo3G algorithm (Osinski and Weiss 2005) was used to identify the ideal clusters and their labels which is used as normalized titles.

Bag of words (BOW) representation is commonly used in text classification and it was adopted for job text representation in both stages in previous version of our job classification system. However, BOW lacks semantic relationships so it can not handle synonyms, polysemous words and multi-word expressions. Given the recent success of word and document embedding techniques that provide semantic relations among words and documents, the focus of this paper is improving our system by using semantically rich document representations based on these techniques.

Word and Document Embedding

Distributional Semantic Models (DSMs) build representations of words in high-dimensional vector spaces based on the contexts in which they occur. GloVe (Pennington, Socher, and Manning 2014) applied dimensionality reduction on word co-occurrence counts matrix to get the vector representations corresponding to each word. Neural Network trained DSMs are shown to perform better than GloVe for a range of lexical semantics tasks (Baroni, Dinu, and Kruszewski 2014). Word2vec (W2V) (Mikolov et al. 2013), uses a shallow neural network to produce high dimensional vector representations for words and phrases. Neural network optimizes accuracy of prediction of surrounding words for each word, based on the skip-gram model. The relative placement of these vectors in high dimensional space turns out to preserve some semantic relationships of corresponding words. A W2V vector represents a single word or short phrase, whereas we need vector representations of job titles. (Le and Mikolov 2014) proposed $paragraph\ vectors$, to

represent whole paragraphs or documents produced by Neural Networks whose objective is to predict words in the document. They found it to have significantly higher accuracy in their sentiment analysis and information retrieval tests compared to the state of the art. We evaluated it in job title classification as well, but it did not improve our results.

In Carotene we replaced BOW with word and document embedding based representations for job ad titles and used corresponding similarity measures in the kNN classification component. Our comprehensive performance comparisons show improvement in multiclass classification accuracy.

Related Work

Enriching vectors and semantic kernels are the two most commonly used semantic enrichment techniques for text classification. In the enriching vectors approach (Huang et al. 2012), a document representation is enriched with some or all of the following: hypernyms, synonyms and related concepts. Semantic kernels (Wang and Domeniconi 2008) leverage a semantic proximity matrix to transform document representations into linearly separable semantic representations. (Albitar and Fournier 2014), compared their success on medical text classification, and found that enriching vectors performed better whereas semantic kernels introduced noise in document representations, and degraded performance. Semantic kernels are usually used with SVMs and applied before classifier training time. However, (Lu and Zhai 2006) details a semantic kernel approach that actually improved the performance of the SVM algorithm when the dimensionality of the input feature space is large and training data is scarce. For job title classification, LinkedIn uses a phrase-based classification system that relies on the near-sufficiency property of short text (Bekkerman and Gavish 2011). Near-sufficiency of short text means short documents usually have higher signal to noise ratio which increase document classification accuracy. Also in our experiments, job titles alone gave more accurate classification results. A semantic enrichment approach to job title classification is discussed in (Malherbe, Cataldi, and Ballatore 2015). This approach semantically enriches job categories with contextually relevant terms derived from a corpus of job ads. A field-to-field similarity matching approach then matches job ads to job categories.

Methods

Basic Definition and Notation

Document Document concept, in this work, refers to a number of consecutive words: $D = \langle w_1, w_2, ..., w_n \rangle$.

Query Job Ad A query job ad is formed of title and description documents, i.e., $J_q = \langle D_{qt}, D_{qd} \rangle$, where description is usually a longer document than title.

Training data for kNN The training data for our kNN classifier contain pairs of a raw job title document, D_{rt}, and the matching normalized title document D_{nt}. $J_{tr} = \langle D_{rt}, D_{nt} \rangle$ Raw titles are titles from real world job ads, and have many-to-one mapping to normalized titles. Table 1 shows a sample from the training data.

Raw Title	Normalized Title
Software Developer III	Software Engineer
Software Developer Consultant	Software Engineer
C++ Linux Embedded SW Eng	Software Engineer
Mobile Applications Developer	Mobile SW Engineer
Web Integration Intern	Integration Engineer
RN Full Time pm Shift	Registered Nurse
Registered Nurse PCU PRN	Registered Nurse

Table 1: Sample Training Data

Given a query job ad, the kNN classifier is supposed to find the similar training data examples (neighbors) and then classify the query according to the normalized titles of those retrieved neighbors. While most queries will have a non-empty description, the training data doesn't have one. Given this heterogeneity between query and training data, one major challenge is to find a common representation that would have meaningful distances for the kNN classifier. The rest of this section covers our approaches to tackle this problem.

Baseline

In the previous version of Carotene, we created document clusters using cosine distance on BOW representation of training documents. Our baseline is the fine-grained classifier component which used open-source search engine Lucene to implement a kNN classifier.

D2V: W2V-based Document Embedding

General Framework W2V models give a dense vector representation for every single word, while our job classification system operates on the level of documents, i.e., title, and description. In this work, we adopt a weighted average framework to generate document vectors. Given a W2V model and a document $D = \langle w_1, w_2, ..., w_n \rangle$, the document vector representation is calculated as:

$$V(D) = \frac{\sum_{w_i \in D} \alpha_i \cdot w2v(w_i)}{\sum \alpha_i}$$

where $w2v(w_i)$ is the vector generated by the W2V model for word w_i and α_i the weight assigned to w_i in forming the document vector. Several weighting strategies employed in our work to determine the weights, α_i, will be discussed in next section. Given the vector representation of all 4 documents; raw title (D_{rt}), normalized title (D_{nt}), query title (D_{qt}) and query description (D_{qd}), we form the final vector representation for training data and query jobs as follows:

$$V(J_{tr}) = \beta_1 \cdot V(D_{nt}) + (1 - \beta_1) \cdot V(D_{rt}) \quad (1)$$

$$V(J_q) = \beta_2 \cdot V(D_{qt}) + (1 - \beta_2) \cdot V(D_{qd}) \quad (2)$$

where $\beta_1, \beta_2 \in [0, 1]$ are the balance factors controlling how much each document contributes to the final vectors.

Uniform weighting The most straightforward strategy is to treat each word in a document equally, i.e.,

$$\alpha_i = 1, \quad 1 \leq i \leq n \quad (3)$$

This strategy can be applied to all 4 documents.

Document type	Weighting strategy
D_{rt}, D_{nt}, D_{qt}	Uniform, Frequency
D_{qd}	Uniform, Tf-idf, Consistency

Table 2: Summary of word weighting strategies

Frequency based weighting Although words in short documents (D_{rt}, D_{nt} and D_{qt}) are supposed to be more informative on average than words in the long job description document (D_{qd}), we still see words that are irrelevant to job categorization such as location (e.g, "New York City", "LA") and Salary (e.g., "40k-60k"). Word frequency is an good option to reduce the contribution of such irrelevant words, i.e.,

$$\alpha_i = log(freq(w_i)) \tag{4}$$

where $freq(w_i)$ is number of occurrences of word w_i in our data set of titles of 40 million job postings. Smoothing with a log scale gave us better results.

Tf-idf weighting for description Tf-idf is a common weighting strategy in text classification tasks. We use tf-idf weights only for D_{qd}, since term frequency (tf) is always 1 in short documents. In a title document, we observed that words rarely repeat, and we remove such duplicates in our preprocessing. Formally,

$$\alpha_{iD} = tfidf(w_i, D) \tag{5}$$

tf-idf weight of word w_i for document D, given the descriptions in the 40 million job dataset.

Consistency weighting for description In a query job $J_q = \langle D_{qt}, D_{qd} \rangle$, title ($D_{qt}$) and description ($D_{qd}$) are referring to the same job opportunity. So it's reasonable to assume the relevant words in D_{qt} should be semantically consistent to words in D_{qt}. With this assumption, we weight the words in D_{qd} by the similarity of their W2V vector to the corresponding title vector $V(D_{qt})$:

$$\alpha_i = sim(w2v(w_i), V(D_{qt})) \tag{6}$$

where $sim(v_1, v_2)$ is a similarity measure between two vectors. We adopted cosine similarity for our implementation.

Table 2 summarizes all the weighting strategies that will be tested in our work. There are 2 options for D_{rt}, D_{nt}, D_{qt} and 3 for D_{qd} which leads to 24 combinations in total as shown in Table 3 in the Experiments section.

Denoising Job descriptions usually contain words that are not category specific, such as equal employment opportunity (EEO) statements, benefits related content. Removing such noise can increase classification accuracy. For this purpose, we ranked the words in the description by the weights, assigned by above weighting strategies for job descriptions, and discarded words with low weights, keeping N words with highest weight for each job description.

Vectors for Training Data and Query Jobs

Title documents and sentences of description documents are transformed to D2V vectors. Vector for job description

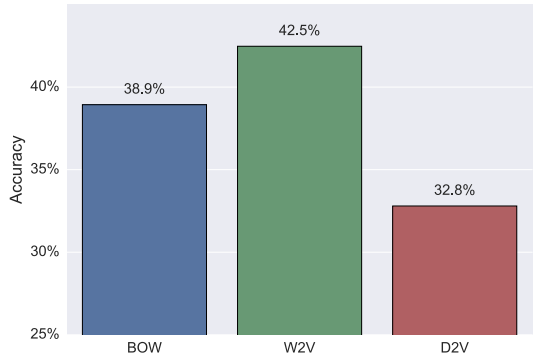

Figure 1: Accuracy of BOW, W2V and D2V based approaches

(D_{qd}) can be formed by aggregating vectors of its sentences in either uniform or consistency weighting strategies, i.e.,

$$V_{d2v}(D_{qd}) = \frac{\sum_{S \in D_{qd}} \alpha_S \cdot d2v(S)}{\sum \alpha_S}$$

where value of α_S is determined by weighting strategies and the similarity to title vector if needed. With description vector calculated, the final vectors for training data and query jobs can be formed in the same way as W2V-based approaches, i.e.,

$$V(J_{tr}) = \beta_1 \cdot d2v(D_{nt}) + (1 - \beta_1) \cdot d2v(D_{rt})$$
$$V(J_q) = \beta_2 \cdot d2v(D_{qt}) + (1 - \beta_2) \cdot V_{d2v}(D_{qd})$$

where $d2v(D)$ is the vector generated by doc2vec for any document D and β_1, β_2 balance factors

Experiments

Evaluation Data

We evaluate the performance of various document embedding methods in kNN job classification task, i.e., the fine-grained classifier in our system. Evaluation data includes 1667 human-labeled query jobs and training data contains 835804 titles that fall into 5425 predefined categories. Overall accuracy is used as the performance metric for all experiments.

BOW vs. W2V vs. D2V

To get a performance overview over different document embedding methods, we first compare the basic W2V and D2V embedding approaches with the baseline. Specifically, 1) Uniform weighting strategy for all documents in W2V based approaches. 2) Basic D2V approach means raw vectors by doc2vec model are used as representation for raw and normalized title in training data and job title in query. In addition, D_{rt}, D_{nt} contribute evenly in forming training data vectors and so do D_{qt}, D_{qd} for query, i.e., $\beta_1 = \beta_2 = 0.5$. Our baseline is the kNN classifier using BOW (unigram + bigram) as described in section 2. Number of neighbors (k) for all three models in this comparison is set to 20 which is

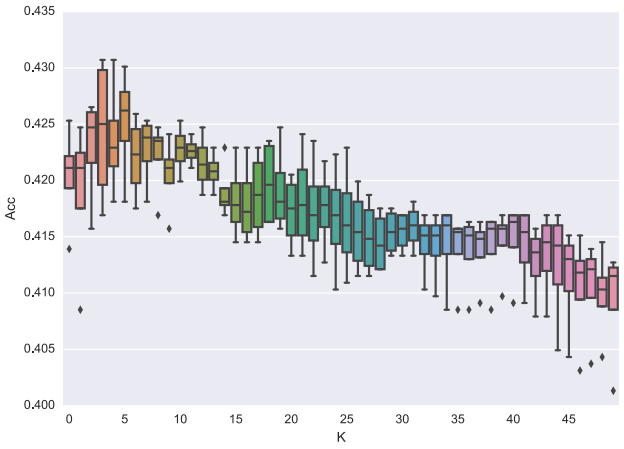

Figure 2: Accuracy with different k values

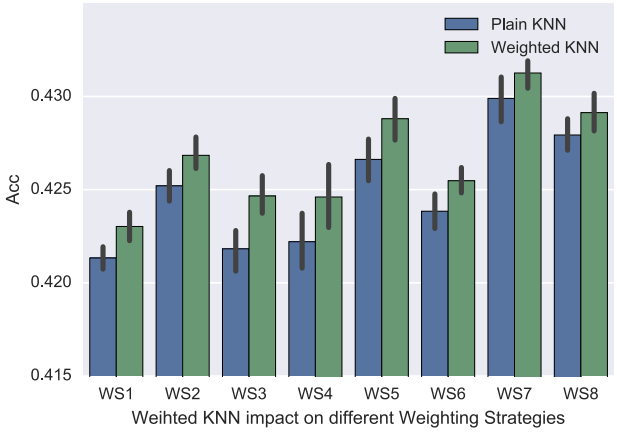

Figure 3: Impact of weighted kNN

the practical optimal based on experiments in our previous work.

As shown in Figure 1, BOW (baseline), W2V and D2V score at 38.9%, 42.5% and 32.8% respectively. W2V based approaches significantly outperform the BOW. However D2V performs much worse than the other two, which is similar to the results obtained in (Dai, Olah, and Le 2015) and (Partalas et al. 2015) that D2V models don't necessarily work well in all cases. So in the rest experiments, we don't include D2V model for comparison.

KNN parameter tunings

Size of Neighborhood As $k = 20$ was the optimal number of neighbors picked empirically in our previous system, we also investigated how value of (k) impacts the performance in W2V based methods. Figure 2 shows the performance of all weighting strategies with varying k from 1 to 50. $w_1 = w_2 = 0.5$ stay the same as in previous experiment. The observation shows that the optimal k for this particular parameter setting is in range between 3 and 14 which leads to 42+% average accuracy. Accuracy begins to drop notably with $k > 15$. Given previous optimal $k = 20$ and W2V based approaches perform better than baseline for $k = 20$, a smaller optimal k could be interpreted as an evidence of stronger capability of W2V based approaches in finding correct neighbors. In the rest of the experiments, we include all accuracy numbers for $k \in [3, 14]$ if not specified otherwise.

Weighted KNN Weighted kNN is a common option in practice to increase the accuracy of kNN. The difference of weighted kNN and original kNN is how to decide the final label after finding the neighbors. Specifically, kNN originally uses simple majority vote, i.e., the most frequent label among neighbors is selected to classify a query data. On the other hand, weighted kNN assigns neighbors' votes weights proportional by their similarity to the query data, i.e., the more similar the neighbor is to query, the higher its label is weighted in final decision. In our weighted kNN implementation, we use cosine distances to weight the neighbors' votes. To see the impact of weighted kNN, tested 8 weighting strategy combination for D_{rt}, D_{nt}, D_{qt}, i.e., $\{\text{Uniform}, \text{Frequency}\} \times \{\text{Uniform}, \text{Frequency}\} \times \{\text{Uniform}, \text{Frequency}\}$ with "Uniform" setting for D_{qd}. In Figure 3, each pair of bars indicates one weighting strategy setting except with or without weighted voting. Obviously in all settings, weighted kNN helps slightly in increasing the accuracy of the model. We will keep using weighted kNN in the rest experiments.

Influence of Balance Factors

In equation (1) and (2), two balance factors (β_1 and β_2) are involved in forming training data and query job vectors. In order to understand the importance of different documents, we explore the best combination of β_1 and β_2 in various parameter settings. Figure 4 shows the results with one particular setting where uniform weighting is applied for all documents and $k = 10$. The orange colored squares indicates performance improvement over the baseline (38.9%), the darker the better, and blue indicates worse. In this particular parameter setting, the optimal combination is $\beta_1 = 0.8$ and $\beta_2 = 0.7$ which achieves the highest accuracy as 43.7%. We plotted the same heat map with other parameter setting and found the optimal combination is in the area of $\beta_1, \beta_2 \in [0.6, 0.8]$. In other words, normalized title D_{nt} is more important than raw title (D_{rt}) in training data, and query job title D_{qt} is more important than query job description (D_{qd}), as we expected.

Effect of Weighting Strategies

As summarized in Table 2, 24 different weighting strategy configurations are applicable in total. Table 3 on next page, summarizes the performance of these strategies with β_1, β_2 and k varying in optimal ranges as suggested in previous experiments. The first 4 columns specify the weighting strategies for each document and last 2 columns present the mean$\pm$std and maximum of accuracy achieved. Table 3 is sorted by mean accuracy. In addition, highest mean and max accuracy are underlined and top 1 is marked with bold font. Apparently weighting strategies for longer job description

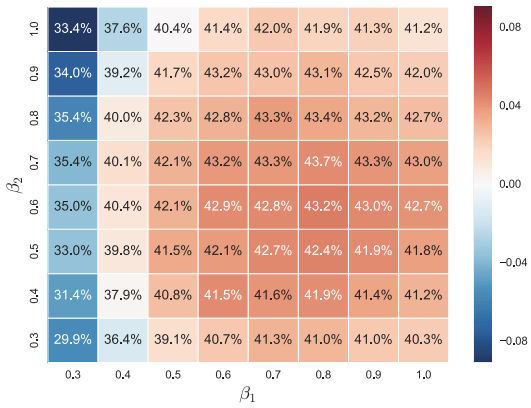

Figure 4: Accuracy of different β_1, β_2

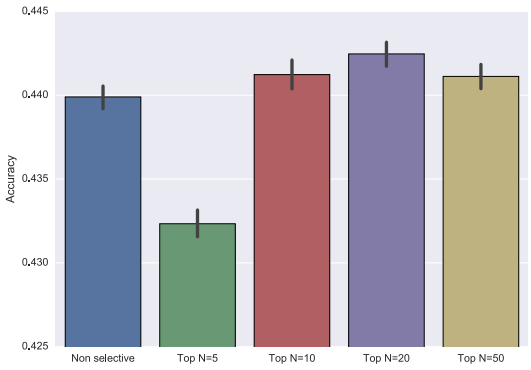

Figure 5: Accuracy of Top N

D_{qd} are beneficial as both tf-idf and consistency configurations consistently outperform uniform ones. On the other hand, frequency based weighting doesn't help much for D_{qt} and D_{rt} as the uniform configurations take the top positions. This is reasonable as words in short text like titles (D_{qt}, D_{rt}) are supposed to be very informative and leave very limited noise that could be reduced by any weighting strategy.

Selective Words for Document Embedding

As the weighting strategies for description in evaluation data (ED) provide the weight scores of each word when calculating the weighted vector average. Based on the same assumption that words with higher weight are semantically more relevant to the corresponding job post, an interesting usage of this score is to rank the words in description and discard the least relevant words. We implemented a TopN strategy that picks N words with highest tf-idf and Consistency Scores for final job vectorization. With other parameters set in suggested optimal range ($k \in [3, 14], \beta_1, \beta_2 \in [0.6, 0.8]$), Figure 5 shows the accuracy summaries of TopN with $N = [5, 10, 20, 50]$ compared to non-selective approaches with tf-idf and Consistency weighting strategies.

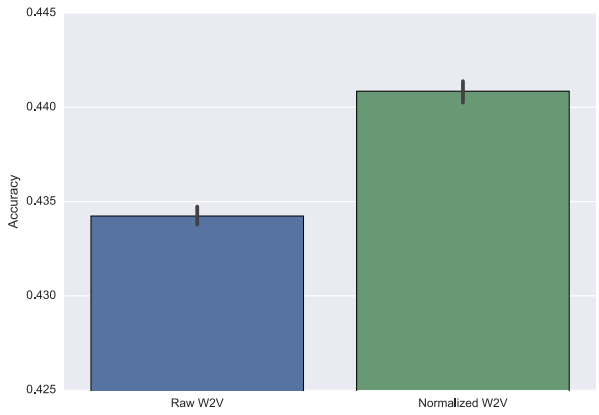

Figure 6: Accuracy comparison of raw and normalized W2V model

Normalization of W2V vectors

The word vectors generated by W2V are of different norms. In the last experiment, we are interested in finding out if and how much the normalization of all word vector will affect the results. We repeat the previous experiment again except with all word vectors normalized immediately after the W2V is trained. As Figure 6 shows, normalized word vectors generate better quality document embedding which leads to higher overall accuracy.

Conclusion

In this work, document embedding with different word weighting strategies are proposed as a replacement of current bag-of-words (BOW) representation in the kNN component of our current job classification system. According to the experiment results, both tf-idf and consistency word weighting strategies for long text (job description) improve the performance significantly while short text (job title, category label) are not sensitive to word weighting. In addition, job document embedding using only top N selective words by weighting score is the best among all configurations we tested. Experiments also verify the effectiveness of weighted kNN, normalized word vectors. Overall, this work provides a good practical experience and guide for applications involving word/document embedding. Interesting future work include 1) applying other alternative distance metrics like the word mover distance for similarity measure in kNN, and 2) job enrichment with skill terms using our internal skills extraction service.

References

Albitar, S., E. B., and Fournier, S. 2014. Semantic enrichments in text supervised classification: application to medical domain. *Florida Artificial Intelligence Research Society Conference*.

Baroni, M.; Dinu, G.; and Kruszewski, G. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, 238–247.

row #	D_{qt}	D_{qd}	D_{nt}	D_{rt}	Accuracy (mean±std)	Accuracy (max)
1	Uniform	Tf-idf	Frequency	Uniform	***44.01±0.4%***	*44.75%*
2	Uniform	Consistency	Frequency	Uniform	*43.97±0.5%*	***45.11%***
3	Uniform	Tf-idf	Uniform	Uniform	*43.67±0.4%*	44.45%
4	Frequency	Tf-idf	Frequency	Uniform	*43.57±0.5%*	*44.99%*
5	Uniform	Tf-idf	Uniform	Frequency	*43.54±0.2%*	44.09%
6	Uniform	Tf-idf	Frequency	Frequency	43.54±0.4%	44.09%
7	Frequency	Tf-idf	Uniform	Frequency	43.54±0.4%	*44.81%*
8	Frequency	Consistency	Frequency	Uniform	43.52±0.7%	*44.57%*
9	Uniform	Consistency	Uniform	Uniform	43.49±0.4%	44.27%
10	Frequency	Tf-idf	Uniform	Uniform	43.48±0.4%	44.33%
11	Uniform	Consistency	Frequency	Frequency	43.47±0.4%	44.27%
12	Uniform	Uniform	Frequency	Uniform	43.40±0.4%	44.27%
13	Frequency	Consistency	Frequency	Frequency	43.28±0.6%	44.39%
14	Frequency	Tf-idf	Frequency	Frequency	43.22±0.4%	44.03%
15	Uniform	Consistency	Uniform	Frequency	43.20±0.3%	43.85%
16	Uniform	Uniform	Frequency	Frequency	43.07±0.5%	43.97%
17	Uniform	Uniform	Uniform	Frequency	42.91±0.3%	43.61%
18	Uniform	Uniform	Uniform	Uniform	42.90±0.4%	43.91%
19	Frequency	Uniform	Frequency	Uniform	42.88±0.5%	43.79%
20	Frequency	Consistency	Uniform	Frequency	42.88±0.5%	43.97%
21	Frequency	Consistency	Uniform	Uniform	42.81±0.4%	43.55%
22	Frequency	Uniform	Uniform	Frequency	42.72±0.4%	43.61%
23	Frequency	Uniform	Frequency	Frequency	42.67±0.3%	43.43%
24	Frequency	Uniform	Uniform	Uniform	42.61±0.4%	43.31%

Table 3: Accuracy of different weighting strategies

Bekkerman, R., and Gavish, M. 2011. High-precision phrase-based document classification on a modern scale. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, 231–239. New York, NY, USA: ACM.

Dai, A. M.; Olah, C.; and Le, Q. V. 2015. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop*.

Huang, L.; Milne, D.; Frank, E.; and Witten, I. H. 2012. Learning a concept-based document similarity measure. *J. Am. Soc. Inf. Sci. Technol.* 63(8):1593–1608.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In Jebara, T., and Xing, E. P., eds., *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 1188–1196. JMLR Workshop and Conference Proceedings.

Lu, X., Z. B. V. A., and Zhai, C. 2006. Enhancing text categorization with semantic-enriched representation and training data augmentation. *J Am Med Inform Assoc.*

Malherbe, E.; Cataldi, M.; and Ballatore, A. 2015. Bringing order to the job market: Efficient job offer categorization in e-recruitment. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 1101–1104. New York, NY, USA: ACM.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 3111–3119.

Osinski, S., and Weiss, D. 2005. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems* 20(3):48–54.

Partalas, I.; Kosmopoulos, A.; Baskiotis, N.; Artières, T.; Paliouras, G.; Gaussier, É.; Androutsopoulos, I.; Amini, M.; and Gallinari, P. 2015. LSHTC: A benchmark for large-scale text classification. *CoRR* abs/1503.08581.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–43.

U.S. Bureau of Labor Statistics. 2010. SOC Major Groups. http://www.bls.gov/soc/major_groups.htm. [Online; accessed 20-November-2016].

Wang, P., and Domeniconi, C. 2008. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, 713–721. New York, NY, USA: ACM.