2019 IEEE International Conference on Big Data (Big Data)

DeepCarotene - Job Title Classification with Multi-stream Convolutional Neural Network

Jingya Wang[†*], Kareem Abdelfatah[‡*] , Mohammed Korayem and Janani Balaji
CareerBuilder
Norcross, Georgia, USA
{jingya.wang, kareem.abdelfatah, mohammed.korayem and janani.balaji}@careerbuilder.com
[†]Indiana University Bloomington, Bloomington, Indiana
[‡]University of South Carolina, Columbia, South Carolina

***Abstract*—In online recruitment, job title classification is a fundamental task that enables several downstream applications like job recommendation and ranking for job search. A special case of multi-class text classification, the job title classification problem takes as input two components from a job posting - a short job title and a lengthier job description, and normalizes the raw job title into its closest match from the given taxonomy. Typically, the job title, though shorter in length, contains more targeted signals than the job description, that can contain additional information irrelevant to the context. On the other hand, the job description often provides valuable information that helps steer the classification model towards choosing the best match. Achieving a balance between the two components is not a trivial task. In this paper, we propose a multi-stream CNN based model for job title classification, that learns semantic features on both character and word level. We collected about 15 million data points from one of the largest online job boards, Careerbuilder, to train the model. Due to the universal problem of getting massive labeled data, we adopt a weakly supervised method to efficiently generate noisy labels for this large data set. Compared with the current state-of-the-art job title classification systems, the proposed model, DeepCarotene, shows a significant improvement in performance. This model provides a new direction of CNN based end-to-end approach for job title classification.**

I. Introduction

In the online recruitment domain, classifying job titles into pre-defined occupation categories is a fundamental task that facilitates many downstream applications such as matching jobs with resumes, job ranking, job recommendation as well as labor market analysis. The job titles that need to be classified can come either from candidate resumes, or from job posting of companies as illustrated in Figure 1. In simple terms, job title classification can be thought of as an application of text classification, which is well studied in the natural language processing (NLP) literature. The key difference here is in the the length of the input - the job title is only a few words long in contrast to the lengthier sentences and paragraphs, which are more common in text classification problems. A job title usually consists of a group of words representing the professional field or main skill of the job, eg., "human resource", along with the level of the job, eg., "specialist" or "manager". It is a short term, but not necessarily extendable to be part of a sentence, as addressed in [12]. These differences lead to two major concerns while applying standard text classification methods to job title classification. First, there's not enough context to train a sequence model like the long short-term memory (LSTM) [25] [29] when the input is usually only a few words. Second, unlike in a paragraph, every single word in the job title is more valuable and requires careful interpretation.

(a) Job posting with title "RN - Post Anesthesia Care Unit (PACU)" is classified with code "29.66", which represents "Post-Anesthesia Care unit (PACU) Nurse" which is a subcategory in SOC major group "29" for "Healthcare Practitioners and Technical".

(b) The title "WI - Wauwatosa - Versona Part-time Sales Associate" includes location, company name and the feature of "part-time". The classifier predicts this job posting in a general category "41.1" representing "Retail Sales Associate" in SOC major group "41" for "Sales and Related".

Fig. 1. Examples of categorizing the job posting. The output is a code of pre-defined job title categories. The first two digits before "." represents major group code.

Previous works on job title classification have only focused on applying a select a subset of traditional text classification methods such as hierarchical SVM [31], linear classifier [30] and term frequency based method [12]. Inspired by the recent continuous success of deep learning on NLP and computer vision, and the discovery that character level convolutional neural network (CNN) [15] is sufficient in text classification problems, in this paper, we propose a multi-stream CNN based model for job title classification, considering both characters and words.

The idea of employing auxiliary context for job title classification, in addition to exploiting the potential of each valuable word in the job title, was proposed in [17] and [4]. While a job

[*] Both authors contributed equally to this research. This work was done when they were at Careerbuilder

978-1-7281-0858-2/19/31.00©2019IEEE
978-1-7281-0858-2/19/$31.00 ©2019 IEEE

TABLE I
SOC MAJOR DEFINITION

Major Id	Job title
41	Sales and Related
11	Management
27	Arts Design Entertainment Sports and Media
47	Construction and Extraction
29	Healthcare Practitioners and Technical
13	Business and Financial Operations
35	Food Preparation and Serving Related
25	Education Training and Library
15	Computer and Mathematical
17	Architecture and Engineering
19	Life Physical and Social Science
21	Community and Social Service
23	Legal
31	Healthcare Support
33	Protective Service
39	Personal Care and Service
43	Office and Administrative Support
45	Farming Fishing and Forestry
49	Installation Maintenance and Repair
51	Production
53	Transportation and Material Moving
55	Military Specific
37	Building and Grounds Cleaning and Maintenance

title is usually of limited length, the job description consists of sentences, sometimes even paragraphs of text, providing more detailed information such as - a brief introduction of the company, general responsibility of the position, and sometimes location, requirements and benefits. When job title is not available, or when we want to consider additional information to augment the title classification task, job description is the most discernible choice. In our model, we use the job title along with the job description as inputs for the job title classification task.

Based on the 2018 Standard Occupational Classification (SOC) System published by United States department of Labor [1], there are 23 major groups of job titles as given in Table I. Our work adopts this SOC major system. Additionally, similar to other job title classification works [2], [21], we also add subcategories to these major groups to create a hierarchical taxonomy. Accordingly, our taxonomy consists of two components - a two digit SOC Major code that forms the broad classification layer and a variable major group code (2-4 digits) that gives a more granular classification. We propose a model that explicitly learns the loss of SOC major to enforce the performance on major level groups.

Due to lack of training data, our previous job title classifier, Carotene [34], used a hierarchical KNN based system to predict the coarse label on SOC major first, and the fine-grained category only within this SOC major. In this paper, we propose an end-to-end system to predict the SOC major and the subcategories in one network. This helps to avoid the problem of error propagation between levels in hierarchical systems [21], where once the first step - in this case the prediction of SOC major, produces any error, there's no chance to correct it in lower level prediction.

[1]https://www.bls.gov/soc/2018/major_groups.htm

Many state-of-the-art models in text classification are based on deep learning techniques which requires massive sets of training data. However, obtaining traditional hand-labeled training data from domain experts is very expensive and hence, procuring informative training data becomes a new challenge. Active learning helps handle large scale labeling tasks by selecting data points near decision boundaries before sending them to domain experts for labeling. When there's only a limited set of labeled data available, semi-supervision learns to make use of the unlabeled data. When a pre-trained model from a related task is available, transfer learning methods fine tune this model for a new task with a smaller labeled data set. In practice, due to the cost, time or the lack of domain experts, collecting even a small set of high quality data is not easy. In such scenarios, weak supervision methods can be employed to best utilize lower-quality labels and labels at a higher abstraction level. For job title classification problems, due to lack of domain experts, collecting large scale job title labels is very expensive and the quality is hard to guarantee. As one of the largest job boards, Careerbuilder maintains millions of job postings in a large variety of categories. To build our next generation job title classifier, we use a weakly supervised model to obtain large scale training data with noisy labels from Carotene, our existing model for job title classification.

If we consider job title classification as a matching problem to match a query job title to one of the pre-defined categories, these text matching problems usually includes two main forms: local matching and semantic matching. Local matching focuses on matching the query terms based on spelling and pattern. It helps with misspelling, punctuation and emoticons. On the other hand, semantic matching, or more generally addressed as distributed matching, is to project the query terms and the candidates to a higher dimensional space and perform matching based on the distributed representation. The architecture of our proposed model is inspired by [20] that achieves better performance by jointly learning two neural networks for both of local matching on character level and semantic matching on short terms and sentences level.

Convolutional neural network achieves remarkable performance on many natural language processing (NLP) problems such as sentiment analysis [32], sentence classification [14] and question answering [5]. The convolution layer learns to map the input matrix to higher dimensional space with different size and value of filters. This process is considered as extracting semantic features of the input layer. Our model is built with multiple convolution layers to learn semantic feature from input embedding.

In this paper, We show an end-to-end CNN based model for job title classification with both job title and job description. This model is jointly trained with character stream for local matching and word stream for semantic matching. A multi-task loss function is applied to enforce the performance of SOC major prediction. We also propose a weakly-supervised system to generate labeled data from noisy input. To summarize, the contributions of the paper can be listed as follows:

1) Propose a multi-stream CNN based model for end-to-

end job title classification
2) Define a multi-task loss function to minimize error propagation
3) Develop a weakly-supervised system to obtain labeled data from noisy input

The rest of the paper is organized as follows: Section II glosses over the related work in the field and Section III describes the proposed system. The results are explained in Section IV, while Section V concludes the paper with scope for future work.

II. Related Works

Job title classification includes concepts from several fields such as word embedding document embedding, sentence classification and semantic search. Given below is a brief survey of the state of the art in each.

A. Word Embedding

The traditional way to represent words and documents in vectors is to have a sparse one-hot vector for each word, with the dimension of each vector being the size of the vocabulary. This is however not a space-efficient choice, especially when dealing with a large vocabulary. [8] introduced the word-hashing method to represent words in form of letter tri-grams. Each word consists of a sequence of a three-letter combination, thereby enabling words with similar spelling to be represented in vectors close to each other. Tri-gram representation [9] is also broadly adopted in characterizing sentences. Representing sentences in terms of three-words combination could help in mapping sentences to lower dimensional vectors, while helping the model to consider combination of words rather than treating words independently. However, both the methods above suffer from the problem of sparsity [7]. In our method, we simply refer to the index of the characters and words in a look up table to represent words and documents, while the actual capturing of the semantics happens after the input layer. Given that we are applying convolutional neural network to learn the high level features, this low dimensional representation is adequate and efficient.

B. Learning Semantic Representation

After obtaining the vector representation, we are ready to learn the semantics of words and documents. Semantic word embedding works well with problem specific measurements. In [6], Guo et al., use doc2vec [16] and GloVe [23] to encode the semantic relationship between words and documents, and finally use a cosine similarity score to match a query word to a word in a document. Similarly, [13] derive a similarity score for various length of documents based on cosine similarity of their word2vec [19] representations. On the other hand, many works explore different ways to learn task specific semantics. In [18], a multi-task deep neural network is designed to learn a classification and matching problem alternatively to achieve vector representation when training data is limited. Words are first represented in letter tri-gram by a fixed word-hashing layer [8]. The semantic representation is learnt from this letter tri-gram representation with one fully connected layer. In our work, we propose that a convolutional neural network could learn a better representation by mapping the input vector to non-linear space.

C. Deep Learning for Semantic Feature Extraction

Deep learning architecture is successfully applied to learn high level semantic features for text [3], [11], [32]. The work in [26] takes text as input directly to implicitly learn the letter n-gram based word n-gram representation in convolution layer with filters in a variety of sizes. In this end-to-end architecture, another convolution layer is utilized to learn the semantic embedding and a max-pooling layer is applied to extract the saliency. A fully connected layer finally maps it to a vector representation. [14] also applies the convolution and max-pooling architecture while representing the input words statically. [7] and [33] use a variation of recurrent neural network. A semi-supervised CNN based text classification system is introduced in [10] to learn text representation from unlabeled data set before training a supervised CNN model for classification. These models show great improvements in document understanding related problems but are not directly applicable to word representation problems.

D. Weak Supervision

Weak supervision is a general concept referring to generating training data with - cheaper, lower-quality labels from non-experts; labels from higher level supervision over unlabeled data from subject matter experts (SME); and as in this project, labels from one or more noisy pre-trained models [1]. Using heuristic functions to get noisy labels over multiple domain-specific primitives, [27] learns a generative model to predict probabilistic labels. Extended from this work, [24] introduces a weak supervision labeling system, Snorkel, to create large training data set automatically. The proposed method takes the probabilistic output from the generative model to train a discriminate model with noise-aware loss function.

E. Job Title Classification

The specific problem of job title classification is not commonly addressed in academia. There's even less work using deep learning and end-to-end method. [21] takes word2vec representation as input and uses convolution with a max-pooling architecture. This work, however, is not fine grained and doesn't go further beyond the 23 Standard Occupation Classification (SOC major) level.

III. Proposed Model

In this paper, we use a multi-stream CNN based model for job title classification. We extract three levels of features from the input text - characters, short terms and sentences. CharCNN builds a feature representation for characters in the job title while WordCNN has a similar structure to extract features for job title and job descriptions.

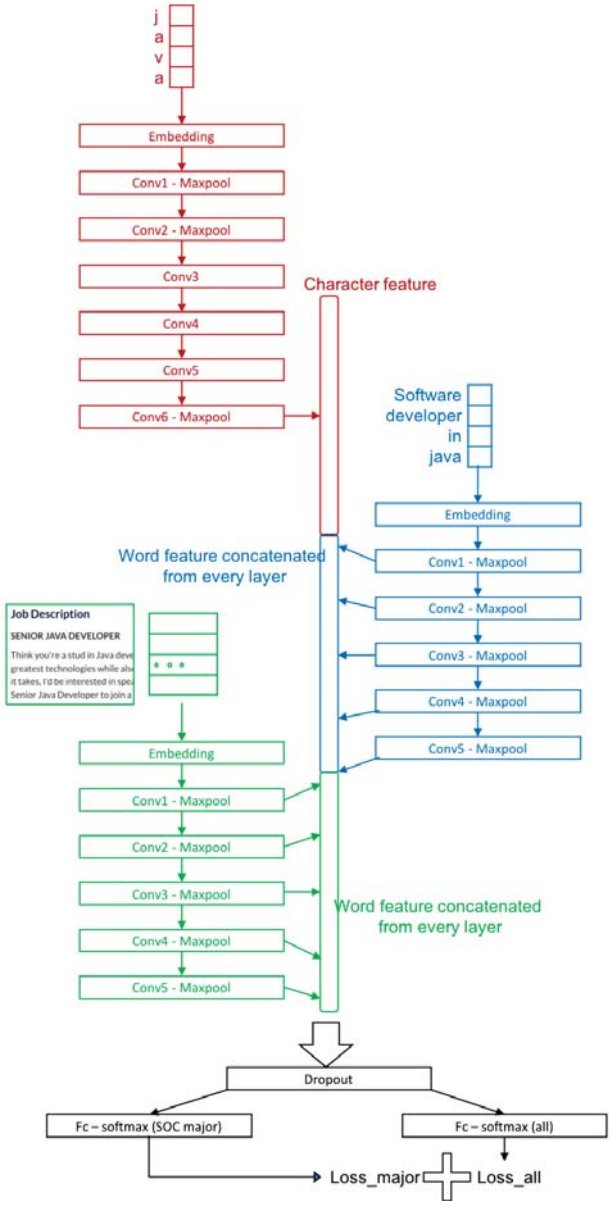

Fig. 2. Our model has three components - charCNN for job title (in red), wordCNN for job title (in blue) and wordCNN for job descriptions (in green). The output of the last convolution layer in charCNN is used as the character level feature while wordCNN takes the concatenation of all max-pooling output as word level features. The two wordCNN for job title and job descriptions are independent. All these three networks are learned simultaneously during training process (illustrated in black at the bottom).

A. Learning Domain Specific Embedding

In online job posting domain, it's possible to create a dictionary of limited size for the vocabulary using all the training samples. For word representation, we use the index in this dictionary for each word in job title and job description to build the vector representation. Compared to the very sparse representation of using a vector the size of the entire dictionary for words and a matrix for short terms or sentences by stacking these high dimensional vectors, our index based representation is more space efficient. With this vector of input text, we employ an embedding layer to learn task specific representation. In our end-to-end CNN based system, we learn this embedding layer together with the job title classifier. Similar words used in job titles under the same job category are supposed to have embedding vector closer to each other.

B. CharCNN

The CharCNN for job title learns to extract features of local pattern on character level. We use a one-hot representation for character since the vocabulary size $N_C = 70$ is not large at all. Let ${x_{char}}^i$ be the N_C-dimensional one-hot vector corresponding to the i^{th} character in each job title. The representation of job title $x_{char} \in R^{lN_C}$ is fixed to the input size of $l = 128$ characters and is truncated if longer or padded with zero if shorter.

According to the architecture illustrated in Figure 2, this input vector representation is reshaped in embedding layer, and then passed through 6 convolution layers. The first two convolution layers have 128 filters with kernel size 7 to capture features across more consecutive characters while the other convolution layers take the same number of filters, but with kernel size 3. For example, the first convolution layer c_1 takes input x_{char}. By applying 128 filters $w_k \in w$ where $k \in 1, 2, ..., 128$, $w_k \in R^{N_C * h}$ and $h = 7$, the convolution operation $c_1^i = f_{conv}(w * x_{char;i:i+h-1} + b)$ projects every 7 consecutive characters to a 128-dimensional feature vector. The first two and last one convolution layers are followed by max-pooling with kernel size 1*3 to reduce computation space and extract the most salient local features. The output of last max-pooling layer is flattened to be the character feature vector represented as V_c.

C. WordCNN

The WordCNN is a convolutional neural network, used to extract semantic features for short terms of job title and long paragraphs of job description. WordCNN takes the job title truncated or padded to fixed length of input in $C_{title} = 16$ words or $C_{desc} = 512$ words for job descriptions. Given the vocabulary in size N_W, for each input vector x_{word}, $x_{word}^i \in \{0, 1, ..., N_W - 1\}$ represents the index of the i^{th} term in job title or job description. The network architecture is the same in both cases as in Figure 2. The index vector x_{word} is projected to a 300-dimensional vectors through embedding layer. The output of embedding layer is then passed through 5 convolution layers. Every layer has 128 filters in size of $n = [1, 2, 3, 4, 5]$ from lower to top layers with stride 1. Next,

after every convolution layer we apply ReLU layer to add non-linearity. To extract the most significant features along each filter and reduce the space of parameters, each ReLU layer is then followed by a max-pooling layer with kernel size $(C_{title} - n + 1) * 1$ for job title and $(C_{desc} - n + 1) * 1$ for job description. The wordCNN feature for job title is the flattened output vector of the last max-pooling layer represented as V_w and so for job description as V_s.

D. Multi-stream CNN Model

We jointly train all of the above three CNN models to extract features on characters, short terms, and paragraphs. We use "charCNN" for job title matching on character level and produce character features V_c. Meanwhile, "wordCNN" is learned for job titles and job descriptions to produce word features V_w and V_s respectively. All the three streams are learned simultaneously in our end-to-end job title classification system. The final feature to represent our input job title and description is obtained by concatenating features from all three streams $V_f = concat(V_c, V_s, V_w)$.

E. Multi-task Loss Function

After obtaining the features from each CNN stream, a dropout layer is applied to add randomness to this rich feature representation. We propose a multi-task loss function for this hierarchical job title classification problem. Job title categories are usually defined by the standard 23 SOC majors, and subcategories based on each SOC major. This proposed network makes predictions for two tasks, the SOC major and overall job title including SOC major and the subcategories. This fulfills two of our major objectives - one is to enforce the performance on SOC major classification so that even if the final prediction is not perfect, there is more chance that the prediction is still within the correct general scope. The other is to avoid the problem of hierarchical model which predicts the subcategory label based on the result of higher level. In hierarchical prediction model, when there's a problem with the top level, there's no chance to adjust it later. We use the sum of the cross entropy loss of these two tasks as our final loss for back propagation.

$$L = -\sum_{i=1}^{N} y_{major}^{(i)} log(\hat{y_{major}^{(i)}}) - \sum_{i=1}^{M} y_{all}^{(i)} log(\hat{y_{all}^{(i)}}) \quad (1)$$

where y_{major} and y_{all} are the true labels for class i, $\hat{y_{major}}$ $\hat{y_{all}}$ are the predicted probability values for that class, and N, M are the size of the output layer which $N = 23$ and $M = 4952$ for major and all classes respectively.

F. Noise-aware Loss Function

We used the labels generated by the previous state-of-the-art Carotene system as our training data. Each sample is labeled with the predicted carotene code and the confidence score. However, the classification results of Carotene are only 50.1% accurate on average. To deal with this noisy labeled data set, we evaluate two alternative noise-aware loss functions, applying confidence score on softmax score before computing loss and on loss directly. The predicted probability $\hat{y_{all}^{(i)}}$ is obtained by applying softmax to the output of the last fully connected layer. In weighted softmax loss function,

$$\begin{aligned} L_{weighted\_softmax} = & \\ & -\sum_{i=1}^{N} y_{major}^{(i)} log(\hat{y_{major}^{(i)}} * \alpha^{(i)}) \\ & -\sum_{i=1}^{M} y_{all}^{(i)} log(\hat{y_{all}^{(i)}} * \alpha^{(i)}) \end{aligned} \quad (2)$$

confidence score $\alpha^{(i)}$ of each training sample is applied to weight the predicted probability. More confident samples impact more to the loss value so as to the gradients. In weighted cross-entropy loss function

$$\begin{aligned} L_{weighted\_cross-entropy} = & \\ & (-\sum_{i=1}^{N} y_{major}^{(i)} log(\hat{y_{major}^{(i)}}) \\ & -\sum_{i=1}^{M} y_{all}^{(i)} log(\hat{y_{all}^{(i)}})) * \alpha^{(i)} \end{aligned} \quad (3)$$

the confidence score $\alpha^{(i)}$ weights the loss directly with the similar logic to have more confidence sample contribute more in computing loss value.

IV. Results

A. Data

The domain specific problem of job title classification usually suffers from limited size of training data [21] or using simple clustering method to estimate the labels [2]. In this presented work, we collect a data set with 15 millions samples of job title and job descriptions. To take advantage of this rich data set, we choose to build a weakly supervised classifier by using cheap and noisy labels. We apply the baseline model to the training data and take the output as ground truth. The label of job title categories has two components, major code from SOC major list and subcategory code pre-defined for each major. Besides being noisy, our data collected from online job postings is naturally unbalanced. Figure 4 shows the distribution of the original training data across the 23 major codes. Figure 5 shows the log with base 10 across all the 4952 subcategories. Some groups such as SOC45 and SOC55 have very small amount of samples.

We employ filter and data augmentation on this noisy and unbalanced data set. In the training data set, the ground truth is generated from the baseline model with a confidence score. For each subcategory of job title, if the size is larger than 1000, we filter out the low confidence data. Therefore, we keep samples only with confidence score greater or equal to 0.8 yielding a 10 millions training set. To up-sample the

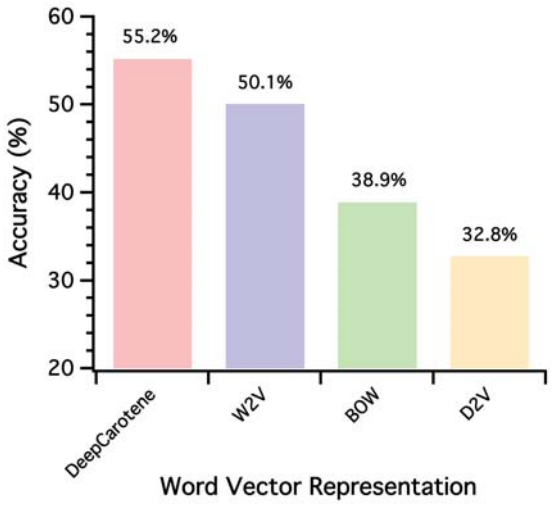

Fig. 3. The accuracy on hand-labeled testing file with proposed and baseline methods.

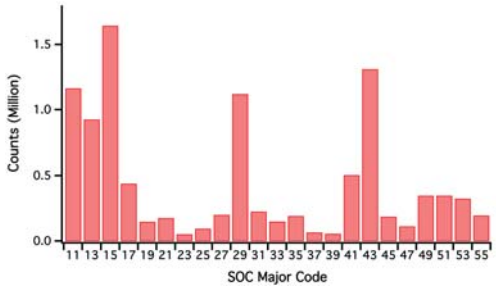

Fig. 4. The distribution of a subset of the original training data across the 23 major codes. Some major groups with fewer samples may give weak precision such as code 39 or get lower accuracy as code 55 as in table III.

underrepresented subcategories, we replicate each sample 20 times with variations in job descriptions.

From this collection of data, we randomly select 30,000 samples to build a validation set. To evaluate DeepCarotene, we randomly select a testing set with 1520 samples. This testing set is manually labeled with one of the 23 SOC major first, then with the corresponding subcategories.

B. Baseline

In our previous in-house job title classifier, we use a hierarchical system called Carotene [34]. The training data are pairs of job posting document and the correspondence normalized job title. Based on this data set, it applies clustering technique to create a taxonomy discovery component. This taxonomy defines the fine-grained categories for each SOC major. Then it learns a two stage hierarchical k-Nearest-Neighbors model to classify the input text to the most appropriate job titles, respectively in root and leaf levels of the taxonomy. Given a query job posting document, the coarse-grained classifier assigns the title to one of the 23 SOC majors. Then the fine-grained classifier is supposed to find the similar training data samples (neighbors) and classify the query according to the normalized titles of those retrieved neighbors.

To represent the input terms in vectors, the baseline model conducts experiments by fine-tuning 3 different models, Word2Vec [19], Doc2Vec [16] and Bag-of-Words [22], [28]. With the same testing data set, in Figure 3, we compare DeepCarotene with all 3 baseline models. The one with Word2Vec representation has the best performance and will be compared with DeepCarotene in details in this work.

C. Evaluation

We first evaluate loss function with and without weighting by label confidence as in Section III-F. We train the neural network with all three formulations of loss functions respectively and plot the loss value in Figure6. Although all three experiments achieve the same accuracy on the hand-labeled testing set, the weighted loss on softmax and on cross-entropy both start with and converge at a lower loss value as in the inset figure on the upper right. The experiments also present that the two weighted methods, weighted softmax and weighted cross-entropy, have no clear difference on the impact of loss eventually.

Next, we evaluate the performance of DeepCarotene on 1520 human-labeled data samples and compare it with the previous semi-supervised baseline model. Table II shows the performance over all the job title categories. The precision for single class is the ratio $tp/(tp+fp)$ where tp is the number of true positives and fp the number of false positives. However, for our multi-class problem, we calculate the precision for each label, and find their average weighted by support to deal with the label imbalance. Similarly, the recall (or known as accuracy) has been evaluated as the ration $tp/(tp+fn)$ where fn is the number of false negatives. Over all the classes, DeepCarotene achieves significant improvements on the weighted average among all the evaluation metrics. This indicates that DeepCarotene give more precise predictions rather than the baseline model.

TABLE II
PERFORMANCE ON ALL CLASSES

Metric	DeepCarotene	Baseline [34]
Precision	**0.630**	0.555
F1-score	**0.565**	0.498
Recall/Accuracy	**0.552**	0.501

Table III presents the performance of DeepCarotene and baseline model aggregated by SOC major. In terms of recall value (the same metrics as accuracy), the two models have very comparable results with the same recall value in 4 major groups while DeepCarotene out-performs in another 11 major groups. DeepCarotene improves the performance in some groups with low quantity of samples in the training set. For example, it increases around 4% in f1-score for $SOC53$ and more than 14% in $SOC19$.

We select four SOC major groups to present the ROC curve in Figure 7. $SOC19$ and $SOC53$ with accuracy at 87.5% and

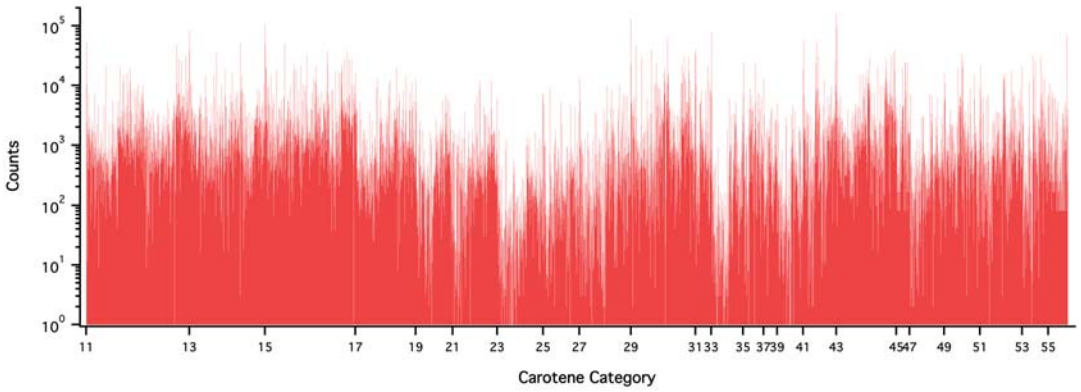

Fig. 5. The log with base 10 across all the 4952 subcategories on the subset of original data.

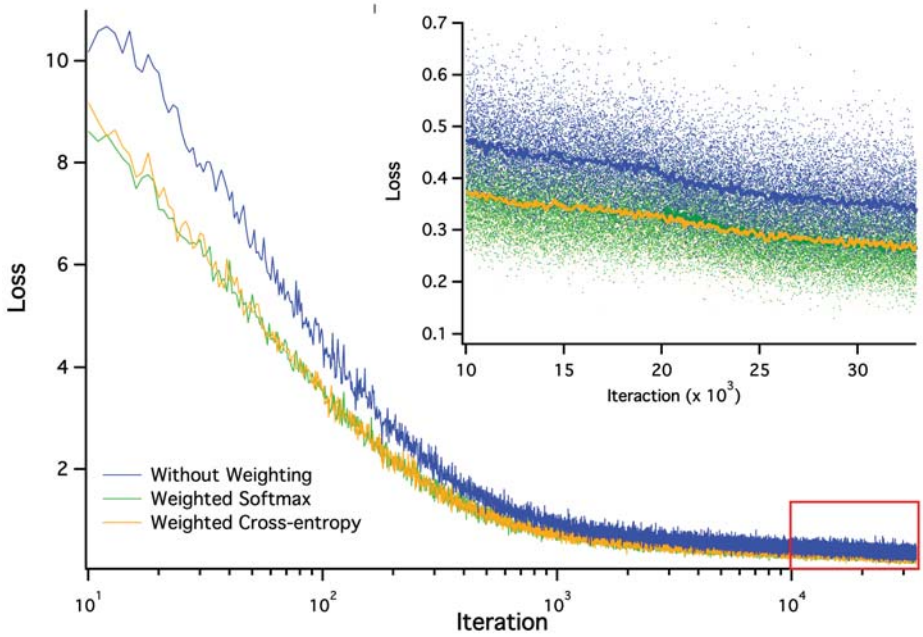

Fig. 6. Over the first 30,000 iterations of training, the value of loss (in blue) as Equation (1), using weighted softmax loss (in green) as Equation (2) and weighted cross-entropy loss (in yellow) as Equation (3). The inset figure on upper right zooms in to show the loss value after 10,000 iterations. It clearly presents the weighted loss converges at a lower value than using the loss without weighting.

89.9% gives the best area value at 97% and 98%. $SOC19$ gives 100% recall with false positive rate less than 20% while the recall of $SOC53$ reaches 80% with less than 5% false positive rate.

V. Conclusion

In this work, we propose a CNN based architecture to model job title classification given title and job description as input. The character stream for local matching and word stream for semantic matching are jointly trained with multi-task loss function to enforce the performance of SOC major prediction. The proposed DeepCarotene achieves improvement over our previous semi-supervised model with around 7% and 6% for precision and f1-score respectively. The deep learning model gives more precise predictions rather than the semi-supervised model over all classes and provides a promising architecture for job title classification.

In the future, this system can be extended and improved in directions towards both data and method. Lacking of high quality labeling data is the bottle neck to build an end-to-end job title classification system. The unbalanced distribution of training data is also a main challenge and efficient data augmentation with good variety of weak categories could improve the performance. To take advantage of the job description, we can extract the most deterministic subset of text in a separate model or introduce attention mechanism in WordCNN for job description.

TABLE III
PERFORMANCE ON SOC MAJORS

	DeepCarotene			Baseline [34]			
Major Id	Precision	Recall	F1	Precision	Recall	F1	Support
11	0.643	**0.812**	0.718	0.647	0.674	0.660	144
13	0.811	**0.699**	0.751	0.808	0.683	0.740	123
15	0.853	**0.884**	0.868	0.899	0.872	0.885	164
17	0.789	**0.769**	0.779	0.857	**0.769**	0.811	78
19	0.583	**0.875**	0.700	0.417	0.833	0.556	24
21	0.765	**0.839**	0.800	0.897	**0.839**	0.867	31
23	0.941	**1.000**	0.970	0.938	0.938	0.938	32
25	0.927	0.927	0.927	1.000	**0.951**	0.975	41
27	0.833	**0.851**	0.842	0.927	0.809	0.864	47
29	0.922	0.914	0.918	0.908	**0.931**	0.919	116
31	0.947	0.706	0.809	0.949	**0.740**	0.831	51
33	0.933	**0.933**	0.933	0.963	**0.867**	0.912	30
35	0.825	0.733	0.776	0.844	**0.844**	0.844	45
37	0.905	**0.864**	0.884	0.905	**0.864**	0.884	22
39	0.526	0.714	0.606	0.733	**0.786**	0.759	14
41	0.850	0.696	0.765	0.837	**0.813**	0.825	171
43	0.812	**0.903**	0.855	0.790	0.889	0.837	144
45	1.000	**0.545**	0.706	0.909	0.455	0.606	22
47	0.706	0.686	0.696	0.725	**0.829**	0.773	35
49	0.845	0.875	0.860	0.788	**0.929**	0.852	56
51	0.750	0.765	0.757	0.776	**0.882**	0.826	51
53	0.899	**0.899**	0.899	0.836	0.884	0.859	69
55	1.000	**0.500**	0.667	0.833	**0.500**	0.625	10

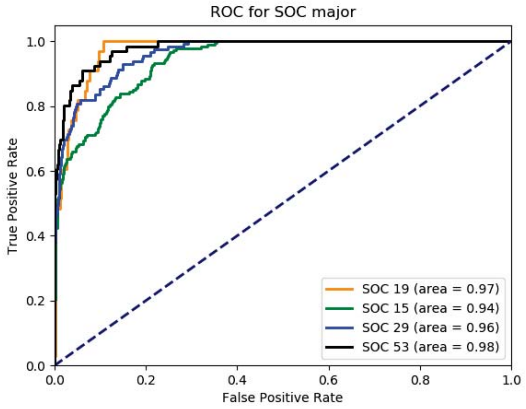

Fig. 7. ROC curve for four different Standard Occupational Classification.

REFERENCES

[1] stanford weak supervision course. https://dawn.cs.stanford.edu/2017/07/16/weak-supervision/.

[2] Eric Boucher and Clement Renault. Job classification based on linkedin summaries. *CS 224D, Stanford*, 2015.

[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[4] Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156, 2016.

[5] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269, 2015.

[6] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 701–710. ACM, 2016.

[7] Abdalraouf Hassan and Ausif Mahmood. Convolutional recurrent deep learning model for sentence classification. *IEEE Access*, 6:13949–13957, 2018.

[8] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

[9] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[10] Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in neural information processing systems*, pages 919–927, 2015.

[11] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[12] Mikael Karlsson and Anton Karlstedt. Product classification-a hierarchical approach. *LU-CS-EX 2016-31*, 2016.

[13] Tom Kenter and Maarten De Rijke. Short text similarity with word embeddings. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1411–1420. ACM, 2015.

[14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[17] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 165–174. ACM, 2016.

[18] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. 2015.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[20] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299. International World Wide Web Conferences Steering Committee, 2017.

[21] Negin Nahoomi. *Automatically Coding Occupation Titles to a Standard Occupation Classification*. PhD thesis, 2018.

[22] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *European conference on computer vision*, pages 490–503. Springer, 2006.

[23] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[24] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282, 2017.

[25] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[26] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110. ACM, 2014.

[27] Paroma Varma, Bryan D He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel Rubin, and Christopher Ré. Inferring generative model structure with static analysis. In *Advances in neural information processing systems*, pages 240–250, 2017.

[28] John Winn, Antonio Criminisi, and Thomas Minka. Object categorization by learned universal visual dictionary. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1800–1807. IEEE, 2005.

[29] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[30] H Yu, C Ho, Y Juan, and C Lin. Libshorttext: A library for short-text classification and analysis. *Rapport interne, Department of Computer Science, National Taiwan University. Software available at http://www. csie. ntu. edu. tw/cjlin/libshorttext*, 2013.

[31] Hsiang-Fu Yu, Chia-Hua Ho, Prakash Arunachalam, Manas Somaiya, and Chih-Jen Lin. Product title classification versus text classification. *UTexas, Austin*, 2012.

[32] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1253, 2018.

[33] Qianrong Zhou, Xiaojie Wang, and Xuan Dong. Differentiated attentive representation learning for sentence classification. In *IJCAI*, pages 4630–4636, 2018.

[34] Yun Zhu, Faizan Javed, and Ozgur Ozturk. Document embedding strategies for job title classification. In *FLAIRS Conference*, pages 221–226, 2017.