February 05, 2019

COMBINING DOMAIN KEYWORDS INTO AUTOMATIC DOCUMENT CLASSIFICATION BY DEEP NEURAL NETWORK

Qian Diao

Abhijit Talathi

Sarah Kuppert

Robert Frost

Adam Mattern

COMBINING DOMAIN KEYWORDS INTO AUTOMATIC DOCUMENT CLASSIFICATION BY DEEP NEURAL NETWORK

AUTHORS:
Qian Diao
Abhijit Talathi
Sarah Kuppert
Robert Frost
Adam Mattern

ABSTRACT

Techniques are described herein for combining the domain keywords into an automatic document classification system. It uses a Deep Neural Network (DNN) model that not only captures the word dependencies and context information in the document, but also encodes the domain knowledge into the category decision process efficiently.

DETAILED DESCRIPTION

Automatic document classification has been studied for a long time and different machine learning algorithms have been used to try to reach high precision and recall, as well as overall accuracy. One of the many methods tested has been to encode domain knowledge into classification models.

When humans manually classify documents, they can create a keyword list to represent important attributes of the category. For example, the 'health' category keywords list could include "therapy, disease, physicians, disabled, skin, doctors, etc.". These manually created keywords help the human labeler quickly make decisions when labeling a document.

Since domain keywords are a useful resource for manually labeling work, it may also be helpful when encoded into the automatic classification process. Therefore, techniques are provided for combining domain keywords into machine learning models to improve automatic classification.

To combine domain keywords into the automatic document classification process, a special Deep Neural Network (DNN) was designed to efficiently encode human knowledge and improve overall accuracy.

Published by Technical Disclosure Commons, 2019

Figure 1 below illustrates an example DNN model for combining domain keywords into automatic document classification.

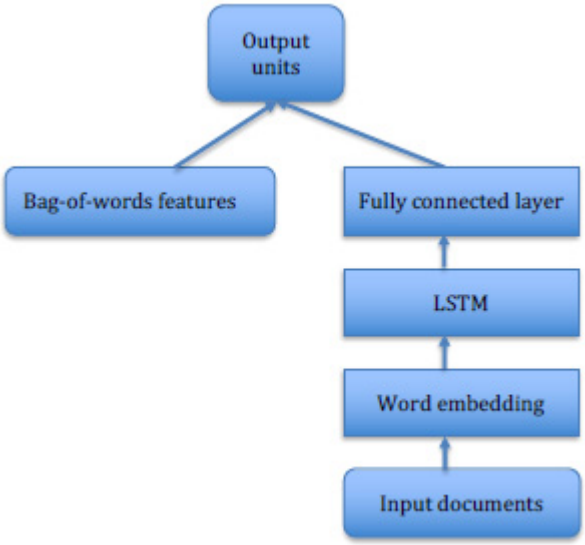

Figure 1

The bag-of-words features are the word features with term frequency-inverse document frequency (tf-idf) term weights. Here, domain keywords will have higher weights because each keyword is heuristically assigned to a pseudo-count (e.g., TF=10) to make sure its weight in the document is emphasized. By using the bag-of-words model, each document becomes a document vector. For example, to avoid out-of-memory (OOM) issues, only the top N (N = 4,800) weighted words may be picked from whole training data set for the modeling, so each document vector dimensionality size is 4,800.

The input documents are fixed-length text chunks (e.g., a chunk may be a sequence with the fixed-length of 500 words).

The word embedding layer is a language modeling way to represent each word as a vector of co-occurring words (e.g., the word 'doctors' may be represented as [0.1, 0.03, …, 0.15] with a dimensionality of 300), so a document (with a fixed-length of 500 words) may be transformed into a matrix (e.g., 500*300 in this case).

Long short-term memory (LSTM) is a special kind of Recurrent Neural Network (RNN) word dependency in the documents. It is a feature extraction hidden layer that reads in a sequence of words corresponding to a document and outputs a single fixed-length real-valued vector.

The fully connected layer here may be optional. It has connections to all activations in the previous layer, as seen in regular neural networks. The output from the LSTM layer represents high-level features in the data. While that output may be flattened and connected to the output layer, adding a fully-connected layer is (usually) a cheap way of learning non-linear combinations of these features.

The output unit is a softmax output layer. It uses L2 regularizations. Without any hidden layers, the keywords feature part (encoded with bag-of-words modeling) may be directly fed into this output unit, which may be viewed as an equivalent case of Logistic Regression (LR) estimation.

The design efficiently combines the domain knowledge (keywords features) into the classification process. It not only captures the word dependencies and context information in the document by using LSTM and word embedding, but also encodes the domain knowledge (keywords part) into the category decision model by the estimation equivalent to the LR model.

Figure 2 below illustrates an example overview of an automatic document classification system.

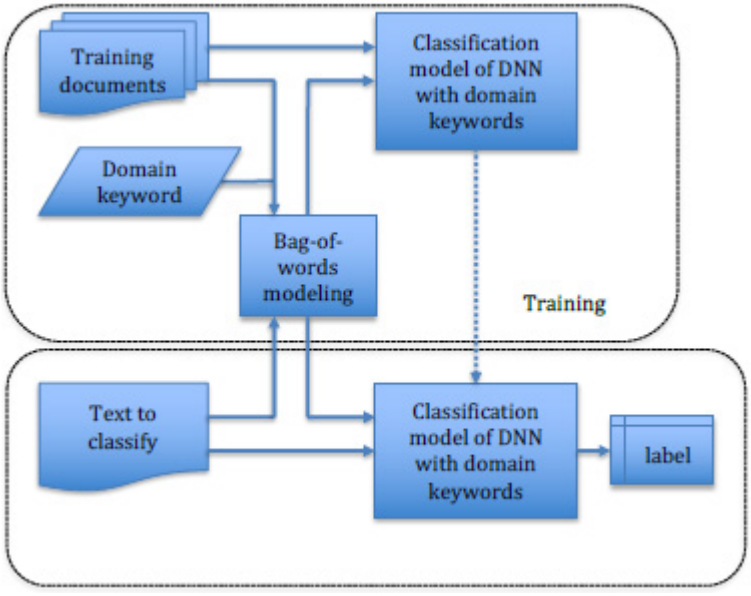

Figure 2

This design may be tested by applying it to a real-world problem, which is to build an automatic document classification system to predict the category label of an incoming

text from a taxonomy containing 81 categories. These categories have no sub-directory and are all in a flat structure.

The testing data set has 19,351 documents. The training data set has 1,014,666 documents. All the documents are unevenly distributed across the 81 categories. They are downloaded from the Internet and are all in English. Because manually creating keywords is a time-consuming process and requires a significant amount of domain knowledge, only two categories, 'health' and 'card,' and their associated keyword lists are used in the experiments. The domain experts manually created 2,142 keywords for category 'health', and 256 keywords for category 'card'.

In order to make better use of these keywords, the top similar words were searched for each keyword using the word2vec model as the keywords extension strategy. After the extension, the keywords list size for 'health' becomes 16,602, and the keywords list size for 'card' becomes 3,424. In practice, if the keywords list size is big enough, the extension step may be skipped.

The approach was tested on a real-world problem and the result is promising.

Table 1 below illustrates a comparison with the baseline models on all 81 categories. As shown, the design has the best overall accuracy (F1 score) and the highest recall value.

	Precision	Recall	F1
Our design: Deep Neural Network (DNN) with keywords	0.47	0.43	0.40
Baseline 1: Long Short-Term Memory (LSTM) without keywords	0.50	0.39	0.39
Baseline 2: Convolutional Neural Network (CNN) without keywords	0.46	0.36	0.36
Baseline 3: Logistic Regression (LR) without keywords	0.24	0.16	0.12
Baseline 4: Support Vector Machine (SVM) without keywords	0.52	0.42	0.39
Baseline 5: Naïve Bayes without keywords	0.46	0.38	0.35
Baseline 6: Support Vector Machine (SVM) with keywords using pseudo-count	0.50	0.40	0.37

Table 1

To compare the performance with state-of-the-art of automatic text classification systems, the tests were ran on five different popular baseline models without using any keywords. The best one to combine with domain keywords was selected using the heuristic pseudo counts (TF = 10) to emphasize its tf-idf weight. All of them were compared with this design. The evaluation metrics here are precision, recall and F1 score.

Table 2 below illustrates a comparison with the baseline models on category 'health.' Because the manually created domain keywords all came from two categories, 'health' and 'card,' the comparison results were also given in these two categories in order to be more specific. However, there are only 26 samples in category 'card' in the testing data set, so its result are not as statistically meaningful. Hence, only the result on category 'health' is provided. There are 2,147 samples in that category. As shown, for the category which has the associated keywords, the design has the highest precision and the highest over-all accuracy (F1 score).

	Precision	Recall	F1
Our design: Deep Neural Network (DNN) with keywords	0.71	0.56	0.62
Baseline 1: Long Short-Term Memory (LSTM) without keywords	0.71	0.51	0.60
Baseline 2: Convolutional Neural Network (CNN) without keywords	0.70	0.54	0.61
Baseline 3: Logistic Regression (LR) without keywords	0.70	0.26	0.38
Baseline 4: Support Vector Machine (SVM) without keywords	0.69	0.56	0.62
Baseline 5: Naïve Bayes without keywords	0.64	0.59	0.61
Baseline 6: Support Vector Machine (SVM) with keywords using pseudo-count	0.58	0.59	0.59

Table 2

In Tables 1 and 2, the baseline models include LSTM, Convolutional Neural Network (CNN), LR, Support Vector Machine (SVM), and Naïve Bayes. Among them the best model is SVM, so the sixth baseline model was chosen to add domain keyword knowledge with naïve pseudo-count TF weighting.

Techniques described herein not only capture the word dependencies and context information in the document using LSTM and word embedding, but also encode the domain knowledge (keywords part) into the category decision model by the estimation equivalent to the LR model. The DNN framework may be used to combine the keywords as bag-of-words features with the rest of the framework, as LSTM and word embedding, to efficiently capture both context information in the document and the human domain knowledge. The softmax output layer may be equivalent to an LR model.

In summary, techniques are described herein for combining the domain keywords into an automatic document classification system. It uses a DNN model that not only captures the word dependencies and context information in the document, but also encodes the domain knowledge into the category decision process efficiently.