Name Entity Recognition and Binary Relation Detection for News Query

Dan YE
Department of Computer Science and Information Engineering,
National Taiwan University Taipei, Taiwan

Keywords: Named entity extraction, Binary relation detection, News query, MITIE, Stanford classifier, Weka.

Abstract. This paper proposes an innovative slant on name entity recognition (NER) and binary relation detection (BRD) for current news search and retrieval. The main contribution is to apply MITIE tools to find relevant news according to the given news stories. BRD can retrieve relevant news according to the relationship between given query term and news dataset. Another salient contribution is to accomplish news story agency classification with Stanford classifier and automatic document classification tool WEKA. Given training data, many news stories with agency class labels, WEKA can predict and identify the test news stories’ class labels. The evaluation of relevant news is based on MAP@100 in 20 queries given 160000 news story dataset. The retrieved results of all the queries can demonstrate up to 100 relevant stories in terms of theirs IDs. The evaluation of news agency classification is based on accuracy with correct classified news stories. Scoring with NER yields the better performance.

Introduction

This paper applies multiple kinds of advance information retrieval and extraction tool in the state of art information extraction tools. The MITIE [1] includes tools for performing named entity extraction and binary relation detection as well as tools for training custom extractors and relation detectors.

For query generation, it is common in the IR literature [2] the inverse document frequency idf of a term is a function of the frequency f of term in the collection and the number N of documents in the collection. They use the function log (N/(f+1)).

Baseline algorithm A1-BASE [3] is a simple tf.idf based algorithm. It weights each term by tf.idf, where tf is the frequency of term in the text segment T. This result in larger weights for terms that appear more frequently in T, and larger weights for more distinctive terms of the news story is more likely to find articles related to the story. The baseline algorithm returns two terms with largest weight as the query.

Another baseline algorithm is that Term is weighted by tf.idf$^2$. Named entities are important for issuing focused queries. Idf component is more important than tf. The stem of a word is approximated by taking the first 5 letters of the word. Congress and congressional would share the same stem, congr. The intention is to aggregate the weight of terms that describe the same entity. The weight of a term is c.tf.idf$^2$, where c=1 if term was a noun and c=0.5 otherwise. We use this weighting scheme since nouns are often more useful in queries than others. The weight of a stem is the sum of the weights of its terms.

To issue a query, algorithm determines the two top-weighted stems and top-weighted term for each of these stems. These two stems form the query. Stems are computed by stemming both words. It finds the top-weighted term for two top-weighted stems.

The main contributions can be summarized as below.

1). The innovative application of NER and BRD into news retrieval is proposed.
2). MITIE tools have been applied to find relevant news.
3). The proposed method conquers language barrier to support search in multiple version language. MITIE tools are originally support English character only. Nevertheless, it is difficult for
MITIE tools to support Chinese news retrieval.

4). The large amount of news data in text format or html format should change them into ARFF files that Weka can read. Weka automatic tool input file default format are ARFF files.

5). The detailed analysis of several classifiers in Weka tool is presented.

The rest of the paper is organized as follows. In section 2, MITIE NER and BRD are elaborated. The analysis model for performance evaluation is also shown. Stanford classifier is employed in section 3. IllinoisSL structured prediction model is depicted in section 4. In section 5, WEKA makes prediction in agency labels, which is recapitulated. In section 6, the evaluation of NER model and WEKA prediction accuracy is analyzed. Experiment results and performance analysis of prediction classifier algorithm are carried out. In section 7, related work is discussed. Finally, conclusions are reiterated in the last section 8.

Named Entity Recognition

The state of art classifier methods have been successfully applied, e.g. Bayesian classifiers, decision trees, k-Nearest Neighbor (kNN), rule learning algorithms, neural networks, fuzzy logic based algorithms, maximum entropy and support vector machines.

Multi-label document classification becomes a popular research field. Three classifiers are successfully used for document classification: Naïve Bayes (NB), Maximal Entropy (ME) and Support Vector Machines (SVMs). Implement two approaches give the best classification scores. Automatic document classification includes document analysis, generation of document and class vectors based on document and class representatives, and matching document and class vectors to determine the class where a document belongs.

Stanford WordSegmenter

This tool can split Chinese text into a sequence of words based on some word segmentation standard. It is a Java implementation of the (Conditional Random Field) CRF-based Chinese Word Segmenter [4]. With external lexicon features [5], the segmenter segments more consistently and also achieves higher F measure when training and testing on the bakeoff data.

Stanford Named Entity Recognizer (NER)

Stanford NER [6] is a Java implementation of a Named Entity Recognizer. NER labels sequences of words in a text, which are the names of things, such as person and company names, or location names. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. Included with good named entity recognizers for Chinese, particularly for the 3 classes (PERSON, ORGANIZATION, LOCATION), Chinese models trained on just the 160000 news story as Chinese training data.

Stanford NER is also known as CRFClassifier. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. That is, by training models on labeled data, you can actually use this code to build sequence models.

The software provided is similar to the baseline local+Viterbi model in [7], but adds new distributional similarity based features (in the distSim classifiers). The distributional similarity features in some models can improve performance.

NER GUI

Type./ner-gui.sh on MacOSX to run an NER GUI demonstration. Then using the top option from the classifier menu, load a CRF classifier Stanford classifiers directory of the distribution. Then load a text file from the File menu, finally press the Run NER button to tag the text on named entity. Stanford Named Entity Tagger needs to load CRF classifier.

Binary Relation Detector

Binary relations allow us to predict relations between our named entities. MITIE comes with a number of classifiers through relationship of train data to predict relationship of test data.
**Stanford Classifier [8]**

A classifier is a machine learning tool that will take data items and place them into one of k classes. A probabilistic classifier, like this one, can also give a probability distribution over the class assignment for a data item. This software is a Java implementation of a maximum entropy classifier. Maximum entropy models are otherwise known as softmax classifiers and are essentially equivalent to multiclass logistic regression models (though parameterized slightly differently, in a way that is advantageous with sparse explanatory feature vectors). In other words, this is the same basic technology that you're usually getting in various of the cloud-based machine learning APIs. The classification method is described in [9].

The software requires Java 8. As well as API access, the program includes an easy-to-use command-line interface, ColumnDataClassifier, for building models. Its features are especially suited to building models over text data, but it also supports numeric variables.

**IllinoisSL [10]**

IllinoisSL is a Java library for learning structured prediction models [11]. It supports fast parallelizable variants of commonly used models like structured Support Vector Machines and structured Perceptron. The library consists of a core learning module and several applications, which can be executed from command-lines, a well-known C++ implementation of Structured SVM [12] in one-sixth of its training time. Comparison with other structured learning libraries, IllinoisSL is efficient, general, and easy to use.

Given a set of training data $D = \{x_i, y_i\}_{i=1}^l$, where instances $x_i \in \mathcal{X}$ are annotated with structured outputs $y_i \in \mathcal{Y}$, and $\mathcal{Y}$ is a set of feasible structures for the i-th instance. Structured SVM learns a weight vector $w \in \mathbb{R}^n$ by solving the following optimization problem:

$$
\min_{w, \xi} \frac{1}{2} w^T w + C \sum_i \xi_i^2
$$

subject to $w^T \Phi(x_i, y_i) - w^T \Phi(x_i, y) \geq \Delta(y_i, y) - \xi_i, \quad \forall i, y \in Y_i$

where $\Phi(x_i, y_i)$ is a feature vector extracted from both input x and output y. The constraints force the model to assign higher score to the correct output structure than $y_i$ to $\xi_i$ others. $\xi_i$ is a slack variable and we use $L^2$ loss to penalize the violation in the objection function. IllinoisSL supports two algorithms, a dual coordinate descent method (DCD) and a parallel DCD algorithm, DEMI-DCD. IllinoisSL also provides an implementation of Structured Perceptron. At each step, Structured Perceptron updates the model using one training instance $(x_i, y_i)$ by

$$
\bar{y_i} \leftarrow \arg \max_{y \in \mathcal{Y}} w^T \Phi(x_i, y),
$$

$$
w \leftarrow w + \eta (\Phi(x_i, y_i) - \Phi(x_i, \bar{y_i})),
$$

where $\eta$ is a learning rate. This implementation includes the averaging trick.

IllinoisSL Library [13] provides command-line tools to allow users to quickly learn a model for problems with common structures, such as linear-chain, ranking, or a dependency tree. The user can implement a custom structured prediction model through the library interface. IllinoisSL requires users to implement IInstance, IStructure, Abstract FeatureGenerator, and AbstractInfSolver. IInstance includes the input x, e.g., sentence in POS tagging. IStructure includes the output structure y, e.g., tag sequence in POS tagging. AbstractFeatureGenerator contains a function FeatureGenerator to extract features $\Phi(x,y)$ from an example pair (x,y). AbstractInfSolver provides a method for solving inference, i.e.,
\[ \arg \max_y w^T \phi(x, y) \quad \text{and for loss-augmented inference} \]
\[ \arg \max_y w^T \phi(x, y) + \Delta(y, y'). \] (4)

In POS tagging, this class will include implementations of a Viterbi decoder and hamming loss. Once these classes are implemented, the user can seamlessly switch between different learning algorithms. IllinoisSL package contains implementations of several common NLP tasks including a sequential tagger, a cost-sentive multiclass classifier, and an MST dependency parser.

**Weka Prediction Classifier**

Weka [14] is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well suited for developing new machine learning schemes.

The Weka machine learning is a modern platform for applied machine learning. Weka is an acronym, which stands for Waikato Environment for Knowledge Analysis. Weka makes learning applied machine learning easy, efficient and fun. It is a GUI tool that allows you to load datasets, run algorithms and design and run experiments with results statistically robust enough to publish.

Weka GUI allows you to output predictions [15] based on a previously saved model. After a data model has been saved, one can make predictions [16] for a test set, whether that set contains valid class values or not. The output will contain both the actual and predicted class. The AddClassificationFilter(packageweka.filters.supervised.attribute) can either train a classifier on the input data and transform this or load a serialized model to transform the input data.

For training the classifier, e.g., J48, on the input data and replacing the class values with the ones of the trained classifier. The Explorer section can save and load models article to setup the Explorer. Additionally, it is necessary to check the Output predictions options in the More options dialog. Right clicking on the respective results history item and selecting Re-evaluate model on current test set will output then the predictions as well (the statistics will be useless due to missing class values in the test set, so just ignore them).

**Experiment Results and Evaluation**

To test Chinese text, first run Stanford Chinese Word Segmenter, and then run NER on the output of that. The usage of MITIE examples cpp code is to implement relation detector to make evaluation on relevance of query news.

The performance results of the proposed evaluation scheme and highlight its advantage over NER and prediction classifier is evaluated in this section. Afterwards, the experiment-based accuracy for Weka OneR classifier model compared with different classifiers is presented.

For NER relevance detection [17-22], test sets are 20 queries based on MAP@100 evaluation.

\[
\text{map} @ 100 = \frac{1}{20} \sum_{i=1}^{20} \text{map}_i
\] (5)

There are 1805 test news files and 16000 training news files labeled with news agency APD, CTS and LTN. For prediction classifier labels on test set should use accuracy as a criterion:

\[
\text{accuracy} = \frac{\text{correctly classified news stories}}{\text{total news stories}}
\] (6)

The prediction label results are based on the running of WEKA classifier to obtain predictions/classifications.
Figure 1. Test news data set in WEKA preprocess.

Figure 2. WEKA classifier.

Figure 3. Weka predicted agency visualization.
Figure 4. Predicted agency results in ARFF viewer.

Figure 1 shows test news data sets input into WEKA preprocess system. Agency item is predicted content, which can be set up by manually with APD, CTS or LTN. Figure 2 indicates that multiple classifier schemes in WEKA classification. Meantime, it can configure different functions such as LibLinear, LibSVM or RBFNetwork, RBFClassifier. Figure 3 demonstrates that visualized classifier errors in graph representation with predicted agency. Figure 4 describes that the final predicted results in news agency. Figure 5 indicates that NNGE classifier output in several evaluation performance results.

Table 1 elaborates that the comparison of TP Rate, FP Rate, Precision, Recall, F-measure standard evaluation in different rules with various classifier schemes. OneR scheme is the best approach among these schemes. DTNB is the worst one than others. ZeroR and ConjunctiveRule have the same performance. Table 2 indicates comparison performance of TP Rate, FP Rate, Precision, Recall, F-measure standard evaluation in Prism, JRIP, NNGE, Ridor rules. Prism is best one in performance elevation. However, NNGE has better precision. JRIP is the same as Ridor in various kinds of performance criterions. Table 3 shows the comparison performance of TP Rate, FP

154
Rate, Precision, Recall, F-measure standard evaluation in different trees schemes, e.g. REPTree, LADTree, NBTree, HoeffedingTree, J48. NBTree and HoeffedingTree have superior performance than others. LADTree is better than REPTree and J48 schemes. The Latter two schemes possess the same performance results. Table.4 shows the comparison performance of TP Rate, FP Rate, Precision, Recall, F-measure standard evaluation in different trees schemes, such as DecisionStump, id3, FT, RandomTree, RandomForest, LMT. The statistics can be seen that RandomForest, RandomTree, and id3 have high performance than others. DecisionStump is better than LMT.FT and LMT also have same evaluation results in running experiments. All in all, we should choose which one scheme has the best overall performance among the tradeoff of all the accuracy criterions.

Table 1. Comparison results in different rules with various classifier schemes.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>OneR</th>
<th>ZeroR</th>
<th>Conjunctive Rule</th>
<th>DTNB</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP Rate</td>
<td>1.000</td>
<td>0.440</td>
<td>0.440</td>
<td>0.320</td>
</tr>
<tr>
<td>FP Rate</td>
<td>1.000</td>
<td>0.440</td>
<td>0.440</td>
<td>0.320</td>
</tr>
<tr>
<td>Precision</td>
<td>1.000</td>
<td>0.194</td>
<td>0.194</td>
<td>0.102</td>
</tr>
<tr>
<td>Recall</td>
<td>1.000</td>
<td>0.440</td>
<td>0.440</td>
<td>0.320</td>
</tr>
<tr>
<td>F-Measure</td>
<td>1.000</td>
<td>0.269</td>
<td>0.269</td>
<td>0.155</td>
</tr>
</tbody>
</table>

Table 2. Comparison results in different rules with classifier schemes.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Prism</th>
<th>JRIP</th>
<th>NNGE</th>
<th>Ridor</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP Rate</td>
<td>1.000</td>
<td>0.440</td>
<td>0.333</td>
<td>0.440</td>
</tr>
<tr>
<td>FP Rate</td>
<td>1.000</td>
<td>0.440</td>
<td>0.296</td>
<td>0.440</td>
</tr>
<tr>
<td>Precision</td>
<td>1.000</td>
<td>0.194</td>
<td>0.404</td>
<td>0.194</td>
</tr>
<tr>
<td>Recall</td>
<td>1.000</td>
<td>0.440</td>
<td>0.333</td>
<td>0.440</td>
</tr>
<tr>
<td>F-Measure</td>
<td>1.000</td>
<td>0.269</td>
<td>0.338</td>
<td>0.269</td>
</tr>
</tbody>
</table>

Table 3. Comparison results in different trees with classifier schemes.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>REP Tree</th>
<th>LAD Tree</th>
<th>NB Tree</th>
<th>Hoeffeding Tree</th>
<th>J48</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP Rate</td>
<td>0.440</td>
<td>0.840</td>
<td>1.000</td>
<td>1.000</td>
<td>0.440</td>
</tr>
<tr>
<td>FP Rate</td>
<td>0.440</td>
<td>0.126</td>
<td>1.000</td>
<td>1.000</td>
<td>0.440</td>
</tr>
<tr>
<td>Precision</td>
<td>0.194</td>
<td>0.883</td>
<td>1.000</td>
<td>1.000</td>
<td>0.194</td>
</tr>
<tr>
<td>Recall</td>
<td>0.440</td>
<td>0.840</td>
<td>1.000</td>
<td>1.000</td>
<td>0.440</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.269</td>
<td>0.826</td>
<td>1.000</td>
<td>1.000</td>
<td>0.269</td>
</tr>
</tbody>
</table>

Table 4. Comparison results in different trees with classifier schemes.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Decision Stump</th>
<th>Id3</th>
<th>FT</th>
<th>RandomForest</th>
<th>Random Forest</th>
<th>LMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP Rate</td>
<td>0.480</td>
<td>1.000</td>
<td>0.440</td>
<td>1.000</td>
<td>1.000</td>
<td>0.440</td>
</tr>
<tr>
<td>FP Rate</td>
<td>0.409</td>
<td>1.000</td>
<td>0.440</td>
<td>1.000</td>
<td>1.000</td>
<td>0.440</td>
</tr>
<tr>
<td>Precision</td>
<td>0.442</td>
<td>1.000</td>
<td>0.194</td>
<td>1.000</td>
<td>1.000</td>
<td>0.194</td>
</tr>
<tr>
<td>Recall</td>
<td>0.480</td>
<td>1.000</td>
<td>0.440</td>
<td>1.000</td>
<td>1.000</td>
<td>0.440</td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.345</td>
<td>1.000</td>
<td>0.269</td>
<td>1.000</td>
<td>1.000</td>
<td>0.269</td>
</tr>
</tbody>
</table>

Related Work

Previous algorithms are basically trying to extract keywords [14] from a stream of text so that the keywords represent the “current” piece of the text. Using existing terminology this can be called time-based keyword extraction. There is a large body of research on topic detection and text
summarization. The time-based summarization has also been studied [3], but to the best of our knowledge there is no prior work on time-based keyword extraction.

The just-in-time IR project at MIT [23] has focused on retrieving personal files such as notes and archived email messages that a user would currently find useful. This project first produced the Remembrance Agent, which looks at a document the user is editing in Emacs and matches fragments of this document such as the last 50 words against a corpus of personal files. The follow Margin Notes system performs a similar task, but observes the web pages that a user views in a web browser.

The research on keyphrase extraction [24] and specifically the algorithm in the language model for information retrieval by [25-29] is the most related to this work.

Conclusion

In this paper, WEKA creates prediction analysis method is proposed to make news classification. The key contribution is to choose the maximum prediction accuracy by comparing performance in different classifiers of WEKA. Experiment results were shown to demonstrate that MITIE NER and BRD could obtain better relevance performance. WEKA can make prediction more powerful in practice, which enjoys superior classified performance and accuracy.

References


