Summer Independent Study

Dandan Zheng's Report

***Abstract*—This summer independent study is intended to verify some assumptions based on an ongoing paper [1] and to understand more about machine learning. The machine learning technologies used include label propagation, logistic regression, smoothing, and so on.**

***Index Terms*—Machine learning, Classifier, semi-supervised learning**

I. Dataset

The dataset is Reuters Corpus version one (RCV1). It contains a bunch of labeled documents. The labels are in hierarchical categories meaning that each document can have multiple labels from different hierarchy levels. Also, a document can have multiple labels in the same hierarchy. For example, a document can be categorized both into Corporate categories and Economics categories even though they are in the same hierarchy.

In the interest of the simplicity of the experimentation, only the categories in the first level are considered. They are: CCAT (Corporate and Industrial), ECAT (Economics), GCAT (Government and Social) and MCAT (Markets).

II. Step one – feature selection and doc-word graph

There are four different categories, we choose 10 documents from each category and the words in these documents form the feature set. So, totally 40 documents are used as the "pool" of the features. We also consider these 40 documents as the seed documents which means they are labeled and will be used in the step 2 as the seeds.

Then, we calculate the tf*idf[2] weight for each feature (the words from the 40 documents) to each training document. Currently, the training documents are from RCV1 small training set.

Now we have a graph. On the left are the words, on the right are the documents and between the words and the documents are the tf*idf weight.

III. Step two – label propagation

In step one, we have two outputs. One is the seed documents – documents with labels, the other one is the graph – features and training documents linked by tf*idf value.

In step two, we use these two outputs from step one as the inputs.

Using these two inputs, we can have label propagation [3]. The rough idea of label propagation is that in a graph with weighted edges and some labeled nodes as the seed, the labels can be expanded to siblings based on the confidence provided by the weights. After several iterations, each and every node would have a preferred label.

A library called junto [4][5][6] is used to accomplish the label propagation task. The data file format and the steps are well described in the link [4].

This step turns the unlabeled features (the words from the 40 documents) into labeled features.

IV. Step Three – classifier built using logistic regression

We are going to build Boolean classifiers for each category so that each classifier for that category (the label) would say either yes or no to an input document. Since we have four categories (CCAT, ECAT, GCAT and MCAT), there will be totally four classifiers being built.

In order to build the classifier, we need to get training data to train the classifier. It will be discussed in section A. We will use regularized logistic regression as the classifier. It will be discussed in section B. We'll also show the accuracy of the classifiers comparing to other kind of classifiers in section C.

A. Data preparation

Now we have a bunch of words as features and each word is labeled (result of step 2). We separate the words according to their labels (categories). The word of one label would be a feature of the classifier of the same label. Each category would have a bunch of words linked to it and each word would be in exactly one category. These words are the features for each classifier.

We'll also use the training documents (RCCV1 small training set) to generate training data for each classifier. One piece of training data would be generated from one original training document for each classifier.

Each training data contains a Boolean value followed by a list of words. If the original document is labeled as the same category as of the classifier the training data is assigned to, then the Boolean value would be true, otherwise, it would be false. Then we will go through each word in the document. If the word is in the features of (the labeled words assigned to) that classifier, we put it to the training data's word list, or we just ignore it.

Let's have a concrete and simplified example. Imagine we have the following two documents labeled as CCAT and ECAT.

Doc1 (CCAT): IT companies pay a lot of money to the employees.

Doc2 (ECAT): Banks store money.

For CCAT and ECAT category, we have the following features (words).

Feature for CCAT: companies, employees …
Feature for ECAT: banks, money …

The training data for the classifier of CCAT would be:
Doc1_CCAT: {{true}, {companies, employees, _bias_}}
Doc2_CCAT: {{false}, {_bias_}}

The training data for the classifier of ECAT would be:

Doc1_ECAT: {{false}, {money, _bias_}}
Doc2_ECAT: {{true}, {banks, money, _bias_}}

Notice that each generated data would have a bias word.

B. *Regularized logistic regression*

After reading several documents about logistic regression (e.g. Wikipedia), I cannot find a document describing logistic regression, one way to implement it (stochastic gradient descent) and the way to regularize better than Professor William Cohen's notes on it [7]. The code is written based on the algorithm in the section 4 of [7]. Number 20 is chosen as the number of iterations and other parameters are the same of the recommendation value from [7].

C. *Some results*

F1 score [8] is introduced to be the measure of the accuracy of each classifier. The test data is RCV1 small test set.

Classifier / label / category	**F1Score**
CCAT	71.7%
ECAT	40.2%
GCAT	66.2%
MCAT	64.1%

Table1: Classifiers' accuracy

Then instead of assign different features to different classifiers, all the features are used for each classifier.

Classifier / label / category	**F1Score**
CCAT	77.8%
ECAT	51.2%
GCAT	78.4%
MCAT	69.7%

Table2: Classifier's accuracy. Each uses all the features

Then, regularize the second cluster of classifiers:

Classifier / label / category	**F1Score**
CCAT	77.9%
ECAT	52.0%
GCAT	78.4%
MCAT	70.2%

Table3: Classifier's accuracy. Each uses all the features. Regularized

Then, instead of using features with the same label as opposed to each classifier, features are randomly assigned to each classifier. Each classifier will be assigned roughly the same amount of features. We ran this random feature assign and learning process 10 times. The accuracy is the average number of the 10 round.

Classifier / label / category	**F1Score**
CCAT	72.0%
ECAT	42.9%
GCAT	68.7%
MCAT	62.4%

Table4: Classifiers' accuracy. Features randomly assigned to each classifier, averaging on 10 iterations.

Weka[9] is an off the shelf machine learning library and we use it as a reference to the accuracy of the classifier. The input data for Weka are simply the training documents. Two different classifiers are used. One is Naïve Bayes, the other one is logistic regression. No features are explicated picked as opposed to the classifiers in this article, so they may use all the words in the documents as the features.

Classifier / label / category	**F Score**
CCAT	68.4%
ECAT	44.5%
GCAT	67.7%
MCAT	75.7%

Table5: Weka – Naïve Bayes Classifiers' accuracy

Classifier / label / category	**F Score**
CCAT	69.7%
ECAT	34.0%
GCAT	60.3%
MCAT	69.4%

Table6: Weka – Logistic Regression Classifiers' accuracy

From the result, we understand that the classifiers with all the features still beat the classifier with selected features. In the future work, we may analyze on lower level categories and see if the classifiers with selected features can perform better.

V. STEP FOUR – ASSUMPTION VERIFICATION

An assumption can be derived from [1], theorem 6: when PMI [10] value between two features from different classifiers is smaller, more chances are these two features would be both positive for their classifiers.

We store all the pairs of features which are from different two classifiers. For each of these pairs, we assign a Boolean value. The Boolean value would be true if and only if both of these two features are positive for the classifier. See the definition of positive in section A below. We also calculate the PMI value for the pair. See the detailed calculation in section B below.

A. Positive feature definition

First, we need to define what positive means for the classifier. For the sake of the simplicity of this experimentation, we consider the Naïve Bayes weight as the judgment baseline. For each classifier, we calculate the average word appearance in all the documents labeled in the classifier's category. It equals to the total number of word count in these documents divided by the distinct word count (the vocabulary count). If the appearance time for that particular feature (the word) is greater than the average value, then that feature is positive. Otherwise, it is negative. This means that if we see the particular word more often than the average in the documents with that label, we consider that word positive.

B. PMI value

PMI value of the feature pair is defined as:

$$log \frac{P(x,y)}{P(x)P(y)}$$

$P(x,y)$ means the probability that these two features both appear in the same document. $P(x)$ means the probability that this particular feature appears in one document. The lower the PMI value, the less these two features are related.

Because a large portion of the pair would have $P(x,y) = 0$ due to the limited training documents, we need to smooth the PMI value a bit. A fake document containing all the features is added to the training document. Due to this fake document, any pair of feature would at least appear together in one document.

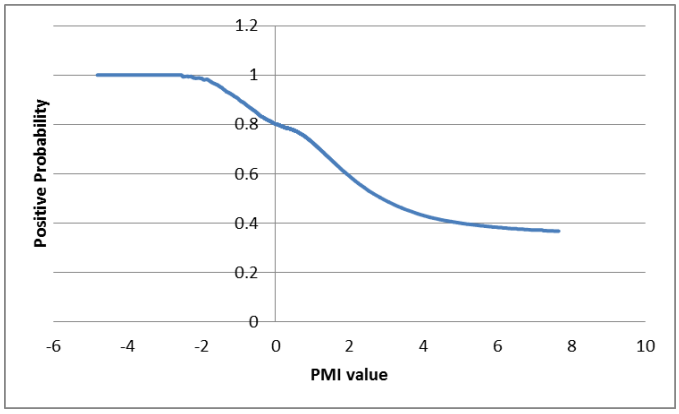
After smoothing, the PMI value equals to:

$$log \frac{\frac{occurence(x,y)+1}{\#doc+1}}{\frac{occurence(x)+1}{\#doc+1} * \frac{occurence(y)+1}{\#doc+1}}$$

$$= log \frac{(occurence(x,y)+1)*(\#doc+1)}{(occurence(x)+1)*(occurence(y)+1)}$$

C. Some results

We calculate the cumulative probability of pairwise positive for each PMI value. In the graph, the x-axis is the PMI value. The y-axis is the pairwise positive (meaning that both of the features are positive) probability of the feature pairs whose PMI values are equal to or less than the x-axis value.



Graph1: Cumulative Positive Probability VS. PMI value

This graph shows that when the PMI value increases, the chances that both two features in the pair are positive are less.

VI. Future Work

We may check the classifier's accuracy based on different hierarchy. Instead of the four big categories (CCAT, ECAT, GCAT, MCAT), we can use the next level hierarchy so that it would be more sparse and more chances that the classifiers with selected features would perform better.

We can generate a graph whose nodes are the features and edges are the PMI value. Using this graph and some seed nodes, we can pick up more high confident features for different classifiers. After that we may use these features to label documents and using these newly labeled documents to be the training document and see if we can get better result.

Also, a "snowball sampling" graph may give us a good visualization of whether features having some low PMI values with other features tend to link to each other or they just tend to link to random features. It would be encouraging if within low PMI bound, we can have more features. The more features, the more chances that we can build better classifiers.

Acknowledgment

Sincerely thanks to Professor William, Professor Avrim and Katie. Without their help and kind guidance, I cannot have this experimentation not even close. They lead me to this very interesting field and I definitely think that it worth further study and learn.

References

[1] The paper from Professor Avrim and co-workers. Not sure OK to share it, put a place holder here.

[2] http://en.wikipedia.org/wiki/Tf*idf

[3] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

[4] https://github.com/parthatalukdar/junto

[5] New Regularized Algorithms for Transductive Learning. Partha Pratim Talukdar, Koby Crammer, ECML-PKDD 2009

[6] Experiments in Graph-based Semi-Supervised Learning Methods for Class-Instance Acquisition. Partha Pratim Talukdar, Fernando Pereira, ACL 2010

[7] Stochastic Gradient Descent (Sgd-notes). William Cohen, ML with Large Dataset class reading, 2012.

[8] http://en.wikipedia.org/wiki/F1_score

[9] http://www.cs.waikato.ac.nz/ml/weka/

[10] http://en.wikipedia.org/wiki/Pointwise_mutual_information