UNIERSITY OF SOUTHERN CALIFORNIA

Individual Report on Project (9): Keyword Extraction - Predicting Stack Exchange Tags

Name: Siyuan Song
email address: siyuanso@usc.edu
Rest of my group:
Liang Liu, Qinqin Zhu
Name of the Bitbucket repository for your group's project:
qinqin_csci544

December 6, 2015

1 PROJECT OVERVIEW

Project Goal
This project comes from kaggle competition (https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction), the goal of the project is to predict the tags (a.k.a. keywords, topics, summaries) on large data (7GB data in total) from the Stack Exchange sites, given only the question text and its title. An example of training data point is shown below:

id: 1
title: how to check if an uploaded file is an image without mime type?
body: I'd like to check if an uploaded file is an image file (e.g png, jpg, jpeg, gif, bmp) or another file. The problem is that I'm using Uploadify to upload the files, which changes the mime type and gives a 'text/octal' or something as the mime type, no matter which file type you upload. \n \n Is there a way to check if the uploaded file is an image apart from checking the file extension using PHP? \n
tags: php, image-processing, file-upload,upload, mime-types

We can see from the above that title is a question which summarizes the content and body is a html file with details, tags are some keywords which are needed to be predicted. The problem is in the NLP field of information extraction.

Methodology
The methodologies of the project consist of four parts.
First, **data cleaning** was done. As the data was retrieved from Stack Exchange sites, data has duplicate, thus duplication was removed. And the content in body is in html file, we removed the html tags. And as usual, we removed punctuations and removed stopwords. And required by some models, we also did lexicon creation. We also did statistics calculation to get an overview of how the tags distributed (almost a normal distribution).

Second we did **feature extraction** which has three parts. We run nltk tool to get the pos tags for files (as pos tagging was too slow, we just did a portion of the training data for classification models mentioned later), CountVetorizer to get the frequencies of words, and TfidfVectorizer to get the TF-IDF features (which is based on inverse document frequency). Those three steps will provide the features needed for models implemented.

Third, we implement some **models** for tags prediction.
1. Baseline model
We simply predicted tags by seeing whether the tag was present in the training data.
2. Classification models
The first set of models we tried classification models (done by myself and details will be given in next part) with pos tags, including CRF, SVM and Adaboost. Adaboost is an algorithm to boost weak predictor to strong predictor by iteratively adjusting weights for different data points according to the results generated from previous step, Those models are slow and we just used 10% of the training data.
3. Nearest Neighbor Search (NNS)
This algorithm is based on the TF-IDF features generated above, and for testing, we first find the tags that were in the training data, if yes, we directly output tags, otherwise assign the tags with the the highest k similarity scores.
4. Tag-Keyword Co-occurence Model
This method is to construct a co-occurence matrix to model the joint probability of word and tags. And we will find the probability $max\ p(tags|word)$ for each word, and return also the k tags with largest probability.
5. SVM and KNN Extension to Co-occurence Model
The model is based on the previous model. The idea is that, after we get the word-tag results from the co-occurence model, we will then use it as input to SVM and KNN models.
6. Fuzzy Nearest Neighbor Search (FNNS)
This algorithm is an extension of nearest neighbor search. First by Implementing the keyword-tag co-occurence model above, we can narrow the search space to 10-20 candidates which reduced the curse of high dimension, then we can do linear NNS.
7. ACT-R's inspired Bayesian Model
This model is an implementing of result in the paper (Byrne, 2013).

Performance Evaluation
We used 90% data for training and 10% data for testing. The evaluation criteria is precision, recall and f1 score which is typical in information extraction problem.
The result is as follows:

Model	F1	Precision	Recall
Baseline Model	0.154	0.167	0.143
CRF with window size 0	0.366	0.654	0.254
CRF with window size 1	0.373	0.638	0.264
CRF with window size 2	0.375	0.633	0.267
SVM with window size 0	0.354	0.636	0.245
SVM with window size 1	0.369	0.615	0.263
Adaboost with window size 0	0.204	0.697	0.120
Adaboost with window size 1	0.207	0.694	0.121
Tag-Keyword Co-occurence Model	0.655	0.635	0.675
Fuzzy Nearest Neighbor Search Model	0.744	0.741	0.747
ACT-R inspired Bayesian Probabilistic Model	0.798	0.811	0.785

Conclusion
Baseline model, as imagined, is the worst for its simplicity. All models implemented as a classification problem, which was popular and took large parts in homework problems, was not good, especially in recall. This is because both pos tags and the models, especially adaboost, which trained models iteratively are slow. Thus we don't have enough computation resources to carry out the models with all data, which makes recall low. And the three models Tag-Keyword Co-occurence Model, Fuzzy Nearest Neighbor Search Model, ACT-R inspired Bayesian Probabilistic Model, are easy to scale. And the ACT-R inspired Bayesian Probabilistic Model, which suits the problem well, performs best.

2 My Project Tasks

My primary task in this problem is to implement the classification models for the project (although the performance turns out not to be good). The idea was inspired by the paper (Carreras, Marquez, Pardo, 2002) which implements adaboost for named entity recognition (and they got the best performance). For the limitation of computation resources, I just randomly chose 10% of the cleaned data.
After I got the cleaned data, pos tag was done by the nltk package as needed by classification models (This step is extremely slow and 10% of the data need around 20 hours to tag). Then a program will be run to generate data with different window sizes as features for classification models. Window size k means that for predicting whether a word will be keyword or not, we will use the word and all words having distance less than or equal to k (at most k-1 words inside the two words) with their pos tags. An example of data points with window size 1 generated is shown below:
Original Data: do|VB a|DT sql|NN connection|NN

Data points generated (four in total as there are four words):
0 0do 0VB 1a 1DT
0 -1do -1VB 0a 0DT 1sql 1NN
1 -1a -1DT 0sql 0NN 1connection 1NN
0 -1sql -1NN 0connection 1NN

After getting the data as shown above, we are able to implement for CRF model as this is the input format asked. I tried CRF models with window size 0, 1 and 2, the result is as follows:

Model	F1	Precision	Recall
CRF with window size 0	0.366	0.654	0.254
CRF with window size 1	0.373	0.638	0.264
CRF with window size 2	0.375	0.633	0.267

We find that with windows size 1, about 1% was improved from window size 0, while to window size 2, no significant improvement has been achieved. Thus I then just tried window size 0 and window size 1 for following models.
Then I implemented SVM and Adaboost (Zhu, Zhou, Rosset, Hastie, 2009) in sckitlearn. Adaboost works extremely well on classification problems with weak learner by adjusting weights for each iteration, a sketch of the algorithm is as follows:

Given: $(x_1, y_1), \cdots, (x_m, y_m)$ where $y_i \in \{-1, 1\}$
Initialize: $D_1(i) = 1/m$ for $i = 1, \cdots, m$.
For $t = 1, \cdots, T$:
1. Train a weak learner $h_t$using distribution D_t. (usually decision tree)
2. Set $\epsilon_t = Pr_{i\ D_t}[h_t(x_i \neq y_i)]$
3. Choose $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
4. Update weight for each data point: $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ where Z_t is to make D_{t+1} to be distribution.
The final model will be: $\text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$

Before implementing sckitlearn, as the input of sckitlearn needs to be a matrix, we further transformed the data preprocessed above into a sparse matrix with scipy (each data have less than 10 features of feature space of thousands of words, sparse matrix will be feasible and save huge amounts of memory). The result for SVM and adaboost are as follows:

Model	F1	Precision	Recall
SVM with window size 0	0.354	0.636	0.245
SVM with window size 1	0.369	0.615	0.263
Adaboost with window size 0	0.204	0.697	0.120
Adaboost with window size 1	0.207	0.694	0.121

conclusion of my part

First, we see that in this problem, CRF model gives us the best result among all classification models for they give the highest recall. Adaboost improves precision a lot while lower recall. I think if more data are incorporated, adaboost will get a higher recall and finally outperform CRF. Because from my former experience, Adaboost usually gives rather good result in classification it's possible a sign showing that Adaboost need more data for implementation. Second, since enlarging window size didn't improve the result a lot, it seems that the results are actually not related with its neighborhoods a lot compared with the word itself (and that's why the models Tag-Keyword Co-occurence Model, Fuzzy Nearest Neighbor Search Model, ACT-R inspired Bayesian Probabilistic Model are better as they focus more on the word itself). Lastly, as the popular saying goes, "more data is always better than less data, better data is always better than more data".

3 Project Resources

* The training and testing data are from Kaggle (https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction)
* We used nltk package to do pos tagging (nltk.pos_tag)
* We used scipy package to store data as sparse matrix (scipy.coo_matrix)
* CRFsuite to implement CRF model (http://www.chokkan.org/software/crfsuite/)
* We used scklearn in following ways:
* CountVectorizer to count tokens (sklearn.feature_extraction.text.CountVectorizer)
* Convert a collection of raw documents to a matrix of TF-IDF features.
(sklearn.feature_extraction.text.TfidfVectorizer)
* Compute the linear kernel between X and Y. (sklearn.metrics.pairwise.linear_kernel)
* Scale input vectors individually to unit norm (sklearn.preprocessing.normalize)
* Implement Kmeans (sklearn.cluster.KMeans)
* Implement SVM model (sklearn.svm.LinearSVC)
* Implement Adaboost model (sklearn.ensemble.AdaBoostClassifier)

4 References

1. Bird, S. K. (2009). Natural language processing with Python. O'Reilly Media.
2. Byrne, C. S. (2013). Predicting tags for stackoverflow posts. 2013. *Proceedings of ICCM*.
3. Facebook. (2013). Identify keywords and tags from millions of text questions. Retrieved from Kaggle competition: https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/
4. J. R. Anderson, D. B. (2004). An integrated theory of the mind. *Psychological review*, 111, 1036.
5. Lott, B. (2012). Survey of Keyword Extraction Techniques.
6. Sahami, M. a. (2006).A web-based kernel function for measuring the similarity. New York.
7. Zhu, J. , Zhou, H. , Rosset, S. , Hastie, T. (2009). Multi-class Adaboost. *Statistics and Its Interface*. Vol. 2, No. 3, 349-360.
8. Carreras, X. , Marquez, L. , Pardo, L. (2002). Named Entity Extraction Using Adaboost. *COLING-02 proceedings of the 6th conference on Natural Language Learning*. 20, 1-4.

5 REVIEW 1

The group number was: 5

What was the goal of the group's project?
The goal of the project is to build a text-based dialogue system. The system will take voice input and return the route between two places.

How did the group attempt to accomplish the goal?
They used the toolkit CMU Sphinx to implement it. And they also build dictionary and language model for simple commands and place names by using Sphinx4 API to convert voice input to strings. Also they implement natural language understanding with keyword spotting.

How did the group evaluate their work and what was the result?
They asked five people to perform some tasks and then give feedbacks regarding to task completion and user satisfaction. And they all give positive feedback for normal place names and 1 negative feedback for short place names.

6 REVIEW 2

The group number was: 15

What was the goal of the group's project?
Build a classifier that given lyrics for a song, genre will be predicted. And they also run various NLP techniques and add lyrics-specific features to see what level of result quality they can achieve.

How did the group attempt to accomplish the goal?
First, they got lyrics of about 300 songs online in 6 genres. For features, they used bag of words and pos-tags and during for each token. For models, they tried Naive Bayes, SVM and CRF.

How did the group evaluate their work and what was the result?
They used 5-fold cross validation and accuracy for evaluation, they also tried the recall, precision and F1 score for each genre. The result shows that:
1. Some genres show strong correlation but others who weak correlation
2. Adding POS tag generally reduces the accuracy
3. Adding timing information can increase the accuracy, but the weight should be carefully designed

7 REVIEW 3

The group number was: 17

What was the goal of the group's project?
The project will answer some questions: 1. Can vast amounts of information from social network help to improve predictions of some stocks? 2. Can sentiment analysis improve predictions accuracy?

How did the group attempt to accomplish the goal?
There are two parts of the project, first, they collected stock data from Yahoo stock API and tweet data from twitter API to get All Elon Musk tweets. And they also NLTK package and TextBlob package for sentiment analysis and used SVM in both without sentiment features and with sentiment features. The second part they crawled 51732 comments about apple from Stocktwits. Use SVM and Linear Regression model for forecasting.

How did the group evaluate their work and what was the result?
They used accuracy for prediction, for the first part features with sentiment data gives 48.24% while features without sentiment data gives 48.05%, and for part 2, SVM model gives accuracy : 43.90% (test on training set) 51.22%(train on 31 data test on all, 21 correct/total 41) and Linear Regression model accuracy: 43.90% (test on training set,18 correct/total 41) 48.78% (train on 31 data test on all 20 correct/total 41).

8 REVIEW 4

The group number was: 19

What was the goal of the group's project?
For the project, if given the sentence, let the classifier predict the most likely preposition. If the predicted preposition is different from that chosen by the user. A wrong usage is flagged.

How did the group attempt to accomplish the goal?
They considered top 25 prepositions and get 15800 examples from CNN, BBC, WSJ, NY Times, LA Times, Huntington Post, Wikipedia, and etc. For features, they used lexical features and combinations of that (for example, token and POS n-grams in a 2 word window around the preposition). For models they used maximum entropy, support vector machine, decision tree and naive bayes.

How did the group evaluate their work and what was the result?
They used Weighted Accuracy for performance analysis, and the result is as follows:
1.Maximum Entropy: 0.28
2. Support Vector Machine: 0.41, perform the best, so chosen to be backend)
3. Decision Tree: 0.31

4. NaÃŕve Bayes: 0.30

9 REVIEW 5

The group number was: 25

What was the goal of the group's project?
The goal of the project is to detect different cuisines (Greek, Chinese, Indian, Mexican or Italian) using text classification.

How did the group attempt to accomplish the goal?
They first collected data from allrecipes.com, bbcgoodfood.com, saveur.com and foodnetwork.com. Then they did data cleaning by removing stop words and do pos tag. For features, they used preparation and Ingredients data, preparation order and pos tags. And they used Maximum Entropy model for classification.

How did the group evaluate their work and what was the result?
They used precision and recall for each class, and there are two conclusions they got: 1. for small amount of data, use Ingredients as features, 2. for larger amount of data, use Ingredients and Preparations will be better.

10 REVIEW 6

The group number was: 26

What was the goal of the group's project?
The goal is to do Named Entity Recognition on news article.

How did the group attempt to accomplish the goal?
They first obtained dataset from Reuters News Articles, and they used Stanford NER-tagger and POS-tagger to generate pos tags and results, they also Manually corrected some wrong tags. They used three models for NER: Averaged Perceptron, HMM + Naive Bayes, CRF. For features, they used word pattern, pos tags and some chunking results etc.

How did the group evaluate their work and what was the result?
For performance, they give F1 scores for each class and each method, it turns out that overall CRF gives the best result: Person: 0.86, Organization: 0.89, Location: 0.87, Others: 0.99

11 REVIEW 7

The group number was: 27

What was the goal of the group's project?
The objective is to detect bullying in social media from textual data such as messages, comments, posts etc.

How did the group attempt to accomplish the goal?
First, they collected data from formspring dataset where multiple instances of cyberbullying was reported. Then they cleaned data like removing special characters. They tried Baseline model, SVM and naive bayes on both Original dataset and balanced data.

How did the group evaluate their work and what was the result?
They used 3-fold cross validation and F1 score as a measure to evaluate the performance for both Bully and NonBully class. And the method Naive Bayes turns out to be best:
Original data: Bully: 40.7%, NonBully: 96%
Balanced data: Bully: 80%, NonBully: 78%

12 REVIEW 8

The group number was: 34

What was the goal of the group's project?
The goal is to implement unsupervised learning of a PCFG (Probalistic Context-free Grammar) from a corpus.

How did the group attempt to accomplish the goal?
The algorithm is based on Iterative Biclustering (PCFG-BCL) where the two steps are repeated: 1. Learn a new AND-OR group by biclustering
2. Updating the Grammar based on the learned rule. And finally, they post-processed the data by adding start rules.

How did the group evaluate their work and what was the result?
1000 sentences were generated and compare the language of the learnt grammar with that of the target grammar with precision, recall and F-score. And the final result is Precision: 41%, Recall: 100%, F-score: 58%

13 REVIEW 9

The group number was: 36

What was the goal of the group's project?
Classify the Yelp review for rating on different business category.

How did the group attempt to accomplish the goal?
First they do basic pre-processing: 1. tokenization, 2. lowercase tokens, 3. remove punctu-

ation, 4. stemming. For feature types, they used four features set: Unigram TF, Unigram TF with rare words removed, Unigram TF-IDF with rare word removed and advanced features with selective bigrams. For models, they tried Naive Bayes, Logistic regression, SVM and random forests.

How did the group evaluate their work and what was the result?
They used Mean Square Error (MSE) for evaluation, the result shows that advanced features with naive bayes gives the best result (under 0.8) while others combination of features and methods are all over 1.0.

14 REVIEW 10

The group number was: 44

What was the goal of the group's project?
Generate a course plan for the following semesters with some rules (their course history, their graduation requirements and etc) and present it to students.

How did the group attempt to accomplish the goal?
They first got fake databases (text files), and do data cleaning and formating, then extracted information from the database (student names, past history, GPA and etc), The algorithm is based on decision tree with priorities on features pre-assigned. Then they will give text output for the plan.

How did the group evaluate their work and what was the result?
They will use human ratings on 10 subjects and be based good, ok, bad scale. The results show that for fluency, 70% people say good and others feel ok. For correctness, 90% good and 10% ok. Lastly, for usefulness, 70% feel good, 20% think ok, and 10% gives bad.