Department of Computing and Electronic Systems, University of Essex

Assignment 2: Information Extraction

Submitted as part of the requirements for:

CE807 Text Analytics

Name: Henny Purnomo

Supervisor: Dr. Alba Garcia Seco De Herrera and Dr. Udo Kruschwitz

Date: 23 April 2018

Table of Contents

List of Tables

List of Figures

1 Method

This project involves similar dataset for training and testing as well as compares its f-score to this NER system in one of experiments in this paper [1].In term of training dataset, Wikiner was collected from automatically extracting the label from Wikipedia with distant supervision method. Unlike supervised learning, in distant supervision, positive examples and negative examples are being gathered for the dataset. In this report [2], they state NER system should be evaluated with a gold standard in order to ensure the accuracy of automatic extraction methods. Therefore, Wikigold has been selected to become testing dataset. Wikigold is the gold standard (manually annotated by expert), so it has exact labels for each term. Wikiner dataset has been used for training as well as Wikigold for testing the system. Because the original format of Wikiner is almost iob format, it was needed to change into the list while Wikigold was needed to add pos tag (with NLTK toolkit), reformat the structure and store it into the list. After that, extracting the features from the Wikiner dataset and feeding it into a classifier. Once Wikigold dataset has been extracted into features, then it was used to test the classifier. In addition, removing 'O' labels from the system is important to reduce the size in dataset since O entities are much more [3].This system employs F1 (also known F-score or F-measure), precision and recall to evaluate performance the classifier on the testing data.

2 Features

As aforementioned above, the data was trained with CRFSuite package. These are the features that have been employed to build the system:

- Lowercase for the current word, its previous word and next word.
- Checking whether the word is uppercase or not for the current word, its previous word and next word.
- Checking whether the word has a titlecased string at least one character or not for the current word, its previous word and next word.
- Checking whether the word is a number or not for the current word.
- Pos tag for the current word, its previous word and next word.
- The first two pos tag character for the current word, its previous word and next word.
- Bias
- Last three characters (suffix) from the current word.
- Last two characters (suffix) from the current word.
- If the current word does not have previous word, it will be labeled as BOS (Beginning of Sentence) whereas the current word does not have next word, it will be tagged as EOS (End of Sentence).

To give an example, one word ("to" from the fourth from sentence) has been taken from training set data which is from sentence 1 (The Oxford Companion to Philosophy says, "there is no single defining position that all anarchists hold, and those considered anarchists at best share a certain family resemblance."). The system will create its features like this:

- 'bias': 1.0
- 'word.lower()': 'to'
  Lowercase from the current word.
- 'word[-3:]': 'to'
  Suffix from the current word.
- 'word[-2:]': 'to'
  Suffix from the current word.
- 'word.isupper()': False
  Whether the current word is in uppercase.
- 'word.istitle()': False
  Whether the current word has a titlecased string at least one character or not.
- 'word.isdigit()': False
  Whether the current word is number or not.
- 'postag': 'TO'
  Pos tag from the current word.
- 'postag[:2]': 'TO'
  Pos tag in first two characters from the current word.
- '-1:word.lower()': 'companion'
  Lowercase from the previous word.
- '-1:word.istitle()': True
  Whether the previous word has a titlecased string at least one character or not.
- '-1:word.isupper()': False
  Whether the previous word is uppercase or not.
- '-1:postag': 'NNP'
  Pos tag of the previous word.
- '-1:postag[:2]': 'NN'
  Pos tag in first two characters from the previous word.
- '+1:word.lower()': 'philosophy'
  Lowercase from the next word.
- '+1:word.istitle()': True
  Whether the next word has a titlecased string at least one character or not.
- '+1:word.isupper()': False
  Whether the next word is uppercase or not.
- '+1:postag': 'NNP'
  Pos tag from the next word.
- '+1:postag[:2]': 'NN'
  Pos tag in first two characters from the next word.

3 Classifier

Identifying named entity in a document or text needs take into account sequence of word rather than just focus on n-gram. There are several methods in sequence learning, for example HMM (Hidden Markov Models), Maximum Entropy Models (also known Logistic Regression) and CRFs (Conditional Random Fields) [4]. According to [5], CRFs outperforms in NER task among the HMM and Maximum Entropy, with F-score 56.03%, 48.21% and 49.09% respectively on detecting Hindi language. As reported by [6], CRFs can easily integrate a huge number of arbitrary, non-independent features while still having efficient procedures for nongreedy finite-state inference and training as well as have ability to automatically construct the most useful feature combinations. In addition, it has function to maximize performance of sequence labeling by modelling the conditional distribution P(Q|O) [4]. Because of those reasons, to train this NER (Named Entity Recognition) system, CRFs algorithm was employed.

There are several toolkits to build the system using CRF such as CRF++ and CRFSuite. One of the reasons that the latter has been chosen to be implemented because it is 11.4 times faster than CRF++ [7]. There are five algorithms are used in CRFSuite, such as LBFGS (LBFGS with L1/L2 Regularization), L2SGD (SGD with L2-regularization), AP (Averaged Perceptron), PA (Passive Aggressive), ARROW (Adaptive Regularization of Weights). Each algorithm has different setting of parameters. As stated in [3], there are several discrete settings for training on CRFSuite. Thus, some experiments regarding that documentation were conducted. It can be seen on the table below.

These are some experiment of hyperparameter settings on CRFsuite, here are some important result:

No	Algorithm	Parameters	Accuracy	F1 Score
1	LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)	c1: 0.1, c2: 0.1, max_iteration: 200, all_possible_transations: True	67.1	92
2	LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)	c1: 0.1, c2: 0.1, max_iteration: 50, all_possible_transations: True	66	92
3	LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)	c1: 1, c2: 1, max_iteration: 50, all_possible_transations: True	64	92
4	LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)	c1: 0, c2: 0, max_iteration: 50, all_possible_transations: True	55	90
5	L2SGD (Stochastic Gradient Descent with L2 regularization)	c2: 1, max_iteration: 50, period: 10, delta: 1e-5	66,9	92
6	AP (Averaged Perceptron)	max_iteration: 10, value: 1e-5	66	93
7	**PA (Passive Aggressive)**	c: 1, error_sensitive:True, averaging:True, epsilon: 1e-5, max_iteration: 100	**67.8**	**93.1**
8	AROW (Adaptive Regularization Of Weight Vector)	variance:1, gamma:1, max_iteration: 100, epsilon: 1e-5	58	90

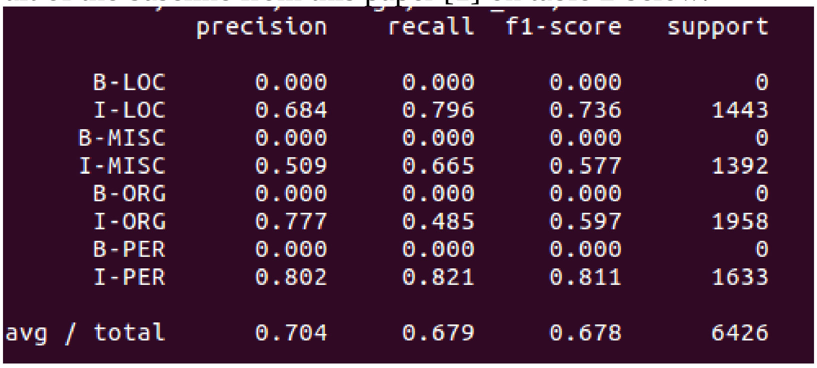
Table 1 Hyperparameter settings on CRFSuite

Based on the table above, algorithm PA (Passive Aggressive) was chosen to train this system because. Also, this setting needs less time to be compiled and achieved higher result (f1 score: 67,8%) rather than the setting of LBFGS with max_iteration:200 (f1 score: 67.1%). In this paper [8] states that CRFSuite with Passive Aggressive learning achieves the highest result. It is because the algorithm skips updates during update the weight vector when the hinge loss is zero (passive) and when the hinge loss is positive, it aggressively modifies the weight vector. In term of parameter for the classifier, it was involved several settings [9]:

- Algorithm: 'pa'
  PA stands for Passive Aggressive.
- C: 1 (default value)
  This parameter means aggressiveness parameter that controls the influence of the slack term on the objective function.
- Error_sensitive:True (default value)
  If it sets to true, means the optimization routine includes into the objective function the square root of the number of incorrect labels predicted by the classifier.
- Averaging: True (default value)
  If it sets to true, means the optimization routine computes the averages of feature weights at all updates in the training process.
- Epsilon: 1e-5 (default value)
  This parameter is determining the condition of convergence.
- Period: 10 (default value)
  This parameter is give the duration of iterations to test the stopping criterion.
- Max_iteration: 100 (default value)
  To optimize algorithm, this parameter sets the maximum number of iterations.

4 Comparison

After choosing the best parameter setting (Passive Aggressive algorithm) for the model, it evaluated with Wikigold dataset. The result can be seen on figure 1. Also, it compares with the result of the baseline from this paper [1] on table 2 below.



```
             precision    recall  f1-score   support

      B-LOC      0.000     0.000     0.000         0
      I-LOC      0.684     0.796     0.736      1443
     B-MISC      0.000     0.000     0.000         0
     I-MISC      0.509     0.665     0.577      1392
      B-ORG      0.000     0.000     0.000         0
      I-ORG      0.777     0.485     0.597      1958
      B-PER      0.000     0.000     0.000         0
      I-PER      0.802     0.821     0.811      1633

avg / total      0.704     0.679     0.678      6426
```

Figure 1 Result of NER system

It is clear to see that for B-LOC, B-MISC, B-ORG, B-PER scores are 0. It is because on training dataset has those labels but on testing set does not have it.

NE Type	CRF			Baseline (Logistic Regression)		
	Precision	Recall	F score	Precision	Recall	F score
LOC	68.4	79.6	73.6	**70.8**	**80.9**	**75.5**
MISC	**50.9**	**66.5**	**57.7**	43.0	58.0	49.0
ORG	**77.7**	**48.5**	**59.7**	63.0	48.0	54.0
PER	**80.2**	82.1	81.1	80.0	**84.0**	**82.0**
Total	**70.4**	67.9	**67.8**	64.6	**68.7**	66.6

Table 2 Comparison the model with baseline

As can be seen in the table above, for overall F-score performance, this system which is used CRF, achieve higher result (1.2% higher) than baseline. Exploiting CRF more likely well perform for detecting MISC, ORG (8.7% and 5.7% higher respectively) than using logistic regression. However, in term of recognising LOC and PER, baseline model slightly performs better (1.9% and 0.9% higher respectively). In respect of overall precision, CRF attains better result 5.8% higher than baseline, but on recall, CRF reaches lower 0.8%. Interestingly, on PER label, CRF obtains lightly higher score of precision (0.2%) but on recall and F-score are lower than baseline.

5 Discussion

Compare to the baseline, we have conducted with same dataset for training as well as testing with distant learning method, but different approach for the classifier. In the baseline system, they use Logistic Regression while in this project exploits CRF. It can be seen, CRF performs slightly better than Logistic Regression. Basically, both of the classifiers well perform on sequence learning. Logistic Regression normalize every time step independently which leads to problem known label bias, whereas CRF fixes it by taking the score of whole sequence before normalizing it [10]. So, that is make sense if CRF outperform Logistic Regression.

Regarding the setting of parameters in CRFSuite, several experiments have been conducted, mainly focus on the mostly used algorithm LBFGS. But, changing its hyperparameters such as c1, c2, max_iteration does not obtain higher result than the default setting of Passive Aggressive algorithm. As previously mentioned in one of papers with regards to Aggressive Passive algorithm, it modifies the weight only if the hinge loss is positive.

References

[1] N. R. W. R. T. M. J. Nothman, “Learning multilangual named entity recognition from Wikipedia”.

[2] N. R. J. N. T. M. J. R. C. D. Balasuriya, “Named Entity Recognition in Wikipedia,” *Proceedings of the 2009 Workshop on the People’s Web Meets NLP, ACL-IJCNLP 2009,* pp. 10-18, 2009.

[3] N. Okazaki, “CRFsuite - Documentation,” 25 May 2016. [Online]. Available: http://www.chokkan.org/software/crfsuite/manual.html. [Accessed 17 April 2018].

[4] A. G. Udo Kruschwitz, *CE807 - Text Analytics, Lecture 5: Information Extraction Named Entity Recognition,* Colchester, 2018.

[5] M. e. a. Agarwal, “Comparative Analysis of the Performance of CRF, HMM and MaxEnt for Part-of-Speech Tagging, Chunking and Named Entity for a Morphological rich language,” *Proc. of Pacific Association For Computational Linguistics,* pp. 3-6, 2011.

[6] W. L. a. A. McCallum, “Rapid Development of Hindi Named Entity,” *ACM Transactions on Asian Language Information Processing,* vol. 2, no. 3, pp. 290-294, 2003.

[7] N. Okazaki, “CRFsuite - CRF Benchmark test,” 25 May 2016. [Online]. Available: http://www.chokkan.org/software/crfsuite/benchmark.html. [Accessed 17 May 2018].

[8] K. B. Leon Derczynski, “Passive-Aggressive Sequence Labeling with Discriminative Post-Editing”.

[9] M. Korobov, “Sklearn-crfsuite Documentation, Release 0.3,” Feb 05, 2018.

[10] M. P. Reddy, *What are the pros and cons of these three sequence models: MaxEnt Markov Model, Conditional random fields, and recurrent neural networks?,* 2017.