Classifier Combination for Telegraphese Restoration

Leo Willyanto Santoso
Department of Computer Science
Petra Christian University
Siwalankerto 121-131 Surabaya, Indonesia
e-mail: leow@petra.ac.id

Abstract—**This paper presents a classifier combination to solve telegraphese restoration problem. By implementing more than one classifier, it can support other classifier, and finally it can improve the performance. Using supplied development data, training data and testing data, the best model had an accuracy F = 79 %.**

Keywords: Classifier combination, telegraphese restoration, Penn Treebank tagset, chunk parsing

I. Introduction

Telegraphese restoration is an interesting topic in machine learning. It is used to restore "telegraphese" to its original form; focusing on case and punctuation. For example, given the following text:

pawang or medicine man johari albert 78 said he had a dream thursday

The correct text as follows:

Pawang, or medicine man, Johari Albert, 78, said he had a dream Thursday

In this research, Penn Treebank Tagset, Chunk and Claws 7 in 5 different taggings, across 5 positions in the window are used. All data will be made available in ARFF format, with a single instance per word token in the source text. For each word token, an instance index, the word, and the class label are provided. The following is a data sample:

```
245,pawang,cap1+comma
246,or,nochange
247,medicine,nochange
248,man,nochange+comma
249,johari,cap1
250,albert,cap1+comma
251,78,nochange+comma
252,said,nochange
253,he,nochange
254,had,nochange
255,a,nochange
256,dream,nochange
257,thursday,cap1
```

The classes describe first necessary changes to the capitalisation of the word, and then any insertions of punctuation to the end of that word. Note that a given word may require multiple changes to its capitalisation, and also potentially multiple punctuation insertions, and that the atomic class labels are additive. The detailed information about class and description can be seen in Table I.

TABLE I. Class and Description

Class	Description	Example
nochange	No changed required	Nochange(medicine) → medicine
allcaps	Convert to all caps	allcaps(unesco) → UNESCO
capN	Convert the letter at position N to upper case	cap1(thursday) → Thursday
+comma	Insert a comma at the end of the word	nochange+comma(78) → 78,
+fullstop	Insert a full stop at the end of the word	nochange+fullstop(popularity) → popularity
+colon	Insert a colon at the end of the word	nochange+colon(it) → it:
+semicolon	Insert a semicolon at the end of the word	nochange+semicolon(10) → 10;
+exclmark	Insert a exclamation mark at the end of the word	nochange+exclmark(proof) → proof!
+questmark	Insert a question mark at the end of the word	nochange+questmark(man) → man?

The research will take the form of a shared task: pre-classified training and development data are provided, to use in feature engineering and the classifier development. After that, unannotated test data will be provided. The output of this research is final classifiers over that data.

In this paper, we present a potential approach for improving the performance of telegraphese restoration by using classifier combination techniques such as bagging and boosting. To the best of our knowledge, this is the first effort that utilizes classifier combination for improving telegraphese restoration.

Combination methods have been applied to many problems in natural-language processing (NLP). For examples: ROVER system for speech recognition [3], the Multi-Engine Machine Translation (MEMT) system [7], and improving lexical disambiguation [1]. Most of these techniques have shown a considerable improvement over the performance of a single classifier and therefore, considering implementation such a multiple classifier system for telegraphese restoration as well decision.

Using classifier combination techniques one can potentially achieve a classification accuracy that is superior to that of the single best classifier. This is based on the assumption that the errors made by each of the classifiers are not identical, and therefore if we intelligently combine multiple classifier outputs, we may be able to correct some of these errors

The remaining part of this paper is organised as follows. Section 2 presents an overview of current proposals for dealing with natural language processing. Section 3 depicts the approach that we have delineated to solve proposed problems. Section 4 discusses the performance of proposed method. Finally, section 5 concludes the paper

II. BACKGROUND AND RELATED WORK

In this section, the previous work of Part of Speech (POS) tagging, Penn Treebank tagset, chunk parsing are presented.

A. *Part of Speech (POS) Tagging*

Part of Speech (POS) Tagging is the process that classifies word into several categories based on its definition and context. POS tagging is now done in the context of computational linguistics; using algorithms which associate discrete terms, as well as hidden parts of speech, in accordance with a set of descriptive tags.

Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times [6].

B. *Penn Treebank Tagset*

Penn Treebank Tagset is developed by researchers of Computer Science and Information Department at the University of Pennsylvania. It annotates naturally; occurring text for linguistic structure. Table II shows the sample of Treebank Tagset.

TABLE II. SAMPLE TREEBANK TAGSET

Tagset	Description
CC	Coordinating conjunction. e.g. and, or, but
CD	Cardinal number
DT	Determiner
EX	Existential *there*
FW	Foreign word
…	…

The complete tagset can be browsed from this URL[1].

C. *Chunk Parsing*

Chunk Parsing or Shallow Parsing is an important step to extract word into linguistic fragment [4]. Compared to full parsers that would fail to deliver any (even partial) linguistic information if the whole utterance cannot be completely analysed in accordance with some competence model of the particular language, this parsing method is robust (since it always delivers some linguistic information).

A chunk parser attempts to model human parsing by breaking the text up into small pieces; each parsed separately. There are some advantages of chunk parsing, such as: better modeling human behaviour. Moreover, chunk parsing is fast because it only deals with small part without recursion process

III. CLASSIFIER COMBINATION

Running this experiment, let's go through steps as follows. The first task is formatting input and output of classifier; this task was done by developing simple software in Java. Next, the final classifier is performed to classify formatted input.

The final classifier was developed using Weka. The Weka workbench itself is a collection of modern machine learning algorithm and data pre processing tools. It includes most of state-of-the-art algorithms for doing classification, including preparing the input data, evaluating learning schemes statistically, and visualizing the input data and the result of learning.

A. *Feature Engineering*

Successful machine learning method involves not only selecting a learning algorithm, but transformations engineering between input and output is also important [2, 5]. In this experiment, the author performed data transformation, feature selection, cleansing data and detecting outliers. Let consider the following fragment of data set (development.data, train.data, test.data) and feature set (development.features, train.features, test.features) as can be seen in Table III and Table IV respectively.

TABLE III. SAMPLE DEVELOPMENT DATA

ID	Word	Class
1	are	cap1
2	we	nochange
3	going	nochange
4	to	nochange
5	remember	nochange

TABLE IV. SAMPLE DEVELOPMENT FEATURES

ID	Word	Tagging
1	be	VBP, _, _, _, _, O, _, _, _, _, _, _, _, _, VBR
2	we	PRP, PRP, _, _, _, B-NP, B-NP , _, _, _, _, _, _, PPIS2, PPIS2
3	go	VBG, VBG, VBG, _, _, B-VP, B-VP, B-VP, _, _, _, _, VVGK, VVGK, VVGK
4	to	TO, TO, TO, TO, _, B-VP, I-VP, I-VP, I-VP, _, _, TO, TO, TO, TO
5	remember	VB, VB, VB, VB, VB, B-VP, I-VP, I-VP, I-VP, I-VP, VV0, VV0, VV0, VV0, VV0

With respect to the data condition above, joining both of them are necessary. By developing simple java program,

[1] http://www.computing.dcu.ie/~acahill/tagset.html

both files are joined together. The result of this process is described in Table V.

TABLE V. OUTPUT OF JOIN FILE

ID	Word	Tagging	Class
1	be	VBP, _, _, _, _, O, _, _, _, _, _, _, _, _, VBR	cap1
2	we	PRP, PRP, _, _, _, B-NP, B-NP , _, _, _, _, _, _, PPIS2, PPIS2	nochange
3	go	VBG, VBG, VBG, _, _, B-VP, B-VP, B-VP, _, _, _, _, VVGK, VVGK, VVGK	nochange
4	to	TO, TO, TO, TO, _, B-VP, I-VP, I-VP, I-VP, _, _, TO, TO, TO, TO	nochange
5	remember	VB, VB, VB, VB, VB, B-VP, I-VP, I-VP, I-VP, I-VP, VV0, VV0, VV0, VV0, VV0	nochange

By applying Part of Speech (POS) tagging approach, Tagging column in Table 4 need to be split to better understanding of each attributes that has correspondents to class. The result of this task is depicted in Table VI as follows.

TABLE VI. OUTPUT OF JOIN FILE

Instance 1

Attribute	Value
ID	1
Word	be
Penn-1	VBP
Penn-2	_
Penn-3	_
Penn-4	_
Penn-5	_
Chunk-1	O
Chunk-2	_
Chunk-3	_
Chunk-4	_
Chunk-5	_
Claw-1	_
Claw-2	_
Claw-3	_
Claw-4	_
Claw-5	VBR
Class	nochange

Instance 2

Attribute	Value
ID	2
Word	we
Penn-1	PRP
Penn-2	PRP
Penn-3	_
Penn-4	_
Penn-5	_
Chunk-1	B-NP
Chunk-2	B-NP
Chunk-3	_
Chunk-4	_
Chunk-5	_
Claw-1	_
Claw-2	_
Claw-3	_
Claw-4	PPIS2
Claw-5	PPIS2
Class	nochange

Using Weka tool, data with missing value are easily deleted. After running function deleteWithMissingClass() using Java+Weka, the quality of input data was gotten.

B. *ARFF Format*

The standard format in representing data set in Weka is ARFF file. To produce ARFF file, the author uses generate-arff script, was developed in perl language.

C. *Feature Selection*

Most of learning algorithms are designed to learn which are the most appropriate attributes to use for making best decisions. The negative effect of irrelevant attributes on the system is it will eliminate and make ambiguity of other attributes which could be the appropriate one [2]. The best way to select relevant attributes is manually, based on the understanding of the problem context and what the attributes mean.

In this problem, when we used all attributes (ID, word, all Tagging, and class), the system looks like give up to produce the output. Attribute ID is not relevant to the machine learning system, and attribute word is harmful, because if we use it as an input, our machine learning may to remember the word only. For example, if in the data set we have 10 words "we", 7 is categorised as nochange class, 2 as cap1 and 1 as nochange+comma, the built system will produce output for word we as the member of nochange class.

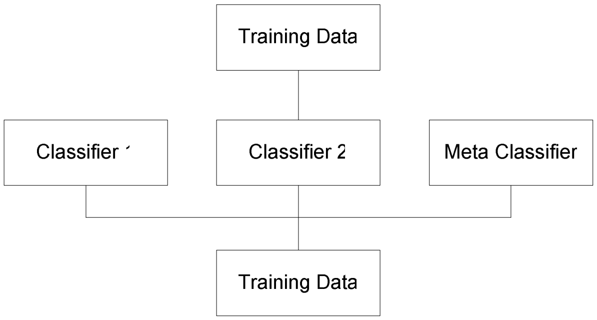
After eliminating 2 attributes, further investigating to other attributes are still crucial. Attribute selection could be done by machine learning algorithm [2]. First, the author tries to apply decision tree algorithm to the full data set, and then select only those attributes that are actually used in tree. Unfortunately, because of big data set (65,000 instances), the system always goes to not responding and out of memory.

Another way to doing selection is by using instanced-based learning method. The author takes sample of instances randomly from data set, then doing "near hits" and "near misses" analysis.

The result is, 4 attributes can be eliminated, 2 from chunks attribute and 1 from claw attribute and 1 from penn attributes.

D. *Classifier Architecture*

In this experiment, author implements classifier combination that consists of two classifiers and 1 meta classifiers to replace vote system in stacking design.



Classifier 1 = IB1
Classifier 2 = ZeroR
Meta Classifier = IB1

Figure 1. Design of Classifier Combination

To combine outputs, we use meta classifier rather than voting, because sometimes the majority prediction from several classifier is incorrect. Actually, voting is good if we have one classifier as an "expert", that we can trust.

Actually, decision tree is quite better classifier, unfortunately, author can't implement tree classifiers, because of big data sets, and so the memory is going to low.

E. *Output*

The performance of classifier combination within supplied development and training data set is described in the Table VII.

TABLE VII. RESULT OF DEVELOPMENT

Classes	F
allcaps	0.81
allcaps+fullstop	0.88
cap1	0.70
cap1+colon	0.72
cap1+fullstop	0.89
cap1-3	0.73
nochange	0.81
nochange+colon	0.64
nochange+comma	0.76

Overall Accuracy: F = 0.79

IV. EXPERIMENTS

From the result, classifier combination is quite good to solve telegraphese problem. Actually, after doing comparison with the same input data but using single classifier (IB1), overall accuracy increases significantly. IB1 has F score around 34% and ZeroR has F score around 42%.

Improvements could be done by focusing on the R score. Applying other machine learning algorithm could assist this problem. To get better performance, we have spent more time at pre-processing task. We can do like manipulating training set, manipulating learning algorithm, manipulating input features and class as well. In manipulating the training set, we can do sampling from original data, and then we develop classifier for each training set.

In manipulating input features and class can be performed. In this approach, we can focus on the weak performance result of class. For example, in this experiment, we split the class into two groups. The first group contains all class that have performance F > 0, and the second class who have F score = 0. Then, investigating for each attributes are necessary to reduce redundancy and ambiguity. By applying symmetric uncertainty formula, redundant attributes could be detected. In manipulating learning algorithm, we can perform the experiment several times with the same data and algorithm.

V. CONCLUSION

Overall, classifier combination performs well than single classifier. Each classifier has advantage and disadvantage, by combining more than one classifier algorithm, better result will we get.

As was argued in experiment part, feature engineering is very important task in machine learning. There are number of improvements which could be done to these models such as re-feature selection to reduce over training and manipulating learning algorithm. These tasks can be expected to give modest boost to the performance of the best model.

ACKNOWLEDGMENT

The author would like to acknowledge Timothy Baldwin, The University of Melbourne, Australia, for his guidance and support in conducting the research presented in this paper.

REFERENCES

[1] E. Brill and J. Wu, “Classifier Combination for Improved Lexical Disambiguation”, Proc. of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics – Volume 1 (COLING ‘98), 1998, pp 191-195, doi: 10.3115/980451.980876.

[2] I. Witten, E. Frank and M. Hall, “Data mining: Practical Machine Learning, Tools and Technique”, 3rd ed. Morgan Kaufmann Publishers, 2011.

[3] J. Fiscus. A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER). Proc. of Automatic Speech Recognition and Understanding (ASRU 1997), Dec. 1997, pp. 347-354, doi: 10.1109/ASRU.1997.659110.

[4] P. Brooks, “SCP: A Simple Chunk Parser”, Artificial Intelligence Center, The University of Georgia. Athens, Georgia, 2003.

[5] P. Tan, M. Steinbach and V. Kumar, “Introduction to Data Mining”, Adison Wesley, 2006.

[6] R. Navigli, “Word Sense Disambiguation: A Survey”, ACM Computing Surveys (CSUR), vol. 41, issue. 2, Feb. 2009, pp. 1-69, doi: 10.1145/1459352.1459355.

[7] S. Jayaraman and A. Lavie. “Multi-engine Machine Translation Guided by Explicit Word Matching”, In Proc. of the ACL Interactive Poster and Demonstration Sessions, June 2005, pp. 101-104, doi: 10.3115/1225753.1225779.