EDNII Error Detection Tool User Manual

Version 1.0

August, 2012

Ngan L.T. Nguyen, Yusuke Miyao

1. Introduction

EDNII is an error detection tool for ESL student essays. EDNII was originally developed for the Error Detection and Correction Workshop 2012 (EDCW 2012), and is released under MIT License.

EDNII implements three different error detection methods: discourse-based, classifier-based, and language model based method, which are called Discourse Model, Classifier Model and Islam Model correspondingly. For the dry run, Discourse Model was used for the Verb-agreement track and Open track, and Islam Model was used for the Preposition track. However, for the formal run, Classifier Model was used in place of Islam Model.

Discourse Model uses rules for both sentence analysis and error detection, based on POS-tagging and chunking results. This model aims to detect subject-verb agreement, verb tense, article, and noun number errors. The detection for subject-verb agreement errors was used for the Verb-agreement track, and that for all types of errors was included in the Open track.

Classifier Model is a reimplementation of Tetreault et al.'s system described in [1]. Similarly, Islam Model is a reimplementation of Islam et al.'s system described in [2]. However, perhaps due to some differences in implementation, the EDNII's performances are not so good as those reported in the papers. We suggest that further analysis should be done for these two models.

More detailed descriptions of EDNII are given below.

2. EDCW2012 configurations and results

Dryrun

	Method	F	P	R

Verb-track	Discourse Model	0.416	0.619	0.313
Preposition-track	Islam Model	0.093	0.128	0.073
Open-track	Discourse Model	0.230	0.310	0.183

Formal run

	Method	F	P	R
Verb-track	Discourse Model	0.432	0.571	0.348
Preposition-track	Classifier Model	0.227	0.171	0.336
Open-track	Discourse Model	0.122	0.192	0.089

3. System notes

- Language: Java
- Entry: main() function of jp.ac.nii.ednii.ErrorCorrection
- Environment: Run ErrorCorrection with the parameter: -Dwordnet.database.dir=[./WordNet-3.0/dict]
- All method descriptions below use the sample data in the examples directory.
- Prerequisites:
  - WordNet-3.0/dict files are placed in EDNII/WordNet-3.0/dict.
  - Google Web 1T is unzipped in: ./web_1T_5-gram_v1/unzipped/index-1gms,...

4. Discourse-based method (Discourse Model)

- Run ErrorDetection for a collection of documents with the following parameters:

args[0] = discourse
args[1] = collection
args[2] = [INPUT DIR]: preprocessing files of each document, which are placed in the same sub-folder of the document as in KJCorpus.
args[4] = [EXTENSION OF POS-CHUNK FILES]
args[5] = [EXTENSION OF OUTPUT FILES]
args[6] = [OUTPUT DIR]: contains output files with the detected learner errors.

- Example:

```
    java -Dwordnet.database.dir=./WordNet-3.0/dict
jp.ac.nii.ednii.ErrorCorrection discourse collection ./examples/
test .edc .gen ./examples/discourse/output
```

- Method details:
  - The system analyzes the structure of all sentences of an input document to find dependent and independent clauses. Then, for each clause, the system finds its subjective noun phrase, main verb phrase, temporal clue phrase, … (jp.ac.nii.ednii.ChunkedSentenceAnalyzer.analyzeStructure). Based on the sentence structure, rules are applied to detect different types of errors.
  - Verb-tense errors: The system implements a simplified version of Temporal Centering theory [3], in which the tense of a clause is considered to be continued in the following clause, if there is no tense change that is signified by the temporal clue associated with that following clause. String-based patterns are used to determine the tentative tense implied by a temporal clue . For example, “three years ago” will go with the past tense. If a conflict between the

tenses of temporal expression and verb of a sentence occurs, a verb-tense error is issued.
    - Subject-verb agreement errors: The system simply checks the agreement between numbers of the head noun of the subjective noun phrase and the verb. If they do not agree in number, then a verb-agreement error is created.
    - Article and noun number errors: The system uses basic grammar rules of the agreement of an article and its govern noun in number and countability. Noun countability is determined based on an in-house morphology dictionary which are extracted from POS-tagged North American News corpus. However, we found that this dictionary does not cover many common forms of common nouns, which causes mistakes in error detection.
- Selling point:
    - Verb tense errors are very difficult to correct since they are related to the tentative temporal semantics of the sentence. We think that by applying Temporal Centering theory, our model can detect abrupt changes in discourse level, and problems related to the use of temporal device of learners' writings.

5. Classifier-based method (Classifier Model)

- Run ErrorDetection for a collection of documents with the following parameters:

args[0] = classifier
args[1] = predict-esl-esl/ predict-native-esl/ predict-native-native
args[2] = [TEMP DIR]: contains temporary files which can be used for further analysis.
args[4] = [MODEL PROPERTY FILE]: model description file in StanfordClassifier format.
args[5] = [MODEL FILE]: model file in StanfordClassifier format.
args[6] = collection
args[7] = [OUTPUT DIR]: contains output files with the detected learner errors.

- Example:

```
      java -Dwordnet.database.dir=./WordNet-3.0/dict
jp.ac.nii.ednii.ErrorCorrection classifier predict-native-
native ./examples/temp ./examples/classifier/model/edcw.prop ./
examples/classifier/model/edcw-ngan.model collection ./examples/
test .txt .gen ./examples/classifier/output
```

- Train a new model:

args[0] = classifier
args[1] = train-esl/ train-native:
args[2] = [INSTANCE FILE]: output instance files, which will be used for training a StanfordClassifier model.
args[4] = collection
args[5] = [EXTENSION OF GOLD-STANDARD FILES]: .txt files in KJCorpus
args[6] = [EXTENSION OF OUTPUT FILES]: unimportant parameter
args[7] = [TEMPORARY DIR]: unimportant parameter

- Example:

```
     java -Dwordnet.database.dir=./WordNet-3.0/dict
jp.ac.nii.ednii.ErrorCorrection classifier train-native ./examples/
classifier/model/esl-instances.train collection ./examples/
train .txt .gen ./examples/classifier/temp
```

- Method details:

    - Similar to Tetreault et al.'s model [1]
    - The idea of the system can be summarized as follows. A multiclass classifier is used to select a correct preposition (among a set of prepositions) for any preposition appearing in a document.
    - The classifier can be trained on native data, and used to predict preposition errors on both native data and learner data. However, the use of the classifier in the two settings, native and learner data, are slightly different. "Train-native", "predict-native-native", and "predict-native-esl" are designed in the way similar to the paper. "Train-esl" and "predict-esl-esl" are different in that they consider DELETE, and INSERT errors.
    - One of the limitations of this model is that it cannot correct INSERT preposition errors.
    - For the dryrun submission, "train-native" and "predict-native-native" options were used. "Predict-native-native" is different from "predict-native-esl" in that the former choose the best preposition solely based on the classifier output, while the later is based on the comparison between scores of learne's preposition and the best prediction of the classifier as in [1].
- Selling point:
    - The classifier method is very common for correcting preposition errors.

6. Language model-based method (Islam Model)

- Run ErrorDetection for a collection of documents with the following parameters:

args[0] = islam
args[1] = [Google Web 1T unzipped 1gms]
args[2] = [Google Web 1T unzipped 2gms]
args[4] = [Google Web 1T unzipped 3gms]
args[5] = [Google Web 1T unzipped 4gms]
args[6] = [Google Web 1T unzipped 5gms]
args[7] = collection
args[8] = [INPUT DIR]: preprocessing files of each document are put in the same sub-folder of the document as in KJCorpus
args[9] = [EXTENSION OF INPUT FILES]
args[10] = [EXTENSION OF OUTPUT FILES]
args[11] = [OUTPUT DIR]

- Example:

```
    java -Dwordnet.database.dir=./WordNet-3.0/dict
jp.ac.nii.ednii.ErrorCorrection islam ./web_1T_5-gram_v1/unzipped/
index-1gms ./web_1T_5-gram_v1/unzipped/index-2gms ./web_1T_5-gram_v1/
unzipped/index-3gms ./web_1T_5-gram_v1/unzipped/index-4gms ./web_1T_5-
gram_v1/unzipped/index-5gms collection ./examples/test .txt .gen ./
examples/islam/output
```

- Method details:
    - This is an implementation of Islam's model [2]. However, it should be noted that we only include preposition errors in our implementation of EDNII to serve our preliminary experiments.
- Selling point:
    - Islam model is claimed to be able to correct multiple types of errors. External data and tools

a. Stanford Classifier

URL: http://nlp.stanford.edu/software/classifier.shtml

Purpose: To train the multi-class classifier used in Classifier Model

b. Stanford Parser

URL: http://nlp.stanford.edu/software/lex-parser.shtml
Purpose: For feature extraction for Classifier Model.

c. WordNet

URL: http://wordnet.princeton.edu/
Purpose: used by Jaws (see below).
Notes: Run ErrorCorrection with this parameter:
-Dwordnet.database.dir=[./WordNet-3.0/dict]

d. Jaws

URL: http://lyle.smu.edu/~tspell/jaws/index.html
Purpose: Java API for WordNet Searching (JAWS); used for lemmatization.

e. JWeb1t

URL: http://code.google.com/p/jweb1t/
Purpose: A tool for searching Google Web 1T 5-gram corpus, used by Islam Model.

f. Google Web 1T

URL: http://www.ldc.upenn.edu/Catalog/docs/LDC2006T13/readme.txt
Purpose: Used by Islam Model to get 5 gram counts.

g. OpenNLP

URL: http://opennlp.sourceforge.net/projects.html
Purpose: For preprocessing native training data for Classifier Model.

h. GeniaTagger

URL: http://www.nactem.ac.uk/GENIA/tagger/
Purpose: Part-of-speech tagging and base-phrase chunking.

7. References

[1] Tetreault Joel, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In Proceedings of the ACL 2010 Conference Short Papers.
[2] Aminul Islam and Diana Inkpen. 2011. Correcting different types of errors in text. In C. Butz and P. Lingras (Eds.): Proceedings of the 24th Canadian Conference on Artificial Intelligence, Canadian AI 2011, LNAI 6657, Springer, pp. 192-203, St. John's, Canada.
[3] Megumi Kameyama, Rebecca Passonneau, and Massimo Poesio. 1993. Temporal centering. In Proceedings of the 31st annual meeting on Association for Computational Linguistics (ACL '93). Association for Computational Linguistics, Stroudsburg, PA, USA, 70-77.