Additional Instructions for Verb Prediction Task

February 10, 2009

1 Training Corpus

The goal of this problem set is to write a program that will predict the most likely verb given a specific context. Specifically, provided with text where a single verb has been omitted, your program is expected to select the most appropriate verb for placement in the text. You will investigate two general approaches to this problem: a language model approach and a classification-based approach. In each case, there will be a set of mandatory experiments and ample room for additional experimentation if you are interested in studying a specific question and/or improving your results on this specific task.

You have been provided with a training corpus, train.data, consisting of a collection of sentences augmented with additional syntactic information, namely part of speech and chunking (shallow parsing) tags. There are 8936 training sentences, consisting of approximately 200,000 words. Based on statistics over this corpus, you have been provided with a list of verbs, verb_list, that we are concerned with, consisting of 925 different verbs derived from 256 base verb forms. Since multiple verbs can occur in a single sentence, there are 16,195 training examples contained in the given training corpora. As an example of the requisite task, we will use an example sentence in the representation provided by a table format.

| X | X | 0 | CC | O | But |
| X | X | 1 | NN | B-NP | consumer |
| X | X | 2 | NN | I-NP | expenditure |
| X | X | 3 | NNS | I-NP | data |
| released | release | 4 | VBD | B-VP | released |
| X | X | 5 | NNP | B-NP | Friday |
| X | X | 6 | VBP | B-VP | do |
| X | X | 7 | RB | I-VP | n’t |
| suggest | suggest | 8 | VB | I-VP | suggest |
| X | X | 9 | IN | B-SBAR | that |
| X | X | 10 | DT | B-NP | the |
| X | X | 11 | NNP | I-NP | U.K. |
| X | X | 12 | NN | I-NP | economy |
| X | X | 13 | VBZ | B-VP | is |
| slowing | slow | 14 | VBG | I-VP | slowing |
| X | X | 15 | DT | B-ADV | that |
| X | X | 16 | RB | I-ADV | quickly |
| X | X | 17 | . | O | . |
There are six columns for each token in a sentence. The first column signifies the target for the verb prediction task and the second column signifies the target for the base verb prediction task. You will run experiments for both tasks independently. If the first two columns are populated by $x$, then that word is not a target word for our specific task. The third column is the serial number of the word in the sentence, starting from 0. The fourth column is the POS tag for the respective word and the fifth column signifies the chunking information. Note that this syntactic information was tagged by a machine learning program and therefore, there is noise in the syntactic information. However, the testing data will be tagged using the same process, so the noise will likely be correlated between the training and test data. Finally, the sixth column is the actual text. Therefore, for this example sentence, there are three training examples, noting that the assumption is that only one word will be removed at a time.

Naturally, you may wish to use additional information sources to tag the training data, including information such as named-entity, word sense disambiguation, WordNet information etc. You are welcome to do this, but you must provide a script such that we can replicate your results and perform the same process on new data.

Since we have constrained the required task by providing a finite number of verbs, our immediate formalization of the problem is a multiclass classification task where the number of verbs (or verb base forms) constitutes the number of classes.

## 2 Language Model

The first part of the assignment is to develop a program that can perform the verb prediction task using a language model. Some minimal requirements of the language model is that it will be based on

- Word level unigram, bigram, and a back-off model.
- POS tag level unigram, bigram, and a back-off model.
- Consider both left sided information and a symmetric window.

While you may be inclined to use language modeling toolkits to complete this part of the assignment, you are not allowed to submit any source code from these toolkits nor can your solution be dependent on these toolkits as there is a lot to be learned about the subtleties of language modeling by coding a system independently.

## 3 A Classification Based Predictor

In the second part of the assignment, you will build upon your experience on utilizing a language model to perform verb prediction using a supervised learning paradigm. The specific learning architecture you will use is SNoW. If you are not familiar with SNoW, you should refer to the manual. It is also recommended that you use the FEX program to extract features from the data for use in the learning procedure. There are many options for both FEX and SNoW, requiring you to perform some experimentation to achieve good results. However, your prediction must be minimally based on

- Word level features, including conjunctions of size $\geq 2$.
- POS level features, including conjunctions of size $\geq 2$. 
• Mixed POS level and word level features.
• Consideration of both a left sided model and a symmetric window.

You may need an additional layer of processing between FEX and SNoW. This is allowed, but once again, your scripts will have to provide the requisite functionality such that this same processing of the feature vectors can be repeated and performed on new, unseen data.

4 Software Requirements

Up to this point, our description of the program requirements have been somewhat abstract. The intention was to provide a clear presentation of the necessary tasks and to give you flexibility in thinking about your design. However, since we also will have to run your programs, we have to place some restrictions (hopefully minimal) on your program design.

First of all, we require that you complete this assignment in a Linux/Mac environment. You are required to submit three shell scripts, a method for compiling your source code, and running it.

4.1 Compilation

The preferred method for compiling your source code would be a single Makefile, such that we could simply type `make` in the submitted directory and everything would compile. However, since we are not restricting the programming language, compilation methods may differ. Therefore, we ask that you create a simple method for us to compile your source code. Seemingly possible options would include makefiles and/or shell scripts, but the only requirement is that it is easy for evaluation purposes.

4.2 Modifying Data

As stated, you may wish to add information to the given data or modify the data provided into a more usable form. Also, in the case of the learning task, we require you to perform feature extraction. Therefore, we ask that everybody submit a shell script called `process_data.sh` that takes at least three parameters, one specifying the input data file (such as the provided data), one specifying which part of the assignment is being performed, and one specifying which experiment is being performed.

If we want to run the data modification script for the language modeling task on the training data for the base verb prediction task, the command line should be

```
./process_data.sh -lm -base train.data
```

and for the supervised learning approach with the full verb forms

```
./process_data.sh -snow -full train.data
```

The output created by this script can be in any form, as long as it is described in your short report and usable by the other requisite scripts. Note that for the learning task, this script must not only preprocess the data, but run FEX and also perform any additional processing before the data is ready for use by SNoW.
4.3 Learning / Estimating Parameters

The second script we require is the script for either calculating statistics in the case of the language model or training SNoW in the case of supervised learning. The name of this script should be \texttt{train.sh} and we will require three command line parameters as stated in the data modification section. Note that we will use the original name of the training data here, so your script must modify variables accordingly to accommodate this.

4.4 Testing

The final script we require is the testing script, which we will call \texttt{test.sh}, which requires three parameters, namely which part of the assignment we are testing, which experiment is being performed, and the name of the testing data file. We will describe the output format requirements and evaluation methods later.

5 Evaluation

The verb-prediction problem is a difficult problem and therefore, a more complex evaluation metric is proposed than just measuring the accuracy.

5.1 Output Format

We require that your testing script \texttt{test.sh} generate output of the form

\[
\text{verb\_prediction1 verb\_prediction2 verb\_prediction3}
\]

for each target instance where \texttt{verb\_prediction1} is your most confident prediction, \texttt{verb\_prediction2} is your second most confident prediction, and \texttt{verb\_prediction3} is your third most confident prediction. You will be provided with a perl script that awards 1 point if \texttt{verb\_prediction1} is correct, 0.5 points if \texttt{verb\_prediction2} is correct, 0.25 points if \texttt{verb\_prediction3} is correct, and 0 points if none of the predictions are correct. Note that the format is basically three single-space separated verbs on each line corresponding with the order of examples in the data file. You may submit less than three predictions, but obviously, you will not receive the corresponding partial credit. To evaluate your results, simply type

\[
\text{perl evaluate_vp.pl predictions answers}
\]

where \texttt{predictions} is the result of the testing script and \texttt{answers} is the file containing the correct verb predictions. Please make sure that this script works with your data before you submit your code, even if your evaluation on the training data is more sophisticated (possibly including testing on the training data and cross validation methods). The resulting metric by which team results will be compared is average points per example (1), which represents the classification error with an arcane error function. You may also wish to report computational requirements, such as time and memory, if you believe them to be interesting.

\[
score = \sum_i \text{points}(i)
\]

The minimal set of experimental results that must be included in the reports are:
As previously implied, we also recommend that you report results either by establishing a training and validation set within the training data or a cross validation approach to estimate the expected performance on the test data. Secondly, for the final report, you must minimally report the same cases trained on the training data and tested on the testing data. In o.w., you are expected to run one model of each 2 types only ONCE on the final testing set.

6 Submitting Your Results

Please put all of your data and source code in a directory called team# where # is the number of your team and generate a compressed tarfile for submission. For example, if you belong to team number three, you should put all of your files in a folder called team3. Then, from the parent directory type tar zcvf team3 > team3.tar.gz. Then send archive to me.

6.1 Summary

If you do everything correctly, we should be able to type the commands something along the lines of

tar zxvf team3.tar
cd team3
make
./process_data.sh -lm -base train.data
./process_data.sh -lm -base test.data
./train.sh -lm -base train.data
./test.sh -lm -base test.data > test_lm_base.out
perl evaluate_vp.pl test_lm_base.out test.base.answers

to get the results for the base verb prediction task for the first part of the assignment (and with little modification to the second part of the assignment and other experiments).