CS 187 - Dependency Parsing
Phase 3: Final Implementation

Lauren Urke, Nathaniel Herman, Henrik Sigstad, Ariel Camperi

How to Run

To run the project, it is sufficient to run the python file `master.py`. There are several additional flags that can be added. To train the model, call `python master.py 1`, and this will train a model and save it to a `pickle` file. Then call `python master.py` and the program will test that model on the test data. We have also included some pre-existing models that we've trained with different SVM types. The file `linear.pkl` contains a model using instances of `LinearSVC()` (using one-vs-many classification), while the file `one_vs_one_linear.pkl` contains a model using instances of `OneVsOneClassifier(LinearSVC())`. To test with either of these pre-trained models, simply rename the model to `models.pkl` and the run the program with no arguments. When training, the default will be to use (2,2) as the context for the target nodes, however custom left and right contexts can be specified by running `python master.py 1 l r` (where `l` and `r` are the left and right context lengths). In addition, `pickle` files named `models_l_r.pkl` are also provided, containing a one-vs-one model trained with the context pair (l, r) specified in the file name. When testing a model that was trained a context pair (l, r) different from the default (2,2), make sure to run `python master.py 0 l r` so that the predicting code feeds in feature sets that match those used to train the model (note that any model to be tested must be named `models.pkl` for the script to recognize it).

State of the Implementation

The implementation works fully, from training data through accuracy evaluation on testing data. We tried several support vector machines from the `scikit-learn` package, in an attempt to match the accuracy metrics found in the paper. Using a one-vs-one classifier, we find the following values (with context (2,2)):

Dependency Accuracy	Root Accuracy	Complete Rate
0.89135	0.93675	0.32711

This result corresponds to column 1 of Table 2 in the paper. It is slightly lower. Using a one-vs-many classifier, we find the following values (again with context (2,2)):

Dependency Accuracy	Root Accuracy	Complete Rate
0.88680	0.93894	0.32285

Having determined that several other SVM options offered by the `scikit-learn` package did not work with this method, we settled on training and testing with the one-vs-one classifier using `OneVsOneClassifier(LinearSVC())`. We then tried different context lengths, and obtained the following results (the (l,r) number pairs correspond to context lengths on either side of the target nodes:

	(2,2)	(2,3)	(2,4)	(2,5)	(3,2)	(3,3)	(3,4)	(3,5)
Dep. Acc.	0.891	0.890	0.890	0.889	0.887	0.887	0.887	0.887
Root Acc.	0.937	0.928	0.934	0.930	0.932	0.929	0.928	0.927
Comp. Rate	0.327	0.330	0.325	0.326	0.312	0.318	0.316	0.317

Our results have somewhat worse accuracy than that in the paper–we suspect a few reasons for this: for one, polynomial SVMs like those used in the paper weren't working at all for us, so we only use linear SVMs, which may have worse accuracy (we tried the Crammer-Singer method, too, but got roughly the same results). Also, the paper does not specify how they handle sentences that couldn't be fully reduced (which happens fairly often). Currently we are not counting them for root accuracy and completion rate.

Statement of Work

Our initial attempts at replication were implemented to various degrees of completion, so for the final implementation we worked off of Nate's and Henrik's versions, which had reasonable accuracy rates when tested on toy data. Nate's implementation was faster, however Henrik's implementation had all accuracy measurements implemented, so they worked together to port this functionality to Nate's code, which used a slightly different encoding for the parse trees. Lauren worked on splitting up the testing and training functionality for reusability, specifically using `pickle` to save trained models. Ariel worked on implementing section 4.2 of the paper, namely using multiple SVMs, based on the POS tag of the left target node in a given target pair (this allows for a greatly reduced running time). We then all worked on improving the accuracy by testing various SVM types. We also all contributed to training and testing different models and compiling them in the writeup.