# Language & Statistics Final Project

Group Members:
Wes Feely, Mario Piergallini, Tom van Drunen, Callie Vaughn

After our initial meeting, we decided to pre-process the text data using TurboParser, a freely-available software package that includes a English part-of-speech tagger and dependency parser. We ran the TurboParser tagger and parser on the training and development data, and used this data for our design of linguistic features. One issue we encountered was that the TurboParser English tokenizer split apart the unknown word tokens ("<UNK>") into three separate tokens ("<","UNK",">"), which we had to handle using a sed script to put the unknown word tokens back together.

The features we initially brainstormed were document length (in number of words), number of unknown dependency relations in each document's parse trees (which receive a special label "DEP"), the number of "bad" consecutive tags (two consecutive determiners, coordinating conjunctions, or possessive-'s), and the number of repeated words in the document. Each of these last three features were normalized by the document's length.

We also had the idea to use topic features from an LDA topic model trained on the 100 million word corpus. We used 30 line segments from the 100MW corpus and treated them as documents, since most news segments were significantly longer. Since generated documents tend to be incoherent, a topic model should not be as certain what the topic is for a generated document. We considered using the maximum topic probability, the standard deviation of topic probabilities and the median topic probability as features to measure topical incoherence. A real document should have a high maximum score, a higher standard deviation and a lower median, all features which indicate a less uniform distribution. These features did, in fact, work well at differentiating real and fake documents in combination with document length. An SMO SVM algorithm was able to get 99.8% accuracy on the train set (10-fold cross validation). But the development set documents were shorter and significantly hurt its performance on those.

| **Feature set** | **Accuracy** |
|---|---|
| POS + Repeated words | 56.5% |
| LDA Topic features + Doc Length | 75.5% |
| Combination | 85.0% |

Table 1: Features and their accuracy

After training and running logistic regression in Weka, our results on subsets of features can be seen above. From these data, we can see that the LDA topic model features were quite effective in determining whether a document was real or fake, reaching 75.5% accuracy when used alone to train our classifier. The POS and repetition features on their own were not especially effective, only providing 56.5% accuracy when used alone to train our classifier. This was due to a few of the linguistic features being almost always zero, because consecutive tags POS-POS never occurred, DT-DT and CC-CC occurred quite infrequently, and "DEP" unknown dependency labels occurred somewhat evenly across the real and fake documents. The DEP feature did provide some value, but the repeated words turned out to be a valuable feature that improved our total performance. It seems likely that it was also picking up on topical coherence. In the end, the combination of our linguistic features and topic model features reached 85% accuracy.

If we were to continue this project, we would do some experiments using Weka to determine which linguistic features were the most useful, to choose the optimal feature set. We would also create new linguistic features, including the probability of each document, using a four-gram language model, which we predict might have accounted for locally correct trigrams in the fake data, which were not correct when extended to four-grams including the neighboring words.

Each group member's contributions to this project were the following:

- **Wes**: Ran TurboParser, wrote bash scripts, wrote Python code to extract linguistic features
- **Mario**: LDA topic model training, Weka algorithm and feature space experimentation
- **Tom**: Wrote Java code to run Weka
- **Callie**: Wrote bash scripts, wrote Python code to extract linguistic features, integrated Tom's Java code into pipeline