CMPT 882 Project Proposal

Andrei Vacariu

Problem

Dependency parsing is important to the Lensing Wikipedia project to understand what's happening in the sentences and pages it's parsing. As the goal for Lensing Wikipedia is to parse all of Wikipedia, it requires a fast and accurate parser which can handle the 53GB dump of all pages in a reasonable time frame.

Graph based models for dependency parsing seem to be one of the most successful approaches to the task of creating dependency trees for the words of a sentence. Using the Eisner algorithm, graphs are constructed and scored by summing over the scores of the possible subgraphs. This is done by computing an equation of the form

$$Score(x, \hat{y}(x); \theta) = \sum_{c \in \hat{y}(x)} ScoreF(x, c; \theta)$$

where x is the input sentence, $\hat{y}(x)$ is the candidate tree, and c are all the subtrees of $\hat{y}(x)$. Traditional scoring functions are of the form $ScoreF(x, c; \theta) = \mathbf{w} \cdot \mathbf{f}(x, c)$ where $\mathbf{w}$ is a weight vector and $\mathbf{f}$ produces all features. In this approach, the feature extraction takes up most of the time, and feature design is difficult and require extensive domain expertise.

Approach

To avoid the issues with feature generation and designing combinations of features, (Pei, et al. 2015) propose a neural network based scoring function. In their approach, they obtain embeddings of unigram features for the subgraph c, and concatenate them into a single vector a. Then a is fed into a hidden layer, and the output is then passed to the output layer, where the scoring function computes a score:

$$h = g(W_h^d a + b_h^d)$$
$$ScoreF(x, c) = W_o^d h + b_o^d$$

In the hidden layer, there is a novel activation function, the *tanh-cube*, defined as $g(x) = \tanh(x^3 + x)$. The cubed term models feature combinations as some terms will have 3 elements of a multiplied together and some will have 2, each with its own weights.

As only unigram features are used as input to the neural network when parsing, there is potential to improve on the performance of (Pei, et al. 2015) by pre-computing the feature embeddings and their hidden-layer values and hard coding them into the code given to the GPU before the Eisner algorithm begins filling in its table.

Dataset

To properly compare performance and accuracy results to the (Pei, et al. 2015) paper, the same dataset will be used: the English Penn Treebank to evaluate the model and Yamada and Matsumoto (2003) head rules used to extract dependency trees. The Stanford POS Tagger will be used to assign POS tags.