Homework #3: Statistical Machine Translation

Anoop Sarkar – `anoop@cs.sfu.ca`

Each group should do one question from the following.

(1) **Tuning as Ranking**

(DT-group) Implement the method described in the following paper for tuning a log-linear model for statistical machine translation. It is sufficient to implement it for the three feature functions used with the simple phrase table and decoder from HW2. Optionally create a plug-in replacement for the moses-mert module in moses.

> Mark Hopkins and Jonathan May. Tuning as Ranking. EMNLP 2010.
> `http://www.aclweb.org/anthology/D11-1125` (slides: `http://goo.gl/R4Y8Y`)

(2) **Earley Parsing**

(LRHiero-group) Implement the Earley parsing algorithm for monolingual sentence parsing. The algorithm is explained in the following slides: `http://goo.gl/LwIWq`.
Use the following grammar:

```
S -> NP VP
VP -> V NP | V NP PP
V -> "saw" | "ate"
NP -> "John" | "Mary" | "Bob" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "dog" | "cat" | "cookie" | "park"
PP -> P NP
P -> "in" | "on" | "by" | "with"
```

Produce a trace of execution for the input sentence using your implementation of the Earley algorithm:

> the cat in the park ate my cookie

The file `earley_setup.py` contains some helpful tips on how to build the data structures required to implement this algorithm. Your program should produce a trace as given in the file `earley-trace.txt`.
Once you are done with the Earley implementation, you can optionally do the following as well. Modify the rule extraction step in Kriya (our local re-implementation of Hiero) to produce Hiero-style synchronous context-free (SCFG) rules that correspond to the Greibach Normal Form SCFG rules as defined in the following paper:

> Taro Watanabe, Hajime Tsukada and Hideki Isozaki. Left-to-Right Target Generation for Hierarchical Phrase-based Translation. COLING-ACL 2006.
> `http://aclweb.org/anthology/P/P06/P06-1098.pdf`

Compare the difference in performance between the normal Hiero rules versus the Greibach normal form rules on Arabic-English SMT using the Kriya implementation of Hiero.

(3) **Latent SVMs for Machine Translation**

(alignment-group) Implement the algorithm for discriminative language modeling described in the following paper. You can use the context-free grammar(s) you have developed for HW2, and use the parser provided to you.

> Colin Cherry and Chris Quirk. Discriminative, Syntactic Language Modeling through Latent SVMs. AMTA 2008.
> `http://research.microsoft.com/pubs/72874/lsvm_amta.pdf`

A component of the implementation: sampling from a language model, is available at:
`http://www.cs.sfu.ca/~anoop/distrib/trigen/gen-from-lm.py`

(4) **Syntactic Sentence Compression**

(syntactic-group) The file `/cs/natlang-data/ZiffDavis-compress/SentencePairsShorter` contains pairs of sentences: the first sentence is a grammatical compressed or shortened version of the second sentence in the pair. Parse each of the sentences using a state of the art statistical parser (Berkeley parser, Charniak-Johnson parser, Bikel parser, MSTParser, or MALTParser) and either learn or write heuristic rules that edit the context-free rules from the longer sentence to produce the shorter sentence. Test your performance on the training data during development and then test your final version on the file `test.original`. Compare your performance against the other `test.*` files.