As before, for this project you may work in a team of either two or three students. The joint report for your team must be submitted in hard copy at the start of the lecture on Tuesday February 12, 2013.

The objective of this project is to understand the basic algorithms for conditional random fields (CRFs) thoroughly. The experimental task is to learn a CRF model that can divide novel English words into syllables correctly. The dataset, which consists of 66,001 English words with syllables separated by hyphens, is available at http://www.cs.ucsd.edu/users/elkan/hyphenation/.

Design and implement a process for generating feature functions automatically from high-level specifications such as “all consecutive sequences of four specific letters.” In your report, explain this process. The implementation of the process does not have to be general-purpose, that is the code does not need to be usable for other applications. However, the process itself should be systematic, meaning that a relatively small program should enumerate all feature functions in several large classes. Make sure that each feature function has value zero for most input-output pairs, and that this common case is handled efficiently both during data generation and during CRF training.

The CRF implementation should handle any small set of output tags. For each input word, there is always one output tag per letter. Design two different sensible schemes for using tags to encode syllables. For example, one scheme is called BIO: the tag B for “begin” indicates that a letter is the first one in its syllable, the tag O for “out” indicates that a letter is the last one in its syllable, and the tag I for “in” indicates that a letter is in the middle of its syllable. For each scheme, make reasonable decisions about what tags to use for short syllables, and for the notional positions immediately before and after the first and last letters of a word. Compare experimentally the accuracy that is achievable with the two different schemes that you propose.

Implement and do experiments with two different training methods. The first method should be Collins’ perceptron algorithm, and the second should be one of the following:

1. stochastic gradient following.
2. a general-purpose numerical optimization package such as an implementa-
tion of L-BFGS, or

3. contrastive divergence.

For each training method, implement all needed CRF-specific algorithms yourself. Note that perceptron training requires only the Viterbi algorithm, while contrastive divergence needs you to understand and implement Gibbs sampling. L-BFGS and stochastic gradient following require computing the gradient.

It is vital to be confident that basic algorithms are implemented correctly. Your report must convince the reader that this is true. For methods that use gradients, use a function to verify that computed derivatives are correct. After you are sure that your algorithm implementations are correct, it is also vital to make them fast. Use a small random subset of words to maximize the speed of your code. One of the big advantages of Matlab is that it provides an excellent profiler that can show you easily which parts of your code are slow.

After you have a correct, fast, and robust implementation of each learning algorithm, tune its settings using cross-validation. Use letter-level accuracy as the performance metric. This is simply the fraction of letters that a method gets exactly right.