Assignment #5: Dependency parsing

## **Objectives**

The objectives of this assignment are to:

* Know what a dependency graph is
* Understand the principles of Nivre's parsing mechanisms
* Extend Nivre's parser with a guiding predicate that parses an annotated dependency graph
* Extract features to learn parsing actions from an annotated corpus
* In this assignment, you will only generate the Weka models from the extracted features. You will complete the parser and apply it in the next assignment.

## **Organization and location**

The fifth lab session will take place on

* Group 1: Wednesday, October 8 from 10:15 to 12:00 in the Alpha room
* Group 2: Wednesday, October 8 from 10:15 to 12:00 in the Gamma room
* Group 3: Wednesday, October 8 from 13:15 to 15:00 in the Gamma room

You can work alone or collaborate with another student.

Each group will have to:

* Write a program that parses a sentence when the dependency graph is known
* Extract features from the parsing actions.

## **Programming**

This assignment is inspired by the shared task of the Tenth conference on computational natural language learning, [CONLL-X](http://ilk.uvt.nl/conll/), and uses a subset of their data. The conference site contains a description of multilingual dependency parsing, reference papers, training and test sets for a variety of languages, as well as evaluation programs. See also [CONLL 2007](http://depparse.uvt.nl/SharedTaskWebsite.html), on the same topic.

In this session, you will implement a dependency parser for Swedish. Should you want to use another corpus, please tell me in advance.

### **Choosing a training and a test sets**

The CONLL-X annotated corpora and annotation scheme are available [here](http://ilk.uvt.nl/conll/post_task_data.html). The Swedish corpus called *Talbanken* was originally collected and annotated in Lund and modified by Joakim Nivre. You can read details on the corpus and references[here](http://stp.ling.uu.se/~nivre/swedish_treebank/).

1. In this assignment, you will use the CONLL-X Swedish corpus. Download the tar archives containing the training and test sets for Swedish and uncompress them: [[data sets](http://ilk.uvt.nl/conll/free_data.html)]. Local copies: [[training set](http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/swedish_talbanken05_train.conll)] [[test set](http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/swedish_talbanken05_test_blind.conll)] [[test set with answers](http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/swedish_talbanken05_test.conll)].

### **Nivre's parser**

For each sentence with a projective dependency graph, there is an action sequence that enables Nivre's parser to generate this graph. Gold standard parsing corresponds to the sequence of parsing actions, left-arc (la), right-arc (ra), shift (sh), and reduce (re) that produces the manually-obtained, gold standard, graph.

Action sequences can be trained from an annotated corpus, or more precisely the next action can be trained from the current parsing context. To be able to predict the next action, gold standard parsing must also extract feature vectors at each step of the parsing procedure. The simplest parsing context corresponds to words' part of speech on the top of the stack and head of the input list (the queue).

Once the data collected, the training procedure will produce a 4-class classifier that you will embed in Nivre's parser to choose the next action. During parsing, Nivre's parser will call the classifier to choose the next action in the set {la, ra, sh, re} using the current context.

1. Discuss how to extend Nivre's parser to carry out a gold standard parsing. Given a manually-annotated dependency graph, what are the conditions on the stack and the current input list -- the queue -- to execute left-arc, right-arc, shift, or reduce? Start with left-arc and right-arc, which are the simplest ones.
2. As main features, you will use three models:
   * The top of the stack and the first word of the input list
   * The two first words on the top of the stack and the two first words of the input list
   * A feature vector that you will design that will extend the previous one with at least one feature. You can read [this paper](http://www.aclweb.org/anthology/C/C10/C10-1093.pdf)(Table 6) to build your vector. Use only POS-based features as your machine--learning algorithm would not scale well with features with many values such as lexical (word) features.
3. Nivre's parser sets constraints to actions. Name a way to encode these constraints as features. Think of Boolean features.
4. Read this program [[1](http://fileadmin.cs.lth.se/cs/Education/EDAN20/programs/parser/src.zip)], notably ReferenceParser.java. Edit the data paths in Constants.java so that they fit your configurations. You will run the ReferenceParser program. For now, it never terminates.

### **Parsing functions**

Using the actions in the set {la, ra, sh, re} will produce an unlabelled graph. It is easy to extend the parser so that it can label the graph with grammatical functions. In this case, we must complement the actions la and ra with their function using this notation for example: la.++, la.+A, la.+F, la.AA, la.AG, etc. where the prefix is the action and the suffix is the function.

Read the complete list of actions extracted from the Swedish corpus in CoNLL-X [here](http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/domain_functions.arff).

### **Extracting features (I)**

The final goal is to parse the Swedish corpus in CoNLL-X and produce a labelled and unlabelled dependency graph. You will show the parsing results at the end of the 5th assignment. In this assignment, you will only generate the Weka models.

You will consider three feature sets and you will learn the corresponding decision tree classifiers using Weka:

1. The first set will use two parameters extracted from the stack and the queue,
2. the second one, four parameters.
3. For the third model, you will extract at least two more features, one of them being the part of speech of the word following the top of the stack in the sentence order.

These sets will include two additional Boolean parameters, "can do left arc" and "can do reduce", which will model constraints on the parser's actions. In total, the feature sets will then have four (respectively six and eight) parameters.

This means that the purpose of this assignment is to generate six Weka models:

1. Three models for the unlabelled graphs
2. Three models for the labelled graphs

To carry this out:

1. Modify the Java program and build the data sets (six in total). You will have to complete:
   * the extractFeatures() method in ReferenceParser.java so that you extract the features.
   * the parse() method in ReferenceParser.java so that you determine the action, for instance la or la.SS. You will write a conditional that calls the oracles and add the action to the transition list using transitionList.add(). Don't forget to carry out the action in each member of the conditional.
2. Download [this header](http://fileadmin.cs.lth.se/cs/Education/EDAN20/corpus/conllx/sv/parsing_header4_java.arff) and create headers to fit the 2-parameter set, your parameter set, as well as the labelled graphs.
3. Generate the six Weka models. (You must use Weka version 3.7.2 or higher.). You will evaluate the model accuracies (not the parsing accuracy) using the summary of the stratified cross-validation produced by Weka and the correctly classified instances

The first lines of your ARFF file for the 4-parameter and labelled version should look like the except below:

nil nil ROOT NN false false sh

ROOT nil NN ++ true false sh

NN ROOT ++ NN true false sh

++ NN NN AV true false la.++

NN ROOT NN AV true false ra.CC

NN NN AV EN false true re

NN ROOT AV EN true false la.SS

ROOT nil AV EN true false ra.ROOT

AV ROOT EN AJ false true sh

## **Complement (Optional)**

Read the text *An Efficient Algorithm for Projective Dependency Parsing* by Joakim Nivre (2003) [[pdf](http://stp.lingfil.uu.se/~nivre/docs/iwpt03.pdf)]. You can find additional references [here](http://stp.lingfil.uu.se/~nivre/docs/cv_eng.html).