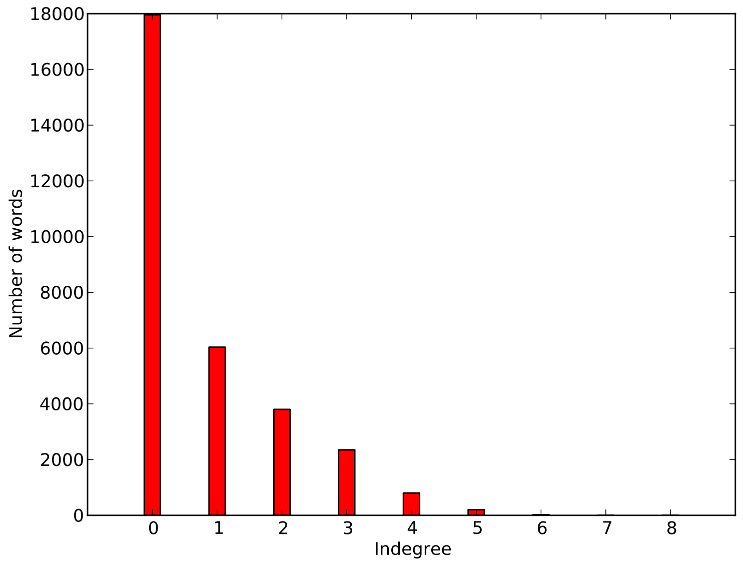
Dependency Parsing, assignment 1

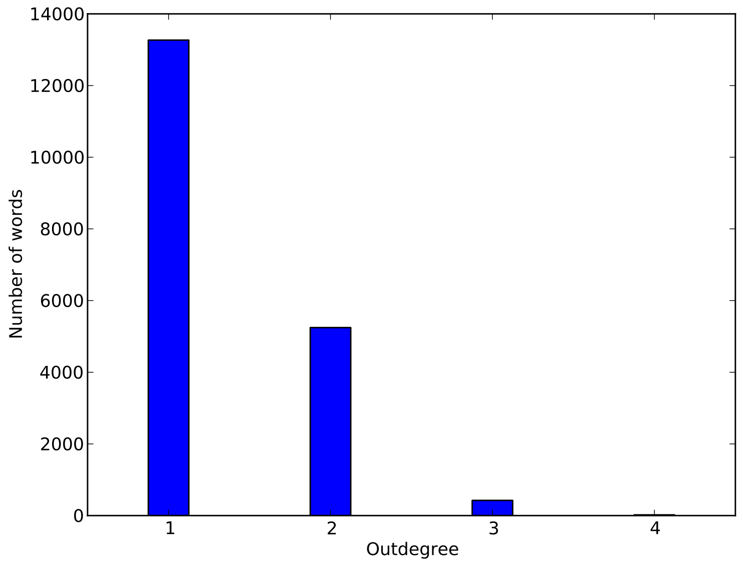
Melanie Tosik

December 17, 2014

1 DM: MRS-Derived Semantic Dependencies

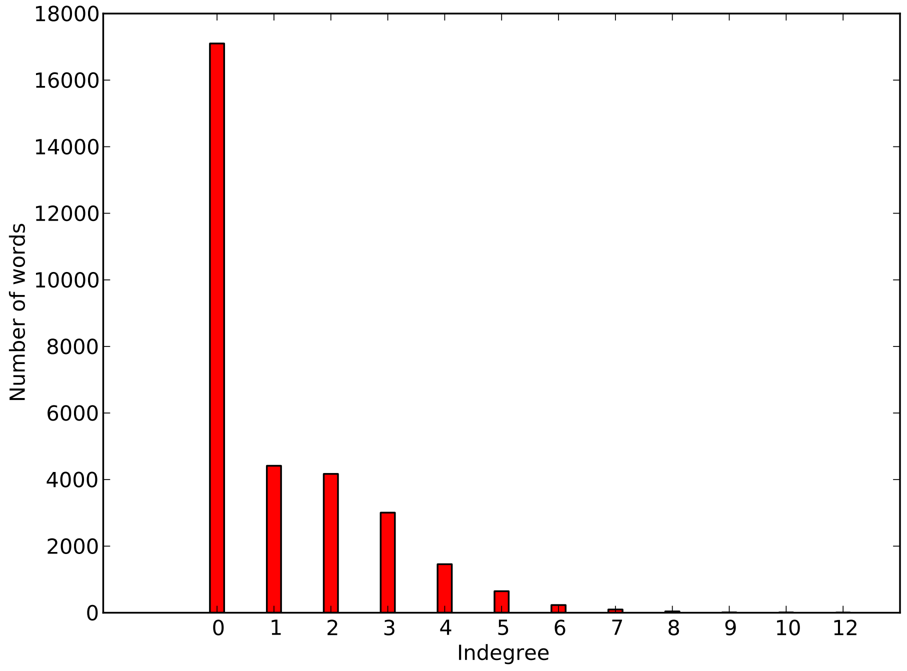
Number of graphs:	1614
Number of words:	31184
Number of different edge labels:	35
Average number of predicates per sentence:	12.1
Average number of singletons per sentence:	4.9

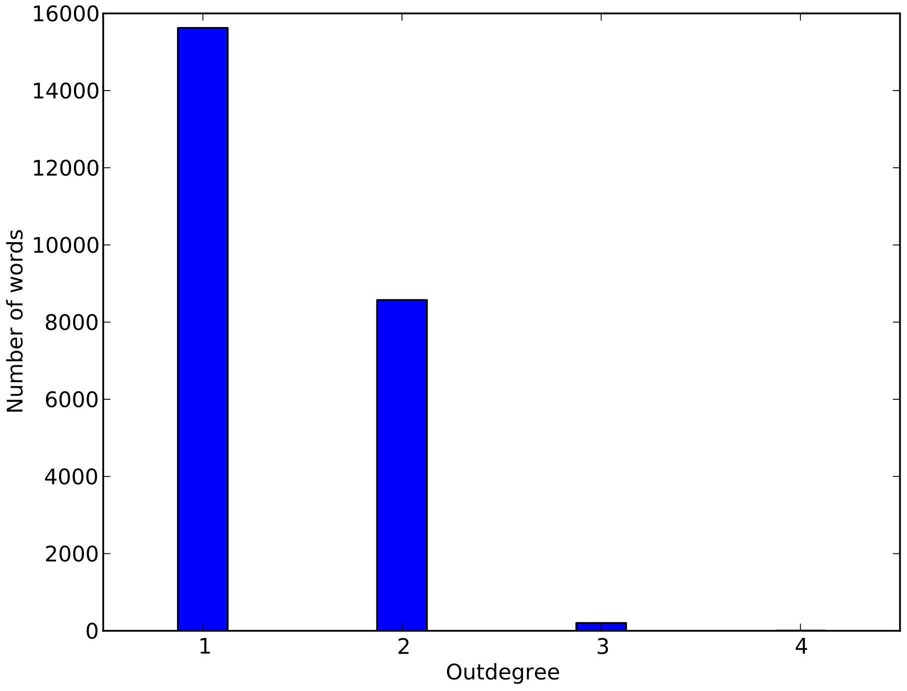




2 PAS: Enju Predicate–Argument Structures

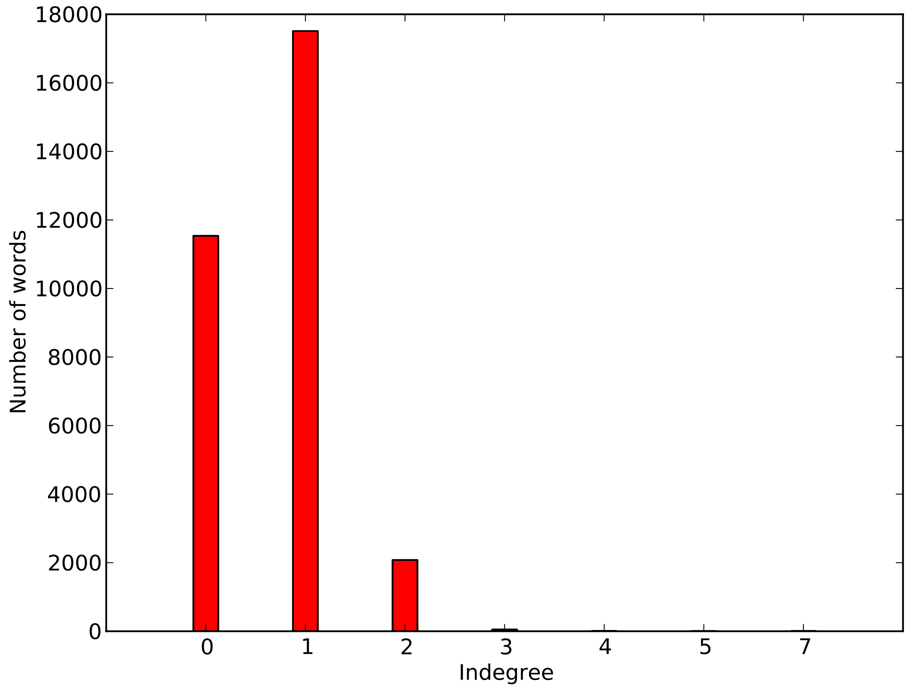
Number of graphs:	1614
Number of words:	31184
Number of different edge labels:	39
Average number of predicates per sentence:	15.3
Average number of singletons per sentence:	1.0

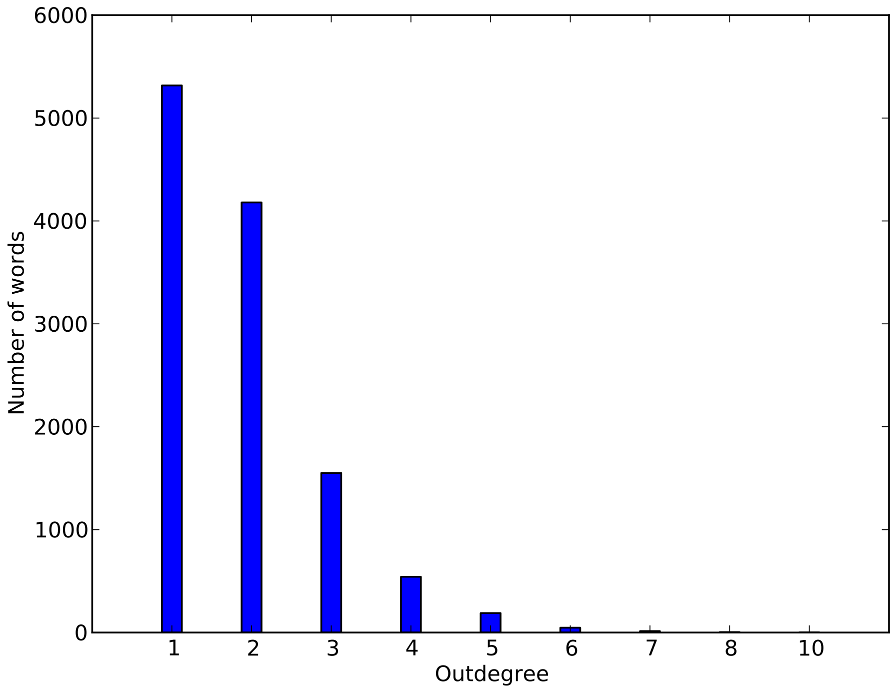




3 PCEDT: Parts of the Tectogrammatical Layer

Number of graphs:	1614
Number of words:	31184
Number of different edge labels:	64
Average number of predicates per sentence:	7.4
Average number of singletons per sentence:	7.8





```
#!/usr/bin/python
#
#   File name:    dp1.py
#   Author:       Melanie Tosik
#   Platform:     OS X (10.9.5)
#   Description: Statistics on semantic dependency graphs

from __future__ import division
import sys
import string
from operator import itemgetter
import matplotlib.pyplot as plt
import numpy as np


class Sim(object):
    """ Provides some basic statistics on semantic dependency graphs """

    def __init__(self, simfile):
        # Dictionary {indegree : number of words with indegree}
        self.indegree = {}
        # Dictionary {outdegree : number of words with outdegree}
        self.outdegree = {}
        self.numbers(simfile)
        #self.plots()

    def numbers(self, simfile):

        # f = open(simfile, 'a')
        # f.write('\n\n')
        # f.close()

        # Contains current sentence
        sentence = []
        # Set of different edge labels
        label_set = set()
        # Count number of sentences
        number_of_sentences = 0
        # Count number of words
        number_of_words = 0
        # Count number of predicates
        number_of_predicates = 0
        # Count number of singletons
        number_of_singletons = 0
        # Dictionary {Column index : Number of pred_arg roles}
        columns = {}

        with open(simfile, 'r+') as f:
            for line in f:
                # Gets single sentences
                if line.strip(): #< DEBUG: file ending with '\n\n'
                    sentence.append(line.split('\t'))
                else:
                    number_of_sentences += 1
                    # Processes current sentence
                    for field_line in sentence:
                        # Ignores sentence prefixes
                        if len(field_line) > 1:

                            # Fields 1-4: id, form, lemma, pos
                            if field_line[1] not in string.punctuation:
```

```
                    number_of_words += 1

                    # Indegree = number of pred-arg roles per
                       row
                    in_cnt = 0
                    for field in field_line[6:]:
                        if field.strip() not in string.
                           punctuation:
                            in_cnt += 1

                    if in_cnt in self.indegree.keys():
                        self.indegree[in_cnt] = self.indegree[
                           in_cnt] + 1
                    else:
                        self.indegree[in_cnt] = 1

                    # Sums number of pred-arg roles per column
                    column_index = 0
                    for field in field_line[6:]:
                        column_index += 1
                        if field.strip() not in string.
                           punctuation:
                            if column_index in columns.keys():
                                columns[column_index] += 1
                            else:
                                columns[column_index] = 1

                # Fields 5,6: top, pred
                if field_line[5] == '+':
                    number_of_predicates += 1

                # Additional fields starting from 7: pred-arg
                   roles
                for field in field_line[6:]:
                    if field.strip() not in string.punctuation:
                        label_set.add(field.strip())

                # Singletons
                # Neither top nor pred and no pred-arg roles
                if field_line[4:6] == ['-','-'] and all(field.
                   strip() == '_' for field in field_line[6:]):
                    number_of_singletons += 1

        # Outdegree = number of pred_arg roles per column
        for out_cnt in columns.itervalues():
            if out_cnt in self.outdegree.keys():
                self.outdegree[out_cnt] += 1
            else:
                self.outdegree[out_cnt] = 1

        sentence = []
        columns = {}

print 'Number of graphs:', number_of_sentences
print 'Number of words:', number_of_words
print 'Number of different edge labels:', len(label_set)
print 'Average number of predicates per sentence:', round((
   number_of_predicates/number_of_sentences),1)
print 'Average number of singletons per sentence:', round((
   number_of_singletons/number_of_sentences),1)
```

```
            print '{Indegree : number of words}: ', self.indegree
            print '{Outdegree : number of words}: ', self.outdegree

    def plots(self):

        # Plot indegree
        # x_axis: Number of words
        # y_axis: Indegree

        histogram_indegree = sorted(self.indegree.items(), key=itemgetter(0)
            , reverse=False)
        hist_dict_in = dict(histogram_indegree)

        plt.bar(range(len(hist_dict_in)), hist_dict_in.values(), align='
            center', width=0.25, color='r')
        plt.xticks(range(len(hist_dict_in)), hist_dict_in.keys())

        plt.xlabel('Indegree')
        plt.ylabel('Number of words')

        plt.show()


        # Plot outdegree
        # x_axis: Number of words
        # y_axis: Outdegree

        histogram_outdegree = sorted(self.outdegree.items(), key=itemgetter
            (0), reverse=False)
        hist_dict_out = dict(histogram_outdegree)

        plt.bar(range(len(hist_dict_out)), hist_dict_out.values(), align='
            center', width=0.25, color='b')
        plt.xticks(range(len(hist_dict_out)), hist_dict_out.keys())

        plt.xlabel('Outdegree')
        plt.ylabel('Number of words')

        plt.show()


if __name__ == '__main__':
    if len(sys.argv) == 2:
        sim = Sim(sys.argv[1])
    else:
        print 'Usage: python dp1.py <data file>'
```