INF5830, Fall 2017
Assignment 4: Semantic Role Labeling
Project B

Andrey Kutuzov

November 14, 2017

Deadline 28 Nov., at 18:00, to be delivered in Devilry

Goals

1. Get familiar with the *CoNLL* format for storing information about semantic roles.
2. Learn to design and train data-driven semantic role labeling models using *scikit-learn* or other machine learning framework.
3. Learn to evaluate different machine learning algorithms and feature sets for semantic role labeling.
4. Learn to inject linguistic constraints into machine learning models.

Introduction

Semantic role labeling (SRL) is the task of identification and classification of predicates and arguments in the sentence. This is the next step after syntactic parsing towards the complete understanding of the meaning of utterance. As such, this task is foundational for many practical NLP applications. In this assignment, you will design and implement a simple semantic role labeling system yourself, using supervised machine learning approach. As a machine learning framework, we recommend *scikit-learn*, but you are free to draw on any other libraries, depending on your preferences: *Weka*, *Keras*, *TensorFlow*, *DyNet*, you name it. The only requirement is that you should implement your own SRL system, not simply take an off-the-shelf tool. Throughout the assignment, the *CoNLL-2008* datasets will be used as the source of training and testing data.

Please make sure you read through the entire assignment before you start. Solutions must be submitted through Devilry[1] by 18:00 on November 28. Please upload a single PDF file with your answers. Your (heavily commented) code and data files should be available in a separate private repository in the UiO in-house Git platform[2]. Make the repository private, but allow full access to

[1] `https://devilry.ifi.uio.no/`
[2] `https://github.uio.no/`

me (https://github.uio.no/andreku). The PDF in Devilry should contain a link to your repository. If you have any questions, please raise an issue in the INF5830 Git repository[3] or email *andreku@ifi.uio.no*. Make sure to take advantage of the group session on November 20.

Recommended reading

1. **Speech and Language Processing**. Daniel Jurafsky and James Martin. 3rd edition draft of June 26, 2015. Chapter 22, 'Semantic Role Labeling'[4]
2. **The proposition bank: An annotated corpus of semantic roles.** Palmer, M, et al., 2005.[5]
3. **Semantic Role Labeling: An Introduction to the Special Issue.** Màrquez, L., et al., 2008.[6]
4. **The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies.** Surdeanu, M., et al., 2008.[7]
5. **The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages.** Hajič, J. et al., 2009.[8]
6. http://scikit-learn.org/
7. http://surdeanu.info/conll08/
8. http://ufal.mff.cuni.cz/conll2009-st/

1 Spreading the word

It seems that the Norwegian Wikipedia lacks an article describing data-driven semantic role labeling. There is a very brief article about semantic roles in general (https://no.wikipedia.org/wiki/Semantisk_rolle), but it doesn't say a word about computational methods and NLP. Your task is to fix this.

This is a collective assignment: all Norwegian-speaking students in the group should work on it. A stub page was created for you (https://no.wikipedia.org/wiki/Semantisk_rolle_oppmerking), containing tentative structure of the article. You should fill it in with a brief gist of what you have learned about semantic role labeling from this course lectures and reading list. There are no restrictions on minimum or maximum size of this article, but it should give an arbitrary reader an idea about what semantic role labeling is.

Each of you can choose any section or subsection of the stub page and work on it. Once you choose your part, make sure to tell the others about it in the Git issue https://github.uio.no/andreku/INF5830/issues/1. Then, you should log in to Wikipedia under your username and edit the page, adding

[3] https://github.uio.no/andreku/INF5830/issues
[4] https://web.stanford.edu/~jurafsky/slp3/22.pdf
[5] http://www.mitpressjournals.org/doi/pdf/10.1162/0891201053630264
[6] http://www.mitpressjournals.org/doi/pdf/10.1162/coli.2008.34.2.145
[7] https://aclanthology.info/pdf/W/W08/W08-2121.pdf
[8] https://aclanthology.info/pdf/W/W09/W09-1201.pdf

your part. Feel free to change the section titles if need be. In the report file, you should mention your Wikipedia username (different for each of you). Link your article to the English Wikipedia article about semantic role labeling (`https://en.wikipedia.org/wiki/Semantic_role_labeling`).

In case you feel that your knowledge of Norwegian is not good enough for this task, you can create (or enrich if it already exists) an article about data-driven semantic role labeling in your native language Wikipedia. The size of one paragraph will be enough. Please do not update the English Wikipedia, it already has much data on that topic.

Please take this task seriously and write responsibly: your text will be read and used by many people. Don't describe things about which you are unsure. Support your text with references, as per Wikipedia guidelines.

2 Implementing semantic role labeling system

In this part of the assignment you will be working to solve the task of **argument classification**, an integral subtask within the larger task of SRL. We will assume that predicates and arguments have been identified and will focus on the task of labeling these arguments with semantic roles. We will be working with the original data set from the *CoNLL08* shared task on syntactic and semantic parsing for English. The assignment will be solved as a supervised classification task. In doing so, you will need to process the data to extract relevant features, format these appropriately and experiment with different machine learning algorithms, in order to arrive at your final solution.

The assignment is divided into five parts:

1. **Data processing**;
2. Training a **baseline system**;
3. **Feature engineering**;
4. Implementing **re-ranking**.
5. **Evaluating the final system**.

You have to submit a written report of 3-6 pages to Devilry, which provides details on your experiments and addresses the questions posed in the assignment. Your code should be available in your Git repository (we remind that the official programming language of the INF5830 course is *Python 3*). The code must be self-sufficient, meaning that it should be possible to run it directly on the training and test data. If you use some additional data sources, these datasets should be put in the repository as well. If you use some non-standard *Python* modules, this should be documented in your PDF report. The repository should also contain the predictions of your system for the test data (more on this below).

2.1 Data processing

The data sets from the *CoNLL08* shared task are available through the Linguistic Data Consortium, of which the University of Oslo is a member. You can obtain the training and test datasets here: `https://github.uio.no/andreku/INF5830/blob/master/project_B/data/project_b_data.tar`.

You are now free to use the data as long as you do **not** distribute it to anyone outside the University of Oslo. The .tar archive contains the following files:

- `README` *(brief description of the data and the shared task)*
- `srl_test.conll.gz` *(compressed test data in the CoNLL format)*
- `srl_train.conll.gz` *(compressed train data in the CoNLL format)*

Make sure that you understand the format of the training data (`srl_train.conll.gz`). You can find information in the course lectures, in the paper from the shared task organizers or on the *CoNLL08* web page. Note that the CoNLL-format used for dependency parsing in the previous assignment has been extended for this task to include information on semantic roles (annotations are taken from *PropBank*, *NomBank* and other resources). Briefly describe these differences in your report. Comment at least on these two questions:

1. Why the number of columns in the data varies?
2. How hyphenated words are treated and how this affects the representation of predicate-argument structure?

The testing dataset (`srl_test.conll.gz`) is different from the training dataset in one respect: all the semantic role labels for arguments are replaced with the placeholder '`ARG`'. See the 'Evaluation' subsection for the information on what to do with it.

NB: the dataset dates back to 2008, which means that it does not use the *Universal Dependencies* PoS tags and dependency relation names. Also, some annotation conventions are different, don't be surprised.

2.2 Training a baseline system

We formulate our classification task as follows: given an argument, a predicate and dependencies structure of the sentence, **predict a semantic role** for the argument. In order to perform this task you will need to consider the following:

- what are the instances for classification?
- which features can be used to represent the instances?

You first step should be writing the code which takes a *CoNLL08* data file and:

- stores the data in a multidimensional datastructure that allows you to access the different fields (e.g. PoS, dependency relation, etc) by token index
  - *Pandas* dataframes[9] are strongly recommended, but it is not a requirement.
- locates the semantic predicate(s) in the sentence;
- extracts semantic arguments, all the while keeping track of the predicate for each argument.

[9] `http://pandas.pydata.org/`

Briefly describe this code in the report.

You now have what you need in order to extract features for your classifier. It is probably a good idea to store the full feature vector for each instance as a dictionary in a list of dictionaries (but you can come with other solutions):

```
[ {feature 1: value 1, feature 2: value 2, ..., feature n: value n},
  {feature 1: value 1, feature 2: value 2, ..., feature n: value n} ]
```

A separate list should contain the correct semantic role labels (classes) for your instances in the same order as in the feature vectors list: for example, `classes[i]` contains the class label for the i_{th} instance.

Your initial baseline system should extract and use the following basic features:

1. lemma and sense number of the predicate (for example, '*give.01*');
2. PoS-tag of the predicate (for example, '*VBD*');
3. PoS-tag of the argument (for example, '*NN*');
4. The grammatical function of the argument, as expressed in its dependency relation (for example, '*OBJ*')

Implement a system which trains on these baseline features and evaluate it (see the 'Evaluation' subsection below). You can use any machine learning classifier (SVM from *scikit-learn* is probably a good start, but try a couple of other algorithms as well). For a start, you can look at the code of the supervised word sense disambiguation system[10] that we discussed at the lab sessions. Describe the evaluation results in the PDF report.

2.3 Feature engineering

Extract additional features for your system. You should introduce at least 4 new features. In order to do so you may glance at the lecture slides or at the literature for inspiration. Write a short description of your additional features and how these were extracted. Provide examples of some feature values.

As part of your submission for this assignment you should process the training data and print the feature vectors of the first 10 instances along with their classes to a file entitled *features.txt*. This file should be uploaded to your Git repository.

2.4 Re-ranking

We are sure your systems are now up and running but so far they use only the **local scoring**. This means that your feature vectors probably do not contain any information about *other arguments* in the same sentence. This can lead to the situation when the final predicted labels contradict basic linguistic intuition (for example, we have **two** A0 arguments for the verb '*give*', thus stating that there are two AGENTs). To decrease the probability of such a scenario, you have to implement a re-ranking system ('**global scoring**').

[10] `https://github.uio.no/andreku/INF5830/blob/master/wsd_code/semcor_supervised.py`

Conceptually, this is a post-processing step with the aim to make sure that the final predicted semantic role structure of the sentence conforms to some linguistic constraints (for example, '*verbal predicates can't have more than one A0 arguments*', etc). In order for this to work, you should modify your main system so that for each instance it outputs not a single predicted semantic role, but a vector of probabilities of each possible semantic role for this instance. At the re-ranking stage, your system should select the most probable set of predictions which does not violate linguistic constraints. Note that the re-ranker takes the whole sentence (or at least all arguments of the current predicate) as an input, so you will have to devise some way of storing sentence-level information.

You can either invent the linguistic constraints yourself or consult the literature. Use at least two constraints. Report the evaluation results for all combinations of your constraints.

2.5 Evaluation

You should evaluate all variants of your system using 10-fold cross-validation[11] on the training data. For each variant, report:

1. accuracy;
2. precision, recall and macro F1 score;
3. the same metrics separately for nominal and verbal predicates;
4. five semantic roles which obtained the highest F1 scores;
5. five semantic roles which obtained the lowest F1 scores;

Describe the differences in your systems' performance in the PDF report. Give your opinion on the influence of different features or re-ranking. How far are your models below the state-of-the-art results for English SRL?

As a final step, choose the best final configuration of your system. Then, implement a function or a separate script which takes as an input the test dataset (`srl_test.conll.gz`) and uses your trained model to label the arguments in it (replaces '`ARG`' with predicted semantic roles, leaving everything else intact). The version of `srl_test.conll.gz` with the labels predicted by your system should be uploaded to your Git repository.

Gold standard data for these semantic roles is kept secret until everyone submits their results. After that, we will evaluate your submissions and publish your systems' rankings on the UiO Github.

Good luck with your submissions and happy coding!

[11] `http://scikit-learn.org/stable/modules/cross_validation.html`