nlp 2020 course

Assignment 4: Relation Extraction

In this assignment, you will implement a Relation Extraction (RE) system, making use of the output of the various NLP tools you learned about in the course. The assignment is somewhat open-ended, and you are not restricted as to the methods you are expected to use. You can write either a rule-based system, a machine-learning based system, or a hybrid system combining both approaches.

Relation Extraction (RE)

RE aims at extracting semantic relations between entities from a given text. We will focus on extracting binary relations, i.e. relations between exactly two entities, from sentences. We will denote the relations `rel(arg1,arg2)` where `arg1` and `arg2` are the two entities in a predefined relation `rel` within a sentence `S`.

Given a sentence, the task is to decide whether the sentence contains a mention of a relation of interest, and if yes -- extract the relation name and the two corresponding entities.

In this assignment we will be looking for the following relations:

- `Work for`, e.g. `Work_For(Yoav Goldberg, Bar Ilan University)`
- `Live in`, e.g. `Live_In(Ido Dagan, Israel)`

Note that the order of the entities within a relation is important, e.g. compare `Work_For(Yoav Goldberg, Bar Ilan University)` and `Work_For(Bar Ilan University, Yoav Goldberg)`. The first relation is correct while the second is not (although it could be nice if that was the case).

You need to identify one of these relations (WorkFor or LiveIn). It is your choice which of the relations to address.

Requirements

You need to implement:

1. a system that gets as input a text file with multiple sentences, and returns as output a list of extracted relations together with the sentences the relations were extracted from. (40%)
2. an evaluation program that gets your programs's output and the gold-standard data, and outputs the Precision, Recall and F-1 scores. (10%)
3. You will also submit a report. (50%)

Your main grade will be based on the report -- but note that the report will be based on what

you actually implemented, and the results you obtained.

Data

We provide you with two datasets, `TRAIN` and `DEV`. Each set consists of two files:

1. Corpus (Corpus.TRAIN.txt / Corpus.DEV.txt) -- These files contain sentences, some of which mention relations you'll need to extract. Each line is of the format `sentence_id<TAB>sentence_text`. The sentence ids correspond to sentences in the annotation files (see below).

2. Annotation (TRAIN.annotations / DEV.annotations) -- these files contain the gold-standard annotation of the relations, which should be extracted from the sentences of the corresponding `Corpus` files. Each line represents a single relation extracted from a sentence. The corresponding sentence is given is brackets for your convenience. In the file, columns are separated by TABs, while tokens are separated by whitespace.

3. Processed Corpus (Corpus.TRAIN.processed / Corpus.DEV.processed) -- these are the Corpus files, with lemmas, POS-tags, Named-entity-tags and dependency-tree annotations. The annotations are produced using the spacy NLP toolkit, using the following script: spc.py.

   You can use these files directly, or run `spacy` yourself. Running `spacy` has some benefits: it provides also NP chunks information (which can be inferred from the dependencies, but perhaps easier with the `spacy` API, see the commented-out example in the spc.py script), as well as easy access to word vectors, and a nice API for handling accessing tags and lemmas information, as well as dependency-tree navigation.

   The `spacy` documentation is available here. It also lists the POS tag set, Named Entities tag set, and so on.

   Spacy is a python package. If you use Java, you can either read the processed file and use it, or use Stanford's CoreNLP library to process the Corpus files.

You can download all the data in a single file: data.tar.gz

Additional Resources You may (but are not required to) also use additional resources such as the paraphrase database, word-vectors like you've used in assignment 3, and WordNet.

Input and Output formats

The input to your program could be either in the format of the `.processed` files we provide, OR of the `.txt` files we provide. It is your choice.

The outpt should be in the Annotation file format, with the original sentence in brackets being optional.

Work Cycle

You should use the training set's "Annotation" file to manually analyze examples of relations, which should be extracted from different sentences. You should use the "Corpus" file of the

training set in order to better understand the preprocessed data. Using the analyzed examples, you should devise a method for extracting the selected relation of interest. Note that identifying a relation includes identifying the participating entities, as well as the relation type.

You can use either a rule-based approach, a learning-based approach, or a hybrid approach. Note that the training set is quite small, so a learning-based approach could easily overfit.

When developing your system against the training set, keep in mind that the sentences in the training set are just examples, and try to make a system that can generalize beyond the development set.

You should follow the following methodology:

1. Use the provided training set to develop an initial RE system.
2. When done, test its performance for the given dev set. If the results are lower than you would expect based on the system's performance for the training set, you can return to (1) to perform error analysis (see below) over the training set and try to improve the system according to the understanding you gained. Note that the dev set should be used only to give you indication of how the system performs for the sentences, which were not seen before. Using it to analyze how to improve the performance is possible, but note that you may fall into over-fitting (your submitted system will be evaluated on a hidden set).
3. After several iterations, you will come up with a configuration of your system, which performs reasonably well both for the development and for the test sets. This will be the configuration which you'll submit for this assignment.

The results of your final system will be further evaluated using a hidden set, which is not published along with the assignment. The results achieved by your system for this blind validation set will account for 10% of your assignment grade. Note that the results of your system for the train and the dev sets, which you obtain as part of the data, will not influence your grade.

Error analysis

In order to better understand why your system makes mistakes, it is common to perform error analysis. One of the standard ways, which we suggest you to follow, is randomly sampling a number of recall mistakes and a number of precision mistakes, and manually analyzing them. In our setting, recall mistakes are relations, which your system did not succeed to extract. Precision mistakes are relations extracted by your system, but absent in the gold-standard annotation. It is useful to try and categorize the mistakes you analyzed into groups, according to the cause of the error. This will help you understand the problem domain better, and may provide hints as to how to improve your system (where to direct your efforts, or which components are needed but missing).

What to submit

You will submit two programs:

`eval file1 file2` this program gets as input two files in the "Annotations" format (bracketed

sentence is optional), and outputs the precision, recall and F1 scores for each relation category, assuming the first file is the gold-data and the second file is the predicted data.

If you work in Java, this should be run as: `java -jar eval.jar file1 file2`

If you work in python, this should be run as: `python eval.py file1 file2`

For a relation to be considered as identified correctly, it needs to match the type of the relation and the two entities in their entirety. It also needs to match the sentence number (if a relation appears in sentence A, but you identified it in sentence B, you will still lose recall for not identifying it in sentence A. And, if the relation does not really occur in sentence B, you will lose precision for incorrectly identifying it.)

`extract file` this program gets as input a file, and outputs a file in the Annotation format, containing the extracted relations.

If you work in Java, this should be run as: `java -jar extract.jar file1 file2`

If you work in python, this should be run as: `python extract.py file1 file2`

The input could be either in the format of the `.processed` files we provide, OR in the `.txt` files we provide. It is your choice. If the file is in the wrong format, your program should exit gracefully with an appropriate message. You can assume the first character in the `.processed` file is a `#`, and the first character in the `.txt` file is not.

You will also submit a `final report` file (named `report.pdf`), as described below.

Report:

You should prepare a short report, containing the following:

1. Your name and ID.
2. Short and clear description of your system. If you have a rule-based system, provide a concise description of each rule, the motivation behind that rule, and how the rule is implemented. If the rules have a specific order, explain this part as well. If you used a machine-learning approach, explain what you did (how you modeled the problem, what are features, how the system is trained). If you have used external resources, descrie what they are and how you used them.
3. Error analysis: can you identify the types of some common recall and precision errors in your program? Can you think of the causes for these kinds of errors?
4. Your evaluation results in the following table (you need to include only one of the rows):

Relation	Dev Recall	Dev Prec	Dev F1	Test Recall	Test Prec	Test F1
Work For						
Live In						

Pay attention: Penalty will be given for poorly written report, make it concise, informative and clear.

Misc

All code files as well as the report should include the username and ID of the submitting student(s), in a clear location in one of the first lines.

You can write your code in either Python or Java (or both).

In case you want to use WordNet, the Python NLTK package provides a convenient interface.

The code should be able to run from the commandline, without using Eclipse or any other IDE.

For Java, please submit both the `.java` source and the compiled `.class` files (and a `jar` file).

The code for all assignments should be submitted in a single `.zip` or `.tar.gz` file.

Please provide instructions on how to run your code (if special files need to be in certain location, special libraries need to be installed, and so on). If your code cannot be run, you will receive a grade of 0.

The `README` file with the instructions should be in an easy-to-find location, and in a plain text format.

The report should be in a `PDF` format.