Middle East Technical University | Department of Computer Engineering

CENG 463

Introduction to Natural Language Processing

Fall 2023-2024

Assignment 2

Regulations

1. The homework is due by **23:55 on January 14th, 2024**. Late submission is not allowed.
2. The homework consists of **2 independent tasks**.
3. Submissions are to be made via ODTUClass, do not send your homework via e-mail.
4. You can use any typesetting tool (LaTex, Word, etc.) while writing your report (but no handwriting). However, you **must** upload the report as a **searchable pdf file**. Other formats will not be considered for grading.
5. You **must** write your codes in **Python3**.
6. Upload 3 files: Your code for Task1 (either a Python or a txt file), your code for Task2, and your report (a single report file consisting of reports for Task1 and Task2).
7. Name your files as in the following:
   *yourStudentId_Task*1.*py* **or** *yourStudentId_Task*1.*txt*,
   *yourStudentId_Task*2.*py*,
   yourStudentId_Report.pdf
   (e.g. 1234567_Task1.py, 1234567_Task2.py, 1234567_Report.pdf).
   Submissions that violate the naming convention will incur a grade reduction of 10 points.
8. Send an e-mail to **garipler@metu.edu.tr** if you need to get in contact.
9. **This is an individual homework, which means you have to answer the questions on your own. Any contrary case including but not limited to getting help from automated tools, sharing your answers with each other, extensive collaboration etc. will be considered as cheating and university regulations about cheating will be applied.**

TASK 1: Dependency Parsing (60 pts)

For this task, you will employ spaCy library of Python, which considerably simplifies the task of dependency parsing. You are expected to train a dependency parser for Turkish and prepare a report about the implementation and performance of your parser. You will work with the data provided by the Universal Dependencies Treebanks for Turkish.

Refer to the documentation of spaCy in order to find out how to use spaCy for this task:
$https://spacy.io/api$.

For more information about the data, you can refer to:
$http://universaldependencies.org/format.html$
$https://github.com/UniversalDependencies/UD\_Turkish-IMST$

Submit/Report:

1. You may write either a Python script or call modules of spaCy one by one from your terminal. Submit your Python script or a txt file in which the calls you have made from the terminal are written in the right order.

2. In your reports, explain your implementation in detail.

3. Evaluate the final version of your parser on the test set. Report labeled and unlabeled attachment scores (spaCy easily does this.). What are those scores? Is there a remarkable difference between those two scores? Why? Explain in your reports. Do not skip any of the questions.

4. Experiment with your parser: Feed some sentences to your parser, observe correctness of the parsing. Report 2 (almost) totally correctly parsed and 2 poorly parsed sentences. What may cause those differences in the correctness of parsing? Report/discuss.

TASK 2: Extractive Summarization (40 pts)

You are provided with a document collection of Australian legal cases from the Federal Court of Australia (FCA). The document collection includes 4000 legal cases. You may use the whole collection or a restricted number of documents (at least 1000) for this task. If you chose the latter option, please report which documents you have used.

The task of extractive summarization can simply be defined as finding and returning the sentences that are most informative about the document they are contained in. To achieve this, in this specific task, you should calculate **Tf-Idf** weights of tokens in the whole document collection. Then for each document, simply calculate **cosine similarity** between sentences in document and the remaining of the document (i.e. employ cosine similarity with tf-idf weighting). Then, return the most informative 5 sentences as the summary of the document. At the end, your summarizer should take a file name as the input and print (to terminal) the summary of that file.

Documents are in xml format. Only use sentences in this task (They are tagged with the $< \backslash sentence >$ tag.). *There are also catchphrases in documents. Do not use them at any stage! Implementations making use of catchphrases will be heavily penalized.*

Submit/Report:

1. Submit your implementation of the task as a single Python file.
2. In your reports, explain your implementation in detail.
3. Experiment with your summarizer: Read at least 5 documents and summarize them using your summarizer. Are summaries sufficiently informative? How would you, as a human, make extractive summaries of those documents? Report the names of the 5 documents you have selected, together with your human-made extractive summaries of them (i.e. select most informative 5 sentences on your own). Is there a big difference between your summary and the automated summary in terms of informativeness? Why? Report.
4. How can this task be enhanced? Discuss. (You may make a short survey about extractive summarization methods.)
5. How can you evaluate such a task? (Again, survey required.)

Remarks

- You can ask anything about the assignment via e-mail or discussion forum.
- You are encouraged to use the discussion forum.
- Submit your code and report even if you cannot go far. You can get partial credit from even preliminary stages of implementation or unapplied ideas.