D2

Anna Batra, Sam Briggs, Junyin Chen, Hilly Steinmetz
Department of Linguistics, University of Washington
{batraa, briggs3, junyinc, hsteinm}@uw.edu

Abstract

This deliverable contains a skeleton of the paper and a list of team members. This is an example citation (Jurafsky and Martin, 2022). According to Jurafsky and Martin (2022), this is another example citation.

1 Introduction

2 Engines

3 System Overview

4 Approach

4.1 Data Pre-processing

4.1.1 Collecting raw text

We used the data from the TAC 2009, 2010, 2011 Shared Task data. To process the given XML files to retrieve the set of document names in docSetA, we used xml.etree.ElementTree. We then figured out which corpus data path matched each document in docSetA.

In order to read in these AQUAINT and AQUIANT2 files that are organized differently, we used lxml.etree as this can parse non-XML compliant files. We found there are three different organization methods that were used, and we made sure to process each kind uniquely. From these files, we parsed the the headline, time, and raw text paragraphs.

4.1.2 Tokenization

After getting the raw text, we used two different tokenization methods: spaCy and NLTK.

From spaCy 2.0 we used the English model tokenizer “en_core_web_sm". We decided on spaCy 2.0 since it fits the python version (Python 3.6) on Patas (the virtual machine we used). We decided not to tokenize the raw text into sentences first. Instead, we just ran a word tokenizer on the raw text. The spaCy tokenizer utilizes a transformer under the hood. We then realized that there was a sentence tokenizer, but since this tokenization method uses transformers on the CPU, it actually takes a really long time to run. Therefore, we decided to leave it as it is and turn to a rule-based tokenizer explained in the next method.

For NLTK, we used the English model tokenizer “tokenizers/punkt/english.pickle". We first ran a sentence tokenizer, and then for each tokenized sentence, we then ran a word tokenizer. This gave us a list of sentences, where each sentence contains tokenized words. The NLTK tokenizer is significantly faster than spaCy’s tokenizer and gives us the correct output we need per D2.

5 Results

6 Discussion

7 Conclusion

8 Appendix A: Workload distribution

8.1 D1 Workload

- Anna Batra set up the Github repository, turned in D1
- Junyin Chen got the team together and set up a communication channel
- Sam Briggs set up the Overleaf file and sent out a when-to-meet to schedule weekly meetings
- Hilly Steinmetz edited the Overleaf file to prepare it for D1.

8.2 D2 Workload

- Anna Batra and Sam Briggs wrote test code to test the file structure of the output docSets, created the outline for the presentation, and updated the report.

- Junyin Chen wrote code for tokenizing documents in the docSets using spaCy, PR reviewed the code to merge with Hilly's, cleaned up the code, and created slides for the pre-processing section.

- Hilly Steinmetz wrote the code for the pre-processing steps before tokenization, such as locating paths for AQUAINT and AQUAINT2 files. Hilly also wrote code for tokenizing documents using NLTK.

9 Appendix B: Code repository and additional software and data used in your system

9.1 Source Code

The repository for our project can be found on Github at github.com/LING-575-Summarization/Summarization.

9.2 Packages

We used the following packages for our system:

9.2.1 Pre-processing

- lxml.etree (for processing AQUAINT, AQUAINT2, TAC files)

- xml.etree.ElementTree (for processing docSetA file lists)

- spaCy 2.0 (for word tokenization on paragraphs)
  - English model “en_core_web_sm"

- NLTK (for sentence and word tokenization)
  - English model
    'tokenizers/punkt/english.pickle'

9.3 Future possible packages

We plan to use the following software for the project:

- Python

- Anaconda (for virtual environment)

- NLTK (for parsing, NER, and more)

- spaCy (for parsing, NER, and more)

- Scikit-learn (for machine learning models)

- Pytorch (for large LMs)

- Hugging Face's Transformer library (for auto-tokenization and find-tune trained model)

References

Dan Jurafsky and James H. Martin. 2022. *Speech and Language Processing*, 3rd edition (draft) edition. Online.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop: Automatic Summarization*.