Summarization Project Report: Deliverable 2

Long Cheng, Sheng Bi, Catherine Wang, Vicky Xiang, Carrie Yuan
University of Washington
`{lcheng97, shengbi, qywang, xinyix7, jiayiy9}@uw.edu`

Abstract

1 Introduction

2 System Overview

3 Approach

3.1 Preprocessing

Our training, dev, and evaluation sets are taken from the TAC 2009/2010/2011 shared task data, which gives lists of documents organized by topic. Our data prepocessing involves two steps: first, we extract topics and documents from the TAC shared task data files through our script `extract.py`, then parse, format, and preprocess the data in each document file. This is implemented in the scripts `{training | dev | eval}_process.py`, which handle formatting and preprocessing for the training, dev, and evaluation splits respectively.

The extraction process involved parsing the TAC data files in XML format, then extracting topic and document IDs. Once `docID`s are extracted, the local path for each document is generated based on the file name, year, and directory. This allowed us to get each file name for all documents in each split of our data.

Once we have the set of articles for our training, dev, and test data, we implement preprocessing by parsing the files from their original XML format and writing out plain text files for each document that contain the preprocessed data. Due to the variations in formatting of the original data, our preprocessing is split into three separate scripts for the training, dev, and evaluation data respectively. These scripts utilize `xml.etree.ElementTree` to parse the XML files, and manually modify the files (by adding `<DOCSTREAM>` tags) if they are not originally in standard XML format. For nonstandard file formats, such as some files in the dev split, we also had to reformat text including replacing `‘&’` with `'and'`, which may impact the summarization system. Once the XML data is parsed, each document is found by its `docID`, and relevant information such as headline and dateline are extracted. Finally, each document is segmented by sentence, and tokenized using `nltk.word_tokenize`. The output files are sorted into subdirectories by topic, and each outputted file is a plain text document that includes the headline, dateline, and segmented and tokenized text of the document body.

4 Results

5 Discussion

6 Conclusion

7 References

A Team Members and Workload Distribution

- Sheng Bi: wrote `extract.py` script for extracting file names from XMLs.
- Long (Victor) Cheng: wrote `{training | dev | eval}_process.py` scripts for formatting the original data files, getting documents from XML files and implementing preprocessing.
- Catherine Wang: made `.sh` scripts to get files given by the file name outputs from `extract.py`; wrote report.
- Vicky Xiang: conducted literature review for summarization task; handled project organization on GitHub; edited and reviewed `extract.py`; responsible for presenting in class.
- Carrie Yuan: made presentation slides.

B Additional software and data

Off-the-shelf tools we have used include:

- NLTK for tokenization
- `xml.etree.ElementTree` for analyzing XML files