LING575 Summarization

2024.01

#D2
Process a docSet

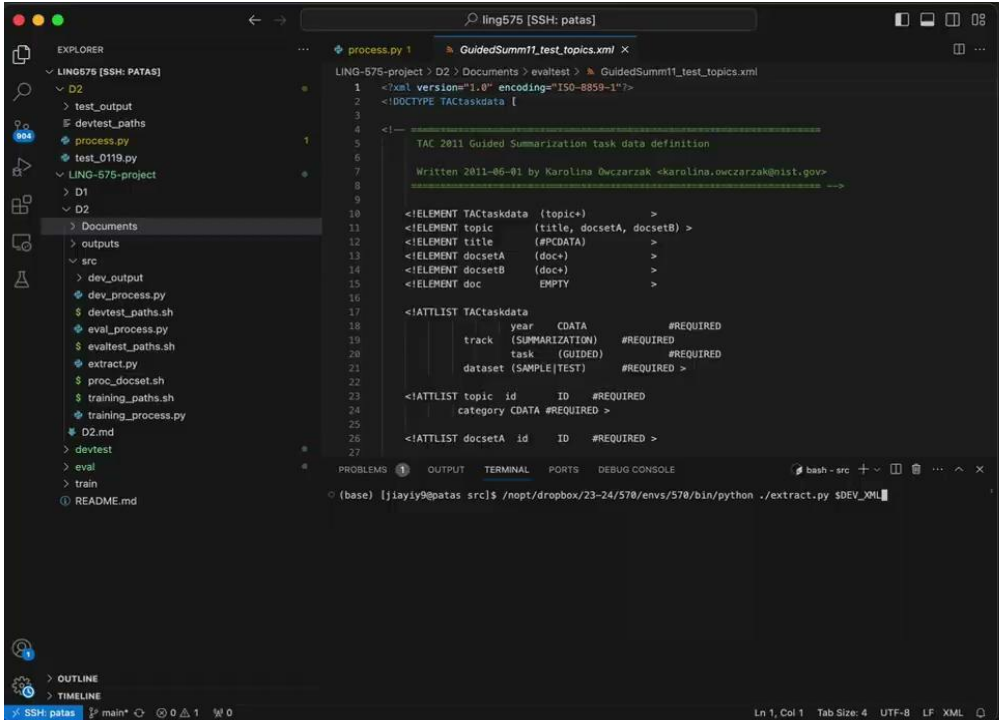
Overview

- Process XML -> Retrieve corresponding articles from DocSets
- Process Articles: Segment paragraph into sentences, tokenize sentences
- Summarization System Plan Overview

extract.py

- Use XML.etree.ElementTree library to parse XML
- Extract topic ids and corresponding doc id from DocSetA
    - topic.docSetA.id
- Transform doc id into path
    - Case by case according to doc id format and year, but overall it is extracting dir_name, year and file_name, and generate the local path accordingly
    - e.g.
        - APW19990914.0234 -> /corpora/LDC/LDC02T31/apw/1999/19990914_APW_ENG
        - APW_ENG_20050609.0625 -> /corpora/LDC/LDC08T25/data/apw_eng/apw_eng_200506.xml

Demo Video – 1



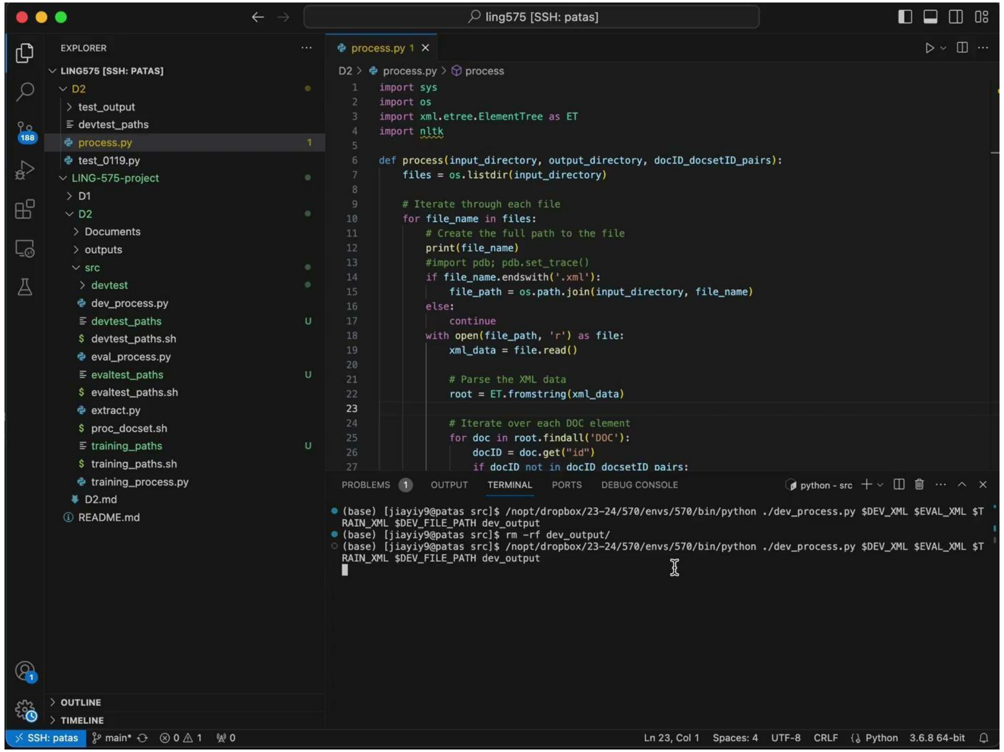
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE TACtaskdata [

<!-- ======================================================================
      TAC 2011 Guided Summarization task data definition

      Written 2011-06-01 by Karolina Owczarzak <karolina.owczarzak@nist.gov>
    ====================================================================== -->

    <!ELEMENT TACtaskdata  (topic+)           >
    <!ELEMENT topic       (title, docsetA, docsetB) >
    <!ELEMENT title       (#PCDATA)           >
    <!ELEMENT docsetA     (doc+)              >
    <!ELEMENT docsetB     (doc+)              >
    <!ELEMENT doc          EMPTY              >

    <!ATTLIST TACtaskdata
                    year    CDATA               #REQUIRED
            track   (SUMMARIZATION)     #REQUIRED
                    task    (GUIDED)            #REQUIRED
            dataset (SAMPLE|TEST)       #REQUIRED >

    <!ATTLIST topic  id       ID    #REQUIRED
           category CDATA #REQUIRED >

    <!ATTLIST docsetA  id     ID    #REQUIRED >
```

```
(base) [jiayiy9@patas src]$ /nopt/dropbox/23-24/570/envs/570/bin/python ./extract.py $DEV_XML
```

process.py

- Use XML.etree.ElementTree library to parse XML
- If the file is not a standard XML (e.g. /corpora/LDC/LDC02T31/apw/1999/19990914_APW_ENG), then modify the file by adding <DOCSTREAM> tag
    - Find doc id by looking for DOCNO keyword
- If the file is standard XML, find the doc by looking for id keyword.
- Get headline and dateline from HEADLINE and DATELINE keyword.
- Get individual sentence by doc.TEXT.findall("P")
- Use nltk.word_tokenize to tokenize the sentences

Demo Video – 2



```python
import sys
import os
import xml.etree.ElementTree as ET
import nltk

def process(input_directory, output_directory, docID_docsetID_pairs):
    files = os.listdir(input_directory)

    # Iterate through each file
    for file_name in files:
        # Create the full path to the file
        print(file_name)
        #import pdb; pdb.set_trace()
        if file_name.endswith('.xml'):
            file_path = os.path.join(input_directory, file_name)
        else:
            continue
        with open(file_path, 'r') as file:
            xml_data = file.read()

            # Parse the XML data
            root = ET.fromstring(xml_data)

            # Iterate over each DOC element
            for doc in root.findall('DOC'):
                docID = doc.get("id")
                if docID not in docID_docsetID_pairs:
```

```
(base) [jiayiy9@patas src]$ /nopt/dropbox/23-24/570/envs/570/bin/python ./dev_process.py $DEV_XML $EVAL_XML $T
RAIN_XML $DEV_FILE_PATH dev_output
(base) [jiayiy9@patas src]$ rm -rf dev_output/
(base) [jiayiy9@patas src]$ /nopt/dropbox/23-24/570/envs/570/bin/python ./dev_process.py $DEV_XML $EVAL_XML $T
RAIN_XML $DEV_FILE_PATH dev_output
```

Methods we are considering for summarization

- TF-IDF
  - Create a Document-Term Matrix using TF-IDF
  - Score the sentences using TF-IDF score for each term
- LLR
  - Create a Document-Term Matrix using LLR
  - Score the sentences using LLR score for each term
- LSI & LDA topic modelling
  - Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA)
  - Used to extract topics from a collection of documents, and the topics can be used as feature
  - Can use key terms or phrases associated with dominant topics as content

Methods we are considering for summarization

Method	Pros	Cons
TF-IDF	- Simple and Intuitive - Efficient for Extractive Summarization	- Rely heavily on word overlap - Don't handle synonyms - Sparse representation - Sentence redundancy - Lack of Context Understanding
LLR	- More nuanced analysis for context - Handles Synonyms and Specific Contexts	- Data sensitivity - Not Suitable for Abstractive Summarization involving rephrasing and paraphrasing.
LSI/LDA topic modeling	- Semantic Understanding - Documents can be represented as a mixture of topics	- Require additional techniques to apply those extracted features - Dimensionality Reduction, potentially loss of information - Sensitivity to Hyperparameters