

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

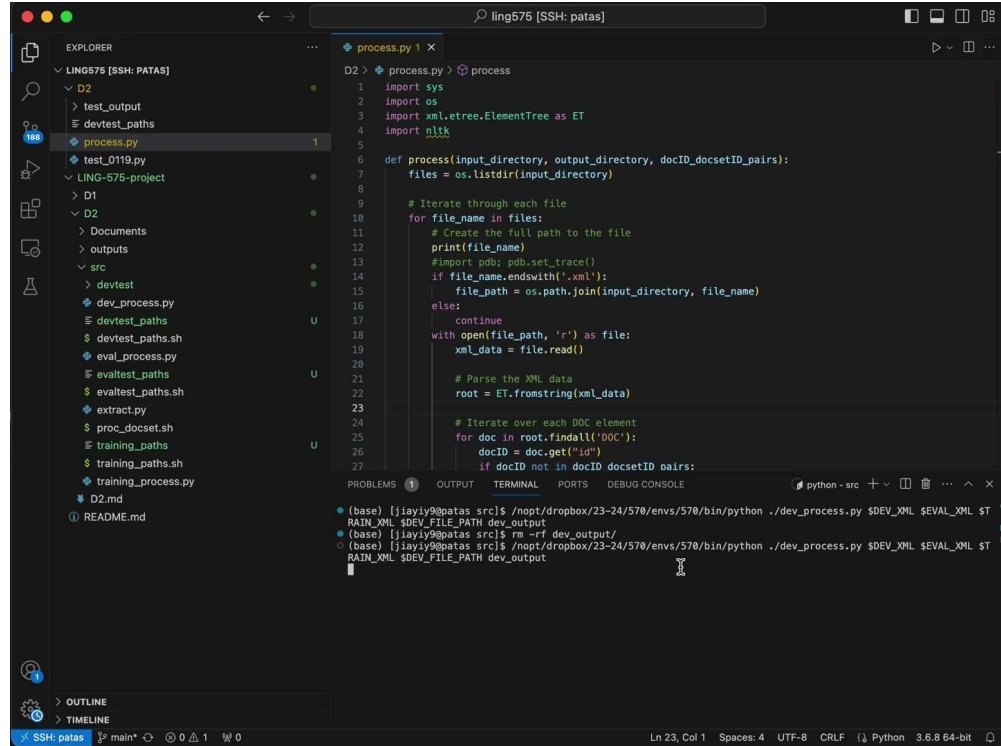
Demo Video - 1

The screenshot shows a VS Code editor interface. The left sidebar contains a file explorer for a project named 'LING-575 [SSH: PATAS]'. The project structure includes a 'D2' directory with subdirectories 'src' and 'outputs'. The 'src' directory contains files like 'dev_test_paths', 'eval_process.py', 'evaltest_paths.sh', 'extract.py', 'proc_docset.sh', 'training_paths.sh', 'training_process.py', and 'D2.md'. The 'outputs' directory is also visible. The main editor window displays the content of 'GuidedSummT1_test_topics.xml'. The XML file is a TAC 2011 Guided Summarization task data definition. It includes a header with version '1.0' and encoding 'ISO-8859-1'. The title is 'TAC 2011 Guided Summarization task data definition'. The docsetA is 'Written 2011-06-01 by Karolina Owczarzak <karolina.owczarzak@nist.gov>'. The docsetB is 'EMPTY'. The XML also defines a list of topics and a list of datasets. The status bar at the bottom shows the current file is 'SSH: patas' and the editor is in 'main' mode.

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



The screenshot displays the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project structure for 'LING575 [SSH: PATAS]' with subdirectories 'D2', 'test_output', 'devtest_paths', and 'process.py'. The code editor shows the contents of 'process.py', which is a Python script that iterates through files in a directory, parses XML data, and prints the results. The terminal shows the output of the script, which is a list of document IDs and their corresponding docset IDs.

```
process.py 1 X
D2 > process.py > process
1 import sys
2 import os
3 import xml.etree.ElementTree as ET
4 import nltk
5
6 def process(input_directory, output_directory, docID_docsetID_pairs):
7     files = os.listdir(input_directory)
8
9     # Iterate through each file
10    for file_name in files:
11        # Create the full path to the file
12        print(file_name)
13        #import pdb; pdb.set_trace()
14        if file_name.endswith('.xml'):
15            file_path = os.path.join(input_directory, file_name)
16        else:
17            continue
18        with open(file_path, 'r') as file:
19            xml_data = file.read()
20
21        # Parse the XML data
22        root = ET.fromstring(xml_data)
23
24        # Iterate over each DOC element
25        for doc in root.findall('DOC'):
26            docID = doc.get("id")
27            if docID not in docID_docsetID_pairs:
```

```
(base) [jiayi9@patas src]$ /n/pt/dropbox/23-24/570/envs/570/bin/python ./dev_process.py $DEV_XML $EVAL_XML $T
RAIN_XML $DEV_FILE_PATH dev_output
(base) [jiayi9@patas src]$ rm -rf dev_output/
(base) [jiayi9@patas src]$ /n/pt/dropbox/23-24/570/envs/570/bin/python ./dev_process.py $DEV_XML $EVAL_XML $T
RAIN_XML $DEV_FILE_PATH dev_output
```

Ln 23, Col 1 Spaces: 4 UTF-8 CRLF Python 3.6.8 64-bit

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	<ul style="list-style-type: none">- Simple and Intuitive- Efficient for Extractive Summarization	<ul style="list-style-type: none">- Rely heavily on word overlap- Don't handle synonyms- Sparse representation- Sentence redundancy- Lack of Context Understanding
LLR	<ul style="list-style-type: none">- More nuanced analysis for context- Handles Synonyms and Specific Contexts	<ul style="list-style-type: none">- Data sensitivity- Not Suitable for Abstractive Summarization involving rephrasing and paraphrasing.
LSI/LDA topic modeling	<ul style="list-style-type: none">- Semantic Understanding- Documents can be represented as a mixture of topics	<ul style="list-style-type: none">- Require additional techniques to apply those extracted features- Dimensionality Reduction, potentially loss of information- Sensitivity to Hyperparameters