Tech Review

CS 410, Fall 2022

Names (NetIDs) of team members

- Adam Florzak (aflor3)

Topic

- TfidfVectorizer

Category

- Useful software toolkits for processing text data or building text data applications.

Description

- Explore how to replicate certain MeTA functionality using sklearn's TfidfVectorizer

Introduction

The MPs for this course leverage metapy, a Python binding for MeTA. However, metapy can be difficult to install and does not appear to be widely used or recently maintained. For that reason, it would be advantageous to identify an alternative software toolkit for processing text data that is able to replicate certain MeTA functionality.

This review attempts to replicate the MP2.4 implementation using TfidfVectorizer from sklearn, and in the process, demonstrates some of the toolkit's most useful features. The example code can be found in the accompanying notebook on GitHub, MP2_4_TfidfVectorizer.ipynb.

Methods

This review uses the MP2.4 baseline metapy code with the addition of a custom BM25 ranker. The rationale for using a custom BM25 ranker as opposed to metapy's included ranker, metapy.index.OkapiBM25(k1=1.2,b=0.75,k3=500), is to control the algorithm results for a better side-by-side comparison between the two toolkits. A two parameter (k1=1.2, b=0.75) BM25 implementation was chosen for this purpose, following the formula found on Wikipedia.

Term IDF is easily obtained in TfidfVectorizer via:

```
vectorizer.idf_
```

However, the IDF formula used by TfidfVectorizer does not align exactly with the results from metapy, so this review makes use of a custom IDF calculation to maintain consistency between the two toolkits.

This review also uses the formula for calculating nDCG mentioned in the metapy documentation, which differs slightly from the standard formula:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \quad p = num\_docs$$

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

Ultimately, both the metapy and TfidfVectorizer implementations were successfully tested against MP2.4's Cranfield dataset containing 1,400 documents and 225 queries.

Results

While TfidfVectorizer supports using stop words, it was first necessary to tokenize the stop words using a temporary instance of TfidfVectorizer. Once the tokenized stop word features were extracted and converted to a list, they could then be input into the main instantiation of TfidfVectorizer.

The TfidfVectorizer also returns a term frequency (TF) sparse matrix by default. To obtain a term count matrix, it is necessary to use either the CountVectorizer toolkit instead or to leverage this workaround:

```
idx_count = super(TfidfVectorizer, vectorizer).transform(corpus)
```

From there, it is relatively straightforward to begin BM25 scoring and compute nDCG for the queries, because TfidfVectorizer returns familiar formats such as scipy sparse matrices and NumPy arrays.

The average nDCG for the Cranfield dataset using metapy is 0.267. By comparison, TfidfVectorizer achieves a better average nDCG of 0.343 using the same dataset, queries, and BM25 parameters.

Discussion

The TfidfVectorizer toolkit is relatively easy to use and can easily be extended to replicate metapy functionality. The biggest advantage that TfidfVectorizer has over metapy is that since it belongs to the larger sklearn ecosystem, there is much better support and documentation.

Additionally, sklearn offers related text feature extraction toolkits for such as CountVectorizer, HashingVectorizer, and TfidTransformer. All of these integrate well with the other sklearn toolkits as well as with popular packages such as NumPy.

Sources

"sklearn.feature_extraction.text.TfidfVectorizer" skikit-learn API. Accessed 6 Nov 2022. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

"MP2_4_TfidfVectorizer.ipynb" GitHub, aflorzak. Accessed 6 Nov 2022. https://github.com/aflorzak/CS410-tech_review/blob/main/MP2_4_TfidfVectorizer.ipynb

"Okapi BM25" Wikipedia. Accessed 6 Nov 2022. https://en.wikipedia.org/wiki/Okapi_BM25

"meta::index::ir_eval Class Reference" ModErn Text Analysis. Accessed 6 Nov 2022. https://meta-toolkit.org/doxygen/classmeta_1_1index_1_1ir__eval.html#a23827b8671dffbfbc85494def49d66c1