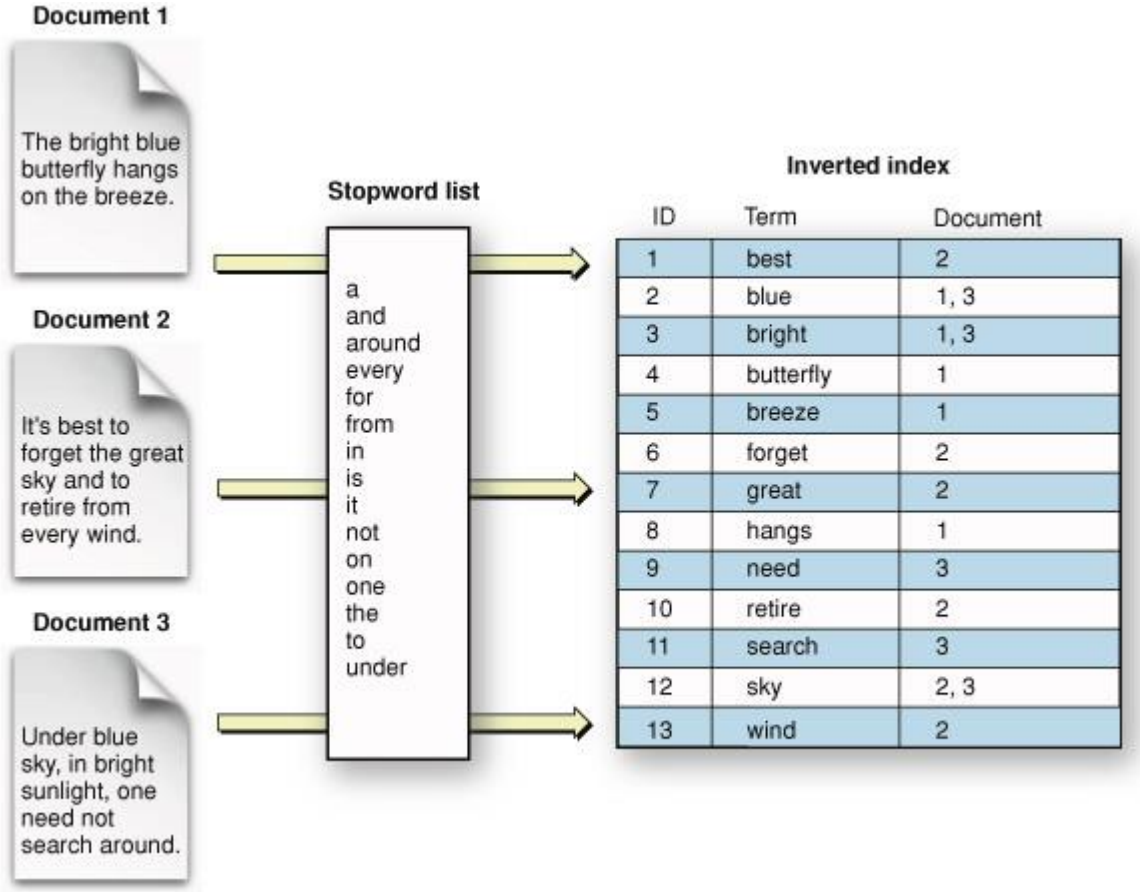
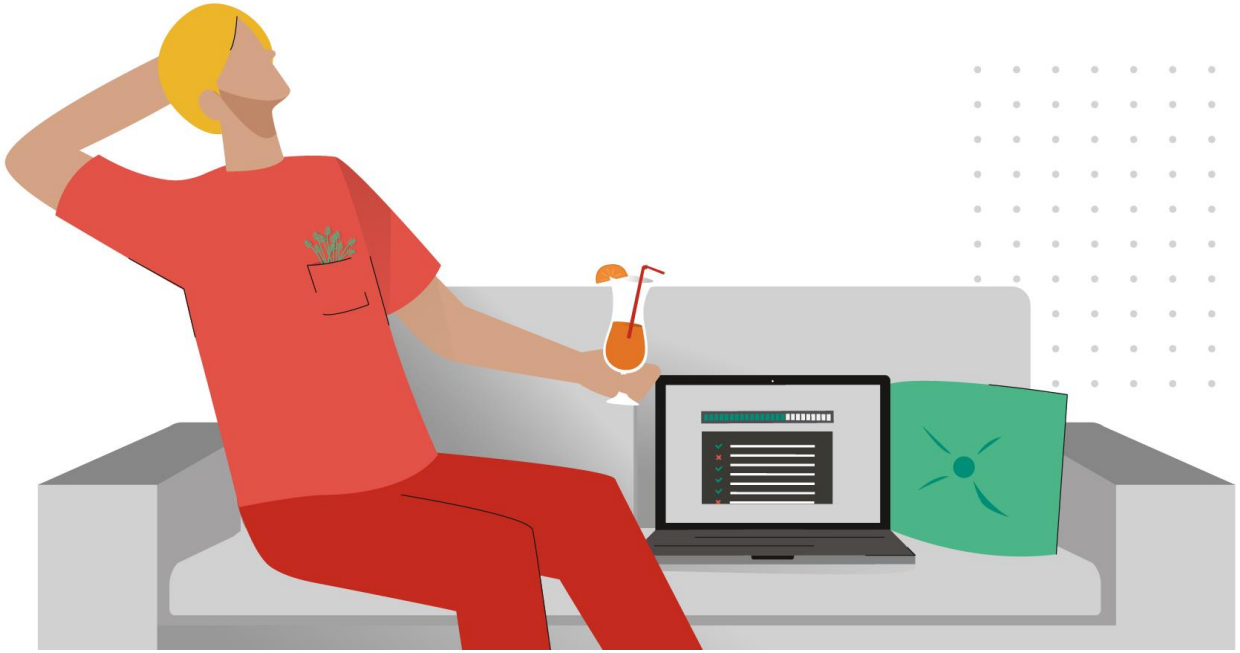
Data Engineering LAB: Python Module

Build Inverted Index

An inverted index is a dictionary, where keys are words(terms) and values are indexes of documents where these keys have occurred. This behavior helps to recommended systems to be faster during the matching and providing us with the most relevant documents.



Typical documents journey to build an Inverted Index



Import
The script has to be able to read the *.txt file to the RAM

Text cleaning
Delete stop words from each document

Inverted Index
Prepare index dictionary based-on cleaned docs

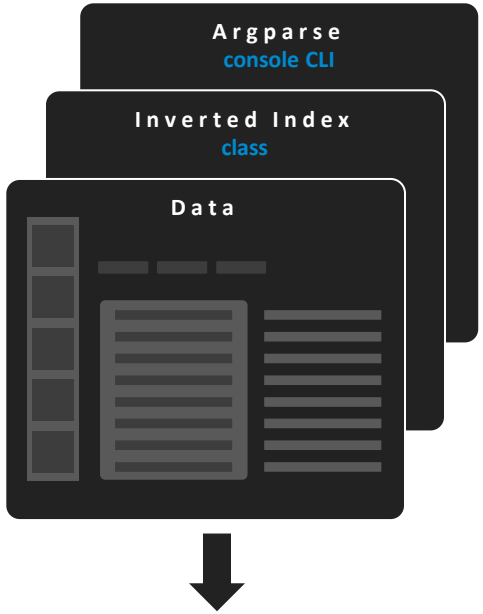
Save Inverted Index
Dump dictionary to the *.json file

Load Inverted Index
Load inverted index from *.json

Enter query
Create a functional to choose a format of entering a query of words to find the necessary docs.

Available formats
By default - stdin.
Optional – File.

Print the result
The script has to be able to show the doc ids in stdout or stderr.



Build class “InvertedIndex” that is able to:

1. **dump** the inverted index to the *.json file;
2. **load** the inverted index from *.json file;
3. **process query** of words to find corresponding doc ids.

External functions:

1. **load_doc** – read the file with documents.
2. **build_inverted_index** – use the result of “load_doc” function and build inverted index based-on docs.
3. **setup_parser** – create argparser to simplify I/O operations.

Data

FILE: wikipedia_sample

Each row of this file consists of:
1. Document id – int
2. Text – string

P.S. Divided by tabulation

Class Inverted Index

```
InvertedIndex.query(words: List[str]) -> List[int]
InvertedIndex.dump(filepath: str) -> None
InvertedIndex.load(filepath: str) -> InvertedIndex
```

Global functions

```
load_doc(filepath: str) -> Dict[int, str]
build_inverted_index(doc: load_doc(...)) -> InvertedIndex
```

Argparse

```
$ python inv_index.py build—dataset path/to/dataset –output path/to/save

$ python inv_index.py query --index path/to/index.json --query [<word> <word> … <word>] --from_file path/to/file_with_query
```

Argparse Lib

mode 1 `$ python inv_index.py build—dataset path/to/dataset --output path/to/save`

mode 2 `$ python inv_index.py query --index path/to/index.json --query [<word> <word> … <word>] --from_file path/to/file_with_query`

advice build, query – subparsers. See the useful materials (1).

advice --query [<word> <word> … <word>] – words to find necessary doc ids. See the useful materials (2).

notice The script have to be able to take the query of words from either sys.stdin or File. By default – stdin, therefore use the following code on the “default” parameter in the “add_argument”.

```
default=TextIOWrapper(sys.stdin.buffer, encoding='utf-8')
```

“WIKIPEDIA_SAMPLE”

332 Animalia (book) Animalia (ISBN 0810918684) is an illustrated children's book by Graeme Base. It was published in 1986. Animalia is an alliterative alphabet book and contains twenty-six illustrations, one for each letter of the alphabet.

12 Anarchism Anarchism is often defined as a political philosophy which holds the state to be undesirable, unnecessary, or harmful.

Consistency

To ensure the same result you need to use the ↓ following code ↓ for breaking down *.txt file into docs.

```
doc_id, content = line.lower().split("\t", 1)
doc_id = int(doc_id)
words = re.split(r"\W+", content)
```

USEFUL MATERIALS

1. Subparsers
2. Parameter “nargs”
3. Introduction to argparse lib
4. Inverted index definition
5. Dataset of documents
6. Stop words