Lab : Exploring the 20 Newsgroups Dataset with Text Analysis Techniques

We went through a bunch of fundamental machine learning concepts in the previous chapter. We learned about them along with analogies, in a fun way, such as studying for exams and designing a driving schedule. Starting from this chapter as the second step of our learning journal, we will be discovering in detail several important machine learning algorithms and techniques. Beyond analogies, we will be exposed to and solve real-world examples, which makes our journey more interesting. We will start with a natural language processing problem—exploring newsgroups data. We will gain hands-on experience in working with text data, especially how to convert words and phrases into machine-readable values and how to clean up words with little meaning. We will also visualize text data by mapping it into a two-dimensional space in an unsupervised learning manner.

We will go into detail for each of the following topics:

- What is NLP and its applications
- NLP basics
- Touring Python NLP libraries
- Tokenization
- Part-of-speech tagging
- Named entities recognition
- Stemming and lemmatization
- Getting and exploring the newsgroups data
- Data visualization using seaborn and matplotlib
- The Bag of words (BoW) model and token count vectorization
- Text preprocessing
- Stop words removal
- Dimensionality reduction
- T-SNE
- T-SNE for text visualization

Pre-reqs:

- Docker

Lab Environment

We will run Jupyter Notebook as a Docker container. This setup will take some time because of the size of the image. Run the following commands one by one:

```
docker run -d --user root -p 8888:8888 --name jupyter -e GRANT_SUDO=yes jupyter/tensorflow-notebook:2ce7c06a61a1 start-notebook.sh
```

```
docker exec -it jupyter bash -c 'cd /home/jovyan/work && git clone https://github.com/athertahir/python-machine-learning-by-example.git && sudo && chmod +x ~/work/prepareContainer.sh && ~/prepareContainer.sh'
```

```
docker restart jupyter
```

Note: After completing these steps, jupyter notebook will be accessible at port 8888 of the host machine.

All Notebooks are present in `work` folder.

Login

When the container is running, execute this statement:

`docker logs jupyter 2>&1 | grep -v "HEAD"`

This will show something like:

```
Copy/paste this URL into your browser when you connect for the first time, to login with a token:
    http://localhost:8888/?token=f89b02dd78479d52470b3c3a797408b20cc5a11e067e94b8
    THIS IS NOT YOUR TOKEN.  YOU HAVE TO SEARCH THE LOGS TO GET YOUR TOKEN
```

The token is the value behind `/?token=`. You need that for logging in.

Note: You can also run following command to get token directly:
`docker exec -it jupyter bash -c 'jupyter notebook list' | cut -d'=' -f 2 | cut -d' ' -f 1`

How computers understand language - NLP

Machine learning driven programs or computers are good at discovering event patterns by processing and working with data. When the data is well structured or well defined, such as in a Microsoft Excel spreadsheet table and relational database table, it is intuitively obvious why machine learning is better at dealing with it than humans. Computers read such data the same way as humans, for example, revenue: 5,000,000 as the revenue being 5 million and age: 30 as age being 30; then computers crunch assorted data and generate insights. However, when the data is unstructured, such as words with which humans communicate, news articles, or someone's speech in French, it seems computers cannot understand words as well as human do (yet).

There is a lot of information in the world is words or raw text, or broadly speaking, natural language. This refers to any language humans use to communicate with each other. Natural language can take various forms, including, but not limited to, the following:

- Text, such as a web page, SMS, email, and menus
- Audio, such as speech and commands to Siri
- Signs and gestures
- Many others such as songs, sheet music, and Morse code

The list is endless and we are all surrounded by natural language all of the time (that's right, right now as you are reading this book). Given the importance of this type of unstructured data, natural language data, we must have methods to get computers to understand and reason with natural language and to extract data from it. Programs equipped with natural language processing techniques can already do a lot in certain areas, which already seems magical!

Natural language processing (NLP) is a significant subfield of machine learning, which deals with the interactions between machines (computers) and human (natural) languages. Natural languages are not limited to speech and conversation; they can be in writing or sign languages as well. The data for NLP tasks can be in different forms, for example, text from social media posts, web pages, even medical prescriptions, or audio from voice mail, commands to control systems, or even a favorite song or movie. Nowadays, NLP has been broadly involved in our daily lives: we cannot live without machine translation; weather forecasts scripts are automatically generated; we find voice search convenient; we get the answer to a question (such as what is the population of Canada) quickly thanks to intelligent question-answering systems; speech-to-text technology helps people with special needs.

If machines are able to understand language like humans do, we consider them intelligent. In 1950,the famous mathematician Alan Turing proposed in an article, Computing Machinery and Intelligence,a test as a criterion of machine intelligence. It's now called the Turing test, and its goal is to examine whether a computer is able to adequately understand languages so as to fool humans into thinking that this machine is another human. It is probably no surprise to us that no computer has passed the Turing test yet. But the 1950s is considered when the history of NLP started.

Understanding language might be difficult, but would it be easier to automatically translate texts from one language to another? In my first ever programming course, the lab booklet had the algorithm for coarse machine translation. We could imagine that this type of translation involves looking something up in dictionaries and generating new text. A more practically feasible approach would be to gather texts that are already translated by humans and train a computer program on these texts. In 1954, scientists

claimed, in the Georgetown experiment, that machine translation would be solved in three to five years. Unfortunately, a machine translation system that can beat human expert translators does not exist yet. But machine translation has been greatly evolving since the introduction of deep learning and has incredible achievements in certain areas, for example, social media (Facebook open sourced a neural machine translation system), real-time conversation (Skype, SwiftKey Keyboard, and Google Pixel Buds), and image-based translation.

Conversational agents, or chatbots, are another hot topic in NLP. The fact that computers are able to have a conversation with us has reshaped the way businesses are run. In 2016, Microsoft's AI chatbot, Tay, was unleashed to mimic a teenage girl and converse with users on Twitter in real time. She learned how to speak from all things users posted and commented on Twitter. However, she was overwhelmed by tweets from trolls, and automatically learned their bad behaviors and started to output inappropriate things on her feeds. She ended up being terminated within 24 hours.

There are also several tasks attempting to organize knowledge and concepts in such a way that they become easier for computer programs to manipulate. The way we organize and represent concepts is called ontology. An ontology defines concepts and relations between concepts. For instance, we can have a so-called triple representing the relation between two concepts, such as Python is a language.

An important use case for NLP at a much lower level, compared to the previous cases, is part-of-speech (PoS) tagging. A part of speech is a grammatical word category such as noun or verb. PoS tagging tries to determine the appropriate tag for each word in a sentence or a larger document. The following table gives examples of English POS:

Part of speech	Examples
`Noun`	David, machine
`Pronoun`	Then, her
`Adjective`	Awesome, amazing
`Verb`	Read, write
`Adverb`	Very, quite
`Preposition`	Out, at
`Conjunction`	And, but
`Interjection`	Unfortunately, luckily
`Article`	A, the

Picking up NLP basics while touring popular NLP libraries

After a short list of real-world applications of NLP, we'll be touring the essential stack of Python NLP libraries in this chapter. These packages handle a wide range of NLP tasks as mentioned previously as well as others such as sentiment analysis, text classification, and named entity recognition.

The most famous NLP libraries in Python include the Natural Language Toolkit (NLTK), spaCy, Gensim, and TextBlob. The scikit-learn library also has impressive NLP-related features. Let's take a look at the following popular NLP libraries in Python:

nltk: This library (http://www.nltk.org/) was originally developed for educational purposes and is now being widely used in industries as well. It is said that you can't talk about NLP without mentioning NLTK. It is one of the most famous and leading platforms for building Python-based NLP applications. You can install it simply by running the following command line in terminal:

```
sudo pip install -U nltk
```

SpaCy: This library (https://spacy.io/) is a more powerful toolkit in the industry than NLTK. This is mainly for two reasons: one, spaCy is written in Cython, which is much more memory-optimized (now you see where the Cy in spaCy comes from) and excels

in NLP tasks; second, spaCy keeps using state-of-the-art algorithms for core NLP problems, such as, convolutional neural network (CNN) models for tagging and name entity recognition. But it could seem advanced for beginners. In case you're interested, here's the installation instructions.

Run the following command line in the terminal:

```
pip install -U spacy
```

Gensim: This library (https://radimrehurek.com/gensim/), developed by Radim Rehurek, has been gaining popularity over recent years. It was initially designed in 2008 to generate a list of similar articles given an article, hence the name of this library (generate similar—> Gensim). It was later drastically improved by Radim Rehurek in terms of its efficiency and scalability. Again, we can easily install it via pip by running the following command line:

```
pip install --upgrade gensim
```

Note: You should make sure the dependencies, NumPy and SciPy, are already installed before gensim.

TextBlob: This library (https://textblob.readthedocs.io/en/dev/) is a relatively new one built on top of NLTK. It simplifies NLP and text analysis with easy-to-use built-in functions and methods, as well as wrappers around common tasks. We can install TextBlob by running the following command line in the terminal:

```
pip install -U textblob
```

TextBlob has some useful features that are not available in NLTK (currently), such as spell checking and correction, language detection, and translation.

Corpus

As of 2018, NLTK comes with over 100 collections of large and well-structured text datasets, which are called corpora in NLP. Corpora can be used as dictionaries for checking word occurrences and as training pools for model learning and validating. Some useful and interesting corpora include Web Text corpus, Twitter samples, Shakespeare corpus sample, Sentiment Polarity, Names corpus (it contains lists of popular names, which we will be exploring very shortly), WordNet, and the Reuters benchmark corpus. The full list can be found at http://www.nltk.org/nltk_data. Before using any of these corpus resources, we need to first download them by running the following command:

```
python -m nltk.downloader all
```

Run Notebook

The Notebook opens in a new browser window. You can create a new notebook or open a local one. Check out the local folder `work/Chapter02` for several sample notebooks. Open and run `exploring_nlp.ipynb` in the `work` folder.

You can open the Jupyter Notebook at `<host-ip>:8888/notebooks/work/Chapter02/exploring_nlp.ipynb`

Getting and Exploring the newsgroups data

Run Notebook

The Notebook opens in a new browser window. You can create a new notebook or open a local one. Check out the local folder `work/Chapter02` for several sample notebooks. Open and run `getting_exploring_newsgroups.ipynb` in the `work` folder.

You can open the Jupyter Notebook at `<host-ip>:8888/notebooks/work/Chapter02/getting_exploring_newsgroups.ipynb`

Thinking about features for text data

From the preceding analysis, we can safely conclude that, if we want to figure out whether a document was from the rec.autos newsgroup, the presence or absence of words such as car, doors, and bumper can be very useful features. The presence or not of a word is a boolean variable, and we can also propose looking at the count of certain words. For instance, car occurs multiple times in the document. Maybe the more times such a word is found in a text, the more likely it is that the document has something to do with cars.

Run Notebook

The Notebook opens in a new browser window. You can create a new notebook or open a local one. Check out the local folder `work/Chapter02` for several sample notebooks. Open and run `thinking_about_features.ipynb` in the `work` folder.

You can open the Jupyter Notebook at `<host-ip>:8888/notebooks/work/Chapter02/thinking_about_features.ipynb`

Visualizing the newsgroups data with t-SNE

We have just converted text from each raw newsgroup document into a sparse vector of a size of 500. For a vector from a document, each element represents the number of times a word token occurring in this document. Also, these 500 word tokens are selected based on their overall occurrences after text preprocessing, removal of stop words, and lemmatization. Now you may ask questions such as, is such occurrence vector representative enough, or does such an occurrence vector convey enough information that can be used to differentiate the document itself from documents on other topics? We can answer these questions easily by visualizing those representation vectors—we did a good job if document vectors from the same topic are nearby. But how? They are of 500 dimensions, while we can visualize data of at most three dimensions. We can resort to t-SNE for dimensionality reduction.

What is dimensionality reduction?

Dimensionality reduction is an important machine learning technique that reduces the number of features and, at the same time, retains as much information as possible. It is usually performed by obtaining a set of new principal features.

As mentioned before, it is difficult to visualize data of high dimension. Given a three-dimensional plot, we sometimes don't even find it too straightforward to observe any findings, not to mention 10, 100, or 1,000 dimensions. Moreover, some of the features in high dimensional data may be correlated and, as a result, bring in redundancy. This is why we need dimensionality reduction.

Dimensionality reduction is not simply taking out a pair of two features from the original feature space. It is transforming the original feature space to a new space of fewer dimensions. The data transformation can be linear, such as the famous one principal component analysis (PCA), which maximizes the variance of projected data, or nonlinear, such as neural networks and t-SNE coming up shortly. For instance, in PCA, it maps the data in a higher dimensional space to a lower dimensional space where the variance of the data is maximized. Non-negative matrix factorization (NMF) is another powerful algorithm, which we'll study in detail in Chapter 3, Mining the 20 Newsgroups Dataset with Clustering and Topic Modeling Algorithms.

At the end of the day, most dimensionality reduction algorithms are in the family of unsupervised learning as the target or label information (if available) is not used in data transformation.

Run Notebook

The Notebook opens in a new browser window. You can create a new notebook or open a local one. Check out the local folder `work/Chapter02` for several sample notebooks. Open and run `tSNE.ipynb` in the `work` folder.

You can open the Jupyter Notebook at `<host-ip>:8888/notebooks/work/Chapter02/tSNE.ipynb`

Summary

In this chapter, we acquired the fundamental concepts of NLP as an important subfield in machine learning, including tokenization, stemming and lemmatization, and PoS tagging. We also explored three powerful NLP packages and realized some common tasks using NLTK and spaCy. Then, we continued with the main project exploring newsgroups data. We started with extracting features with tokenization techniques and went through text preprocessing, stop words removal, and stemming and

lemmatization. We then performed dimensionality reduction and visualization with t-SNE and proved that count vectorization is a good representation for text data.

We had some fun mining the newsgroups data using dimensionality reduction as an unsupervised approach. Moving forward in the next chapter, we'll be continuing our unsupervised learning journey, specifically on topic modeling and clustering.