AI-LABS

OPEN-SOURCE TOOLS, TECHNOLOGIES, AND PLATFORMS

PROBLEMS/USE CASES
NATURAL LANGAUGE PROCESSING – AI LAB

Abstract

Instructions for a hackathon centred around problems using an open solution platform

AI TECH UK

20th April 2023

Table of Contents

Before you start

Think about what you hope to get out of this hackathon. Do you want to focus on just one technology and get the best model you possibly can, or do you want to understand all the technologies? Either is perfectly valid.
Remember learning is a journey not a destination, you can get a working model in a short period of time but what will you have learnt if you just stop there?

Overview of Problem Uses

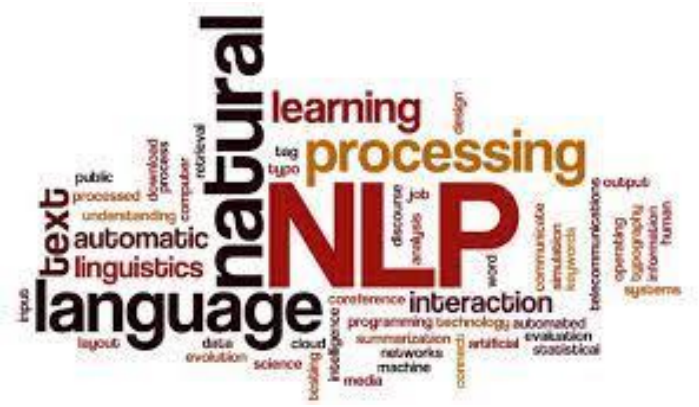
There are four areas of AI that can be explored as part of the AI Tech UK's – open-source Hackathon offering of coding in Python. These are:

01 Data Exploration
02 Machine Learning Algorithms and Applications
03 Computer Vision
04 Natural Language Processing

Each area will have a series of problems with associated notebooks and data files. For each area a structured approach is provided detailing the objective, dataset, algorithms and models and accompanying libraries, useful links, success criteria and in some cases what to do next (a question, a discussion, use of your own data, refer to a specific link/lab/task, or simply reading further resources of your choice).

04 Natural Language Processing

Introduction



Natural Language Processing is about how we interact with computers and human language to perform useful tasks. NLP is a branch of artificial intelligence (AI).
NLP involves techniques, trends and technologies deployed in a range of powerful business cases and applications. The global pandemic has brought the future forward by five years, due to AI adoption, investment, and latest NLP language models.

NLP is performed on text collections (corpora, plural of corpus)

- Tweets
- Facebook Posts
- Conversations
- Movie Reviews
- Documents, Books and many more

To use a range of NLP libraries, techniques, and datasets for specific use cases such as sentiment analysis, machine translation and chatbot development (in lab 3) and appreciate its limitation

Nuances of meaning make natural language understanding difficult

- Text's meaning can be influenced by context and reader's "world view"
- Text is highly contextual, ambiguous, and irregular

Range of NLP use cases:
1 NLP – language functions
2 NLP – document comparison
3 NLP – simple chatbot
4 NLP – spam detection

1 NLP – language functions

Objective

Approach

NLP involves a pipeline of common tasks. Depending on the specific task and goal various other tasks will be performed to the textual data and different approaches to the representation of the data, which will impact the analysis and results.

Dataset

Range of textual data stored in data structures and files such as 'RomeoAndJuliet.txt' and similar documents for comparison.

Libraries

- NLTK(Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries. NLTK has been called "a wonderful tool for teaching and working in, computational linguistics using Python," and "an amazing library to play with natural language."
- TextBlob - A library build on top of NLTK - for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.
- WordCloud - package helps us to know the frequency of a word in textual content using visualization
- Spacy - a free, open-source Python library that provides advanced capabilities to conduct natural language processing (NLP) on large volumes of text at high speed. It helps you build models and production applications that can underpin document analysis, chatbot capabilities, and all other forms of text analysis.
- Imageio is a Python library that provides an easy interface to read and write a wide range of image data, including animated images, volumetric data, and scientific formats.
- NumPy is a library for
- SkLearn library for predictive data analysis – built on NumPy, SciPy, and MatPlotlib – we will use cosine similarity and TDIFvectorise
- **WordNet** is an English word database created by Princeton University **TextBlob** uses NLTK's WordNet interface to look up word definitions, and get **synonyms** and **antonyms**

Algorithms and models

Following on from the approach – there are various NLP techniques and language models used. In this case simple one – but on the other end of the spectrum there are large language model used for training and testing data in real life applications.
Some examples include: tokenisation, string methods and comparisons; part of speech tagging to determine part of speech to determine meaning; extracting noun phrases, stop word elimination; Sentiment Analysis with TextBlob's Default Sentiment Analyzer; language detection and translation, inflection: pluralisation and singularization; speech checking and correction, normalisation: stemming and lemmatization; word frequencies, getting Definitions, Synonyms and Antonyms from WordNet; deleting top words, Ngram (sequence of n text terms); Visualizing Word Frequencies with Bar charts and Word Clouds; Getting the top 20 words; Named Entity Recognition (NER) with SpaCy to determine the text is about; similarity detection accuracy model (simple ~ 40MB /medium ~ 91 MB /large sized ~788MN) and cosine similarity algorithm

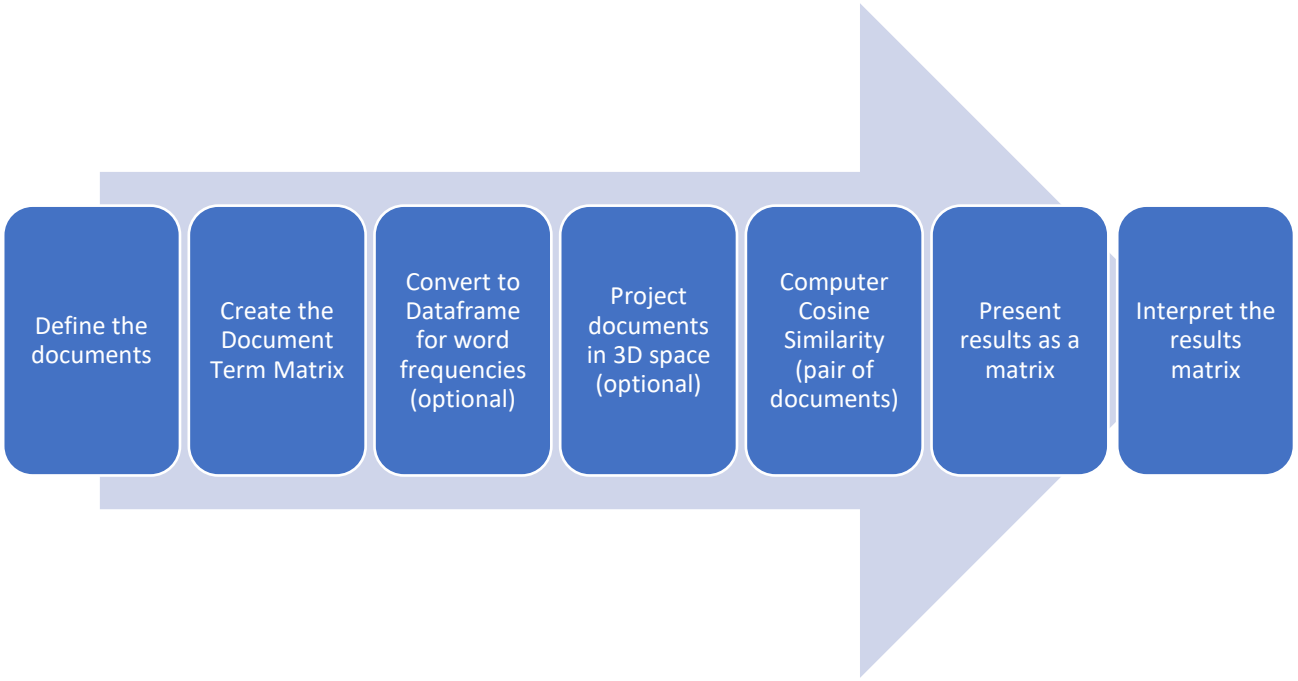
Success criteria.

- To be able to use a range of NLP libraries, techniques and datasets for specific use cases and explore data and make some decisions on the text
- To appreciate some limitations of accuracy, understanding, meaning and context

2 NLP – document comparison

Objective

A common NLP task is comparing documents for similarity and differences. For example, checking an assignment for Plagiarism. Here we will look at some simple documents but use a powerful algorithm used in most document comparisons.



Approach

A document set will have the documents read, defined, and saved in a dataframe for both presenting the word frequencies and to apply the cosine similarity to the pairwise document of the document set to create a matrix presenting the similarities. 1 denotes a 100% match of the same document. The higher the value under 1 the higher the similarity.

Dataset

Here the textual data is saved in strings. This can be replaced by text files that are read into a dictionary of strings. This is then vectorised as ion the lab.

Libraries

Use of Scikit Learn Library and feature extraction for the countVectorizer package, and use of metrics.pairwise and cosine similarly package. Pandas library is used to create a dataframe to compute the word frequencies.

Algorithms and models

Feature extraction of text is computed by the count vectorisation model which removes the stop words. This transformed to a sparse matrix to see the word frequencies of each document. Cosine similarity is a metric [ALGORITHMN] used to determine how similar the documents are irrespective of their size. 3 pairs of different documents are compared using the cosine similarity

Success criteria

1. To prepare documents for comparison and create the document term matrix
2. To create a dataframe to present word frequencies
3. To apply cosine similarity to the pairwise dataframe and then interpret the results matrix

Useful links

- Comparing Documents With Similarity Metrics - https://towardsdatascience.com/comparing-documents-with-similarity-metrics-e486bc678a7d
- Top 6 Ways To Implement Text Similarity In Python: NLTK, Scikit-learn, BERT, RoBERTa, FastText and PyTorch - https://spotintelligence.com/2022/12/19/text-similarity-python/

What to do next

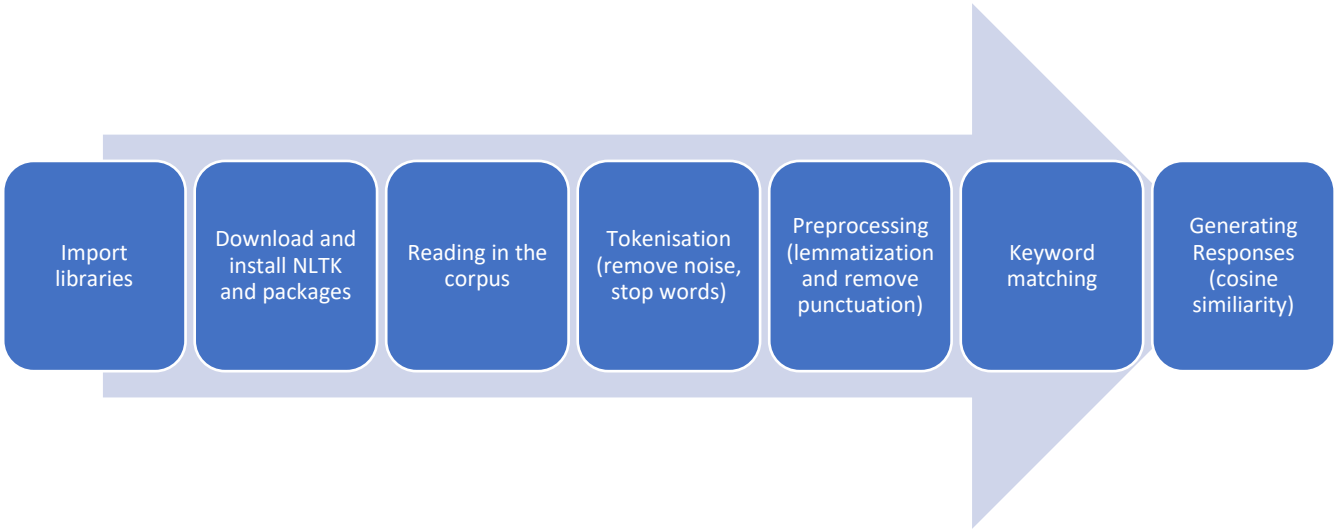
To change the code so that you have real text files for comparison.
Practice recreating the code from the useful links

3 NLP – simple chatbot

You are familiar with chatbots from Google Alexa, Sira and even live chat and more recently ChatGPT (very complex but effective). A simple chatbot is built from scratch using Python. This has no cognitive skills but a good way to get into NLP and get to know about chatbots and understand its limitations and where it needs improvement.

History of chatbots dates to 1966 when a computer program called ELIZA was invented by Weizenbaum. It imitated the language of a psychotherapist from only 200 lines of code. You can still converse with it here: Eliza.

Objective



Approach

Creating a chatbot is based on a textual domain data and questions are asked of the We define a function response which searches the user's utterance for one or more known keywords and returns one of several possible responses. If it doesn't find the input matching any of the keywords, it returns a response:" I am sorry! I don't understand you" chatbot with a natural language generative response. The questions must be structured correctly for adequate keyword matching responses. To generate a response from our bot for input questions, the concept of document similarity will be used. We define a function response which searches the user's utterance for one or more known keywords and returns one of several possible responses. If it doesn't find the input matching any of the keywords, it returns a response:" I am sorry! I don't understand you". You can continue asking question and if you want to exit, type Bye!"

Dataset

A 'chatbot.txt' file is read. The text is an extract from a Wikipedia page on chatbots.

Libraries

As stated in Lab 1 – NLP uses range of standard libraries such as the NLTK, and its packages. Here we invoke WordNet Lemmitizer for popular words. Here it invokes io, string, NumPy, random, scikit-learn feature extraction for the TfidVectoriser package and scikit-learn metric pairwise for the cosine similarity package.

Algorithms and models

Common NLP tasks of tokenisation and pre-processing with the WordNet Lemmatizer are used create the lemma or base form of the word – so it is easy to match words. Keyword matching is limited to the dictionary of greeting-inputs and greeting responses – but can be extended for more intents.
The main work is the generating of the response using two different methods bag-of-words (words considered but not the order of them) and Term Frequency-Inverse Document Frequency, or TF-IDF approach to rescale the frequency words by how often they appear in all documents. TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

In any vectorisation work, linear algebra and matrix manipulation is performed. To generate the response the concept of document similarity is used. We define a function response which searches the user's utterance for one or more known keywords and returns one of several possible responses. If it doesn't find the input matching any of the keywords, it returns a response:" I am sorry! I don't understand you"

Success criteria.

1. To be able to install NLTK, necessary libraries and read in the corpus into a raw string for questioning
2. To be able to tokenise and pre-process the raw string into lemma tokens
3. To be able to apply cosine similarity to generate a response for the question posed using simple keyword matching

Useful links

- How To Build Your Own Chatbot Using Deep Learning - https://towardsdatascience.com/how-to-build-your-own-chatbot-using-deep-learning-bb41f970e281
- 14 most popular platforms to build a chatbot https://marutitech.com/14-powerful-chatbot-platforms/
- 26 Best Real Life Chatbot Examples – well-known brands) https://www.tidio.com/blog/chatbot-examples/

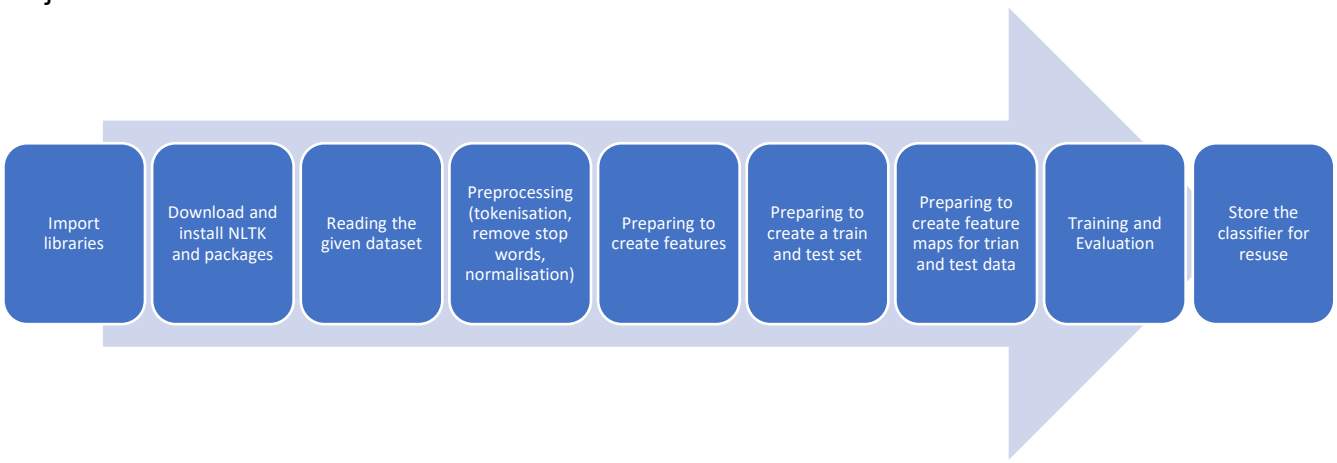
What to do next

To review the link on how to build a chatbot using Deep Learning. What is the difference between our solution and this one. What more needs to be done? What are the challenges of development?

4 NLP – Spam Detection (Email)

This NLP use case detects spam found in e-mails and placed it into a spam folder as opposed to 'ham' folder with acceptable e-mails.

Objective



Approach

This SPAM-HAM detector for e-mails is a recommendation system, based on probabilities of prior knowledge and likelihood of what is a spam e-mail or a ham e-mail. The fundamental Naive Bayes assumption is that each feature makes an independent equal contribution to the outcome. A classifier is created that is evaluated using performance measures of accuracy, precision, and recall.

Dataset

A 'spam message.txt' dataset is used for both training and test purposes. A classifier model is generated as a pickle file for reuse.

Libraries

A range of NLTK packages are downloaded and installed to consider the sentence tokenising, stopword removal, Wordnet knowledge, normalisation of the text via the PorterStemmer, and WordNet Lemmatizer. Also, NLTK will provide methods to classify and apply features to both the training and test dataset. Other libraries include random (to select the train and test messages) and Pandas to read the file into a dataframe and then to create a dataset.

Algorithms and models

The spamClassifer is trained using the Naive Bayes algorithm. This is one of the crucial algorithms in supervised machine learning that helps with classification problems. It is derived from Bayes' probability theory and is used for text classification, where you train high-dimensional datasets. Here we create a binary classifier to make predictions.

Success criteria

1. To be able to read and pre-process the messages ready for creating features
2. To be able to create and train and test set
3. To be able to create features maps for the train and test data

4. To be able to train and evaluate a SpamClassifier

Useful links

- Naives Bayes - https://scikit-learn.org/stable/modules/naive_bayes.html
- Other examples - https://www.geeksforgeeks.org/naive-bayes-classifiers/
- Naïve Bayes Classification- https://www.datacamp.com/tutorial/naive-bayes-scikit-learn
- Classification and its use cases in Machine Learning
    - Recreate either one of the labs from the useful links

What to do next for NLP

Additional mostly free and open-source NLP libraries and APIs:

- Gensim—Similarity detection and topic modelling
- Google Cloud Natural Language API—Cloud-based API for NLP tasks such as named entity recognition, sentiment analysis, parts-of-speech analysis, and visualization, determining content categories and more
- Microsoft Linguistic Analysis API
- Bing sentiment analysis—Microsoft's Bing search engine now uses sentiment in its search results
- PyTorch NLP—Deep learning library for NLP
- Stanford CoreNLP—A Java NLP library, which also provides a Python wrapper. Includes corefererence resolution, which finds all references to the same thing.
- Apache OpenNLP—Another Java-based NLP library for common tasks, including coreference resolution. Python wrappers are available.
- PyNLPl (pineapple)—Python NLP library provides a range of NLP capabilities
- KoNLPy—Korean language NLP
- Latest BERT language models

Machine Learning and Deep Learning Natural Language Applications

- Answering natural language questions—For example, our publisher Pearson Education, has a partnership with IBM Watson that uses Watson as a virtual tutor. Students ask Watson natural language questions and get answers.
- Summarizing documents—analysing documents and producing short summaries (abstracts) that can, for example, be included with search results and can help you decide what to read.
- Speech synthesis (speech-to-text), speech recognition (text-to-speech), inter-language text-to-text translation.
- Collaborative filtering—used to implement recommender systems ("if you liked this movie, you might also like…").
- Text classification—e.g., classifying news articles by categories, such as world news, national news, local news, sports, business, entertainment, etc.
- Topic modelling—finding the topics discussed in documents.
- Sarcasm detection—often used with sentiment analysis.
- Closed captioning—automatically adding text captions to video.
- Speech to sign language and vice versa—to enable a conversation with a hearing-impaired person.
- Lip reader technology—for people who can't speak, convert lip movement to text or speech to enable conversation.