

Natural Language Processing

Micha Birklbauer



Introduction to NLP

- Two blocks:
 - Basics in NLP
 - Machine Learning in NLP
- Aim:
 - At the end of these two blocks you should know how to efficiently deal with textual data, understand and know how to apply common concepts & be able to solve common NLP tasks
- What to not expect:
 - This is not a machine learning course, we will use ML methods but they will NOT be explained in-depth
- Grading for these two blocks:
 - Attendance
 - Two (small) homeworks until the end of the semester

Contents

- Preprocessing
- Feature representations
- Popular NLP packages
 - spaCy
 - NLTK
 - Gensim
- Clustering
- Classification

Preprocessing

- Data cleaning
 - Probably the most important step in NLP
 - Ideally your data quality is already good and no data cleaning is needed, however in reality that is rarely the case
 - In some cases a simple regex pattern suffices, in other cases it's a really cumbersome process
 - When doing machine learning, data cleaning becomes increasingly more important the less data you have and the simpler your model is
 - Generally „Crap in = Crap out“

Preprocessing

- Tokenization:
 - Process of breaking a stream of text up into (key)words, symbols, or other meaningful elements called tokens
 - Tokens are small units of (meaningful) text, such that a comparison between a token in the query and a token in a document can take place
 - Usually tokenization based on whitespace, but we will see that that doesn't always work very well
 - Removing punctuation may or may not be useful depending on further transformations, good for TF-IDF but may lose information for 2Vec models
- The quick brown fox jumps over the lazy dog = the, quick, brown, fox, ...

Preprocessing

- Stopword removal:
 - Stopwords are words that are too frequent in a language or document collection and have very little semantic meaning
 - Removing them reduces the feature space
- The quick brown fox jumps over the lazy dog = quick, brown, fox, ...

Preprocessing

- Stemming:
 - Different word forms may have similar meaning e.g. jumps, jumping
 - Therefore, create a “standard” representation for them by removing the endings
- The quick brown fox jumps over the lazy dog = quick, brown, fox, jump, ...

Feature Representations

- TF-IDF:
 - Term Frequency tf = number of times term t occurs in document d
 - Document Frequency df = number of documents a term t occurs in
 - Inverse Document Frequency $idf = \log_2(\text{total number of documents}/df)$
 - $TF\text{-}IDF = tf * idf$
- Word2Vec, Sentence2Vec, Doc2Vec, ...
 - Each word, sentence, document, etc is represented as an n-dimensional real vector that encapsulates the semantic and syntactic properties of the object
 - These vectors can in turn be used for finding similar words, clustering or classification

Word2Vec

- We can get these vectors by training an artificial neural network on a large corpus to predict the semantic and syntactic meaning of words
- TensorFlow has a nice tutorial for that:
<https://www.tensorflow.org/tutorials/text/word2vec>
- But usually we use pre-trained models for that task since there are plenty good ones out there already e.g. [fastText](#) by Facebook AI Research
- We will use the pre-trained models by spaCy

Popular NLP Packages

spaCy, NLTK, Gensim



spaCy

NLTK

- <https://spacy.io/>
- spaCy is a natural language processing library that comes with many built-in features that solve core linguistic tasks like tokenization, lemmatization, POS-tagging and NER

Exercise 1:

https://github.com/michabirklbauer/hgb_dse_text_mining/blob/master/spaCy.ipynb

https://colab.research.google.com/github/michabirklbauer/hgb_dse_text_mining/blob/master/spaCy.ipynb

NLTK

- <https://www.nltk.org/>
- NLTK - short for Natural Language Toolkit - is a leading platform for building Python programs to work with human language data
- It provides easy-to-use tokenization, stemming, and much more
- I mostly use NLTK for preprocessing tasks because it is more light-weight and straightforward than spaCy in my opinion

Gensim

- <https://radimrehurek.com/gensim/>
- Gensim “Topic Modelling for Humans” is another NLP library in Python
- Gensim offers easy-to-use implementations for TF-IDF and text/document queries

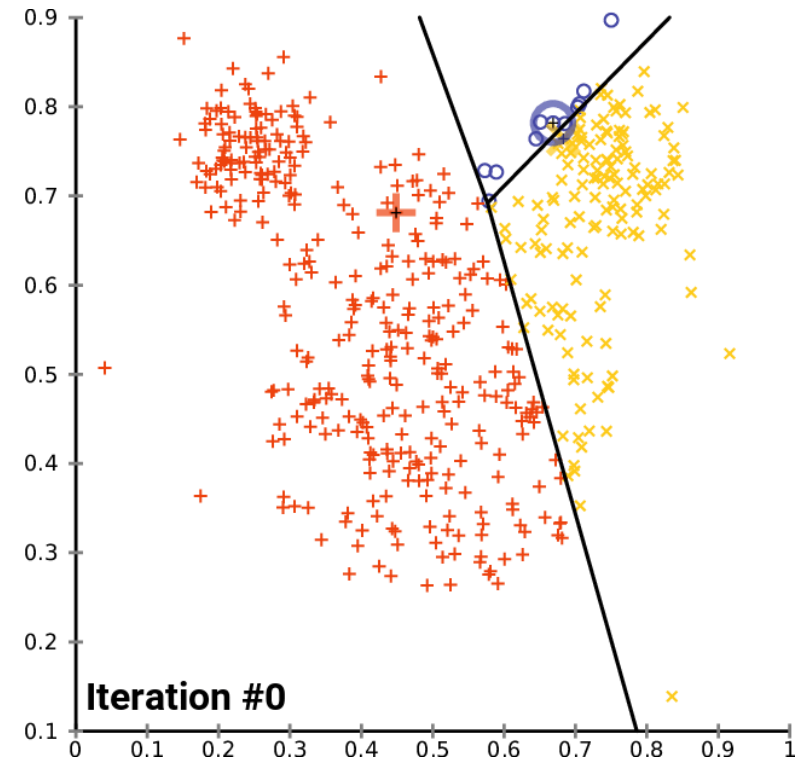
Exercise 2:

https://github.com/michabirklbauer/hgb_dse_text_mining/blob/master/NLTK_Gensim.ipynb

https://colab.research.google.com/github/michabirklbauer/hgb_dse_text_mining/blob/master/NLTK_Gensim.ipynb

Clustering

kMeans



kMeans Clustering

1. Initially select k random datapoints (documents) as cluster centroids
2. Assign each document in the corpus to the nearest centroid
3. Calculate new cluster centroids
4. Repeat 2. and 3. until convergence or maximum iteration (or other stopping criterion) is met

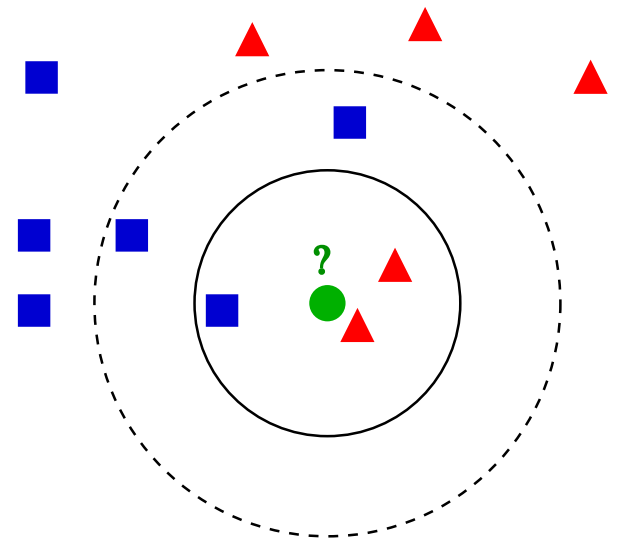
Exercise 3:

https://github.com/michabirklbauer/hgb_dse_text_mining/blob/master/Features_Clustering.ipynb

https://colab.research.google.com/github/michabirklbauer/hgb_dse_text_mining/blob/master/Features_Clustering.ipynb

Classification

kNN, naiveBayes



kNN

- Predict the class that is most frequent among the k closest datapoints (documents)
- (That's it...)

naiveBayes

- The probability of class C given feature F is according to Bayes Theorem given as:

$$p(C | F) = \frac{p(F | C)p(C)}{p(F)}$$

- We want to predict the Class C_{MAP} that has the highest probability given F , we call that the class with the maximum a-posteriori probability

$$C_{MAP} = \arg \max_{C \in H} p(C | F)$$

$$C_{MAP} = \arg \max_{C \in H} \frac{p(F | C)p(C)}{p(F)}$$

naiveBayes

- For a finite set of features F_1, \dots, F_n we can write this down as:

$$C_{MAP} = \arg \max_{C \in H} \frac{p(C)p(F_1, \dots, F_n | C)}{p(F_1, \dots, F_n)}$$

- The denominator is not dependent on C and therefore constant and we can eliminate it:

$$C_{MAP} \sim \arg \max_{C \in H} p(C)p(F_1, \dots, F_n | C)$$

- If we can assume that all features F_1, \dots, F_n are independent we can decompose the last part into:

$$p(C, F_1, \dots, F_n) = p(C)p(F_1 | C)p(F_2 | C) \dots p(F_n | C) = p(C) \prod_i p(F_i | C)$$

naiveBayes

- Our final classifier is therefore:

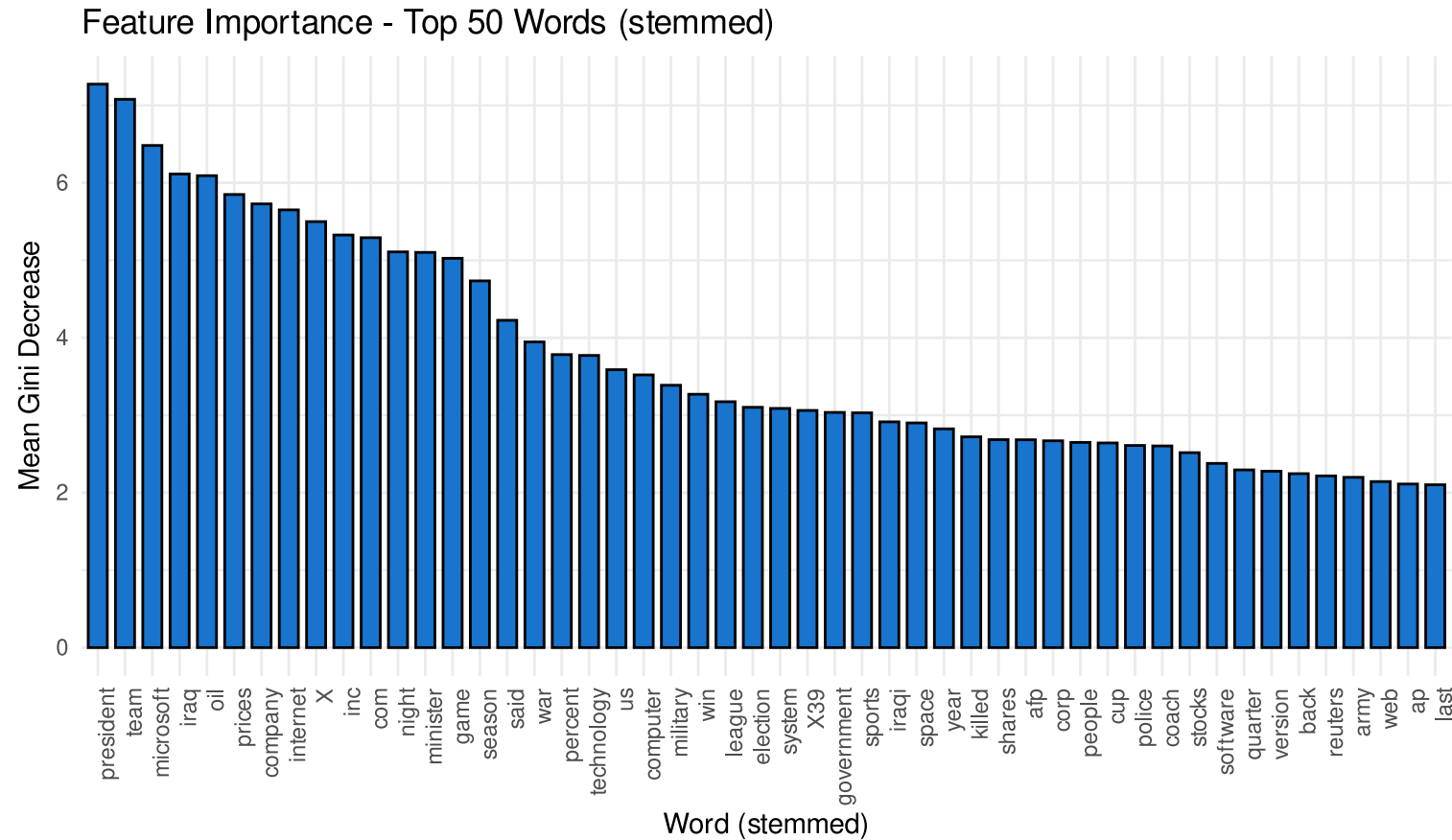
$$C_{NB} = \arg \max_{C_j \in H} p(C_j) \prod_i p(F_i | C_j)$$

Exercise 4:

https://github.com/michabirklbauer/hgb_dse_text_mining/blob/master/Classification.ipynb

https://colab.research.google.com/github/michabirklbauer/hgb_dse_text_mining/blob/master/Classification.ipynb

Feature Importance



Homework

Text Mining (WS23)

Exercise 1

Exercise 1:

Select an english or german wikipedia article of your choice and analyze it with spaCy analogous to what we did in the lecture (and you may reuse the code from the lecture).

Tasks:

- You may directly load the article with python or just copy-and-paste a paragraph into your jupyter notebook.
- Select a spaCy model appropriate for the chosen language.
- You should select a random sentence and pre-process it as the following:
 - Tokenize it and print the tokens (1 pt.)
 - Lemmatize it and print the lemmas (1 pt.)
 - Carry out POS-tagging, print the tags and visualize them as a dependency plot (2 pt.)
 - Carry out NER, print the entities and visualize them with displacy (2 pt.)

You are expected to hand in a jupyter notebook with at least minimal comments in .html format!

Hand-in is due 31st of January 2024, 23:59;

2. Block - Outlook:

- **Artificial Neural Networks**
- **Generative Models**
- **Hands-on: Sentiment Analysis**
- **Hands-on: Show-and-Tell models**

Feedback & Questions:

micha.birklbauer@fh-hagenberg.at

