# Lab 1 (Information Retrieval: Google Play Store)

## Preprocessing

* removing stop words
* replacing each remaining token with its lemma (base form)
* discarding all lemmas that contain non-alphabetical characters

**Tip:** To speed up the preprocessing, you can disable loading those spaCy components that you do not need, such as the parser, and named entity recognizer.

**RQ 1.1:** Why do we remove common stop words and lemmatise the text? Can you give an example of a scenario where, in addition to common stopwords, there are also *domain-specific* or *application-specific* stop words?

* Discard words from the text that do not contain valuable information
* Oftentimes, stopwords occur frequently in a language
* Lemmatization is used to combine different forms of the same word into one unified term (the so-called lemma).
* An example for a domain specific stop word is "student" in documents in the area of education.
  + Riley’s comment: Good example of a domain-specific stop word, though do be careful when removing nouns. Since we're just working with creating representations of documents in this lab, removing a word that appears roughly uniformly in every document works just fine. However, as you'll see in most of the remaining labs, nouns can play a big role in other types of analyses you might do.

## Vectorizing

* Tf-idf (term frequency - inverse document frequency)
* Finding terms with low/high tf-idf:
  + Terms with low idf like game and play are expected to appear in most of the documents in the collection.
  + Some of the terms with high idf are non-english words and therefore don't occur in a lot of different documents and some are non-common words. However, the last 10 terms don't seem to be in the correct alphabetical order. Our assumption is that this is due to some encoding issues.
    - Riley’s comment: np.argsort doesn't seem to play as nicely with the inherent ordering of the vocabulary, for whatever reason. It might be the case that np.argsort doesn't guarantee an ordering of indices when there are ties.
* Extract keywords from a text:
  + Pick k terms with the highest tf-idf value (they are specific for this text)

**RQ 1.2:** In Problem 2, what do the dimensions of the matrix X correspond to? This matrix gives rise to different types of *representations*. Explain what these representations are. What information from the data is preserved, what information is lost in these representations?

* The number of rows corresponds to the number of documents and the number of columns to the size of our vocabulary. The size of the vocabulary is the number of unique preprocessed words.
  + Entry is the tf-idf value of the corresponding word in that document
* Term-frequency:
  + preserves information about the term frequency, but not about how often the term occurs in the whole collection of documents
* Binary representation:
  + we don't preserve any information about how often a term occurs in a document
* Term frequency - inverse document frequency
  + relates the term frequency to the inverse document frequency
  + we preserve information about how often a term occurs in a document compared to how often it occurs in the overall collection of documents
  + we lose information about the absolute frequency of a term

Riley’s comment: One key thing to note about all of the vector-based document representations you've seen is that none of them capture word order, which in turn means that they lose a lot of linguistic information.

**RQ 1.3:** What does it mean that a term has a high/low idf value? Based on this, how can we use idf to automatically identify stop words? Why do you think is idf not used as a term weighting on its own, but always in connection with term frequency (tf–idf)?

* idf value is high
  + term occurs only in a few documents compared to the total number of documents
* idf value is low
  + the term occurs in most of the documents
  + Based on the assumption that stop words occur very frequently and therefore in approximately all documents, terms with low idf values can be assumed to be stopwords.
* The idf value is usually never used alone because values that for example only occur once in one document would have the same value as a term that occurs 100 times in one document and nowhere else. In this case, it is obvious that the term that occurs 100 times should have a higher weight than the term that occurs only once.

## Retrieving

* + kNN (return the 10 most relevant app descriptions for some query)

# 

# Lab 2 (Text Classification: Parliament Speeches)

## Naive Bayes Classifier

* Vectorize speeches and fit NB in pipeline
* Different dummies: random sampling respecting training class distribution, predict the most often one for all observation
* **Undersampling**: randomly remove samples from over-represented classes until all classes are represented with the same number of samples

**RQ 2.1:** Summarise the results of your experiments for Problem 2. Are your results ‘good’ or ‘bad’? How do you determine that?

* seems like the model tends to predict "Socialdemokraterna" and "Moderaterna" quite often as our classes are quite unbalanced and both of them have a lot more observations than the other ones
* without a baseline model we can’t compare it
* precision and recall are quite unbalanced within each class

**RQ 2.2:** Summarise the results of your experiments for Problem 4. Would you think that your results are typical even for other classification tasks? How would *oversampling* have looked like for this task? When would you use undersampling, when oversampling?

* accuray for the model is lower than for the one in Problem 2
  + values for precision and recall are more balanced than before as we have equally sized classes.
    - This observation would also hold for other classification tasks
* **Oversampling:** create a bigger dataset resampling from all parties except the Socialdemocrates until we have the same amount of observations for all parties
  + In a situation, where the smallest class has a very small amount of observations this is probably the more appropriate option.
  + If we used undersampling in that situation, we would lose a lot of potentially valuable information as we would drop a lot of data points when doing undersampling.
  + In a situation where even your smallest class has a large amount of data points you can do undersampling.

Riley’s comment: The usual trade-off is between data set size and training time; though one can also analyze the classifier for properties such as recency bias, where it's more likely to classify examples correctly only if they're more similar to those it has recently learned from. Then it might help to ensure that your data set is completely balanced, even if that means oversampling.

## Grid search

* Set-of-words (binary) model vs. the default bag-of-words model
  + Setting to 1 or number of occurrences
* Bigrams vs. unigrams
  + N-gram is an arrangement of n words, so two word combination vs. one word combination
* Additive smoothing in Naive Bayes (Parameter value of 1 vs. 0.1)
* For the parameters dictionary used for the pipeline: vectorizer\_\_binary where vectorizer is the step in the pipeline and binary is the argument which is part of the grid search

**RQ 2.3:** Which model performed best in your experiments for Problem 6? Why is it important to do a hyperparameter search before drawing conclusions about the performance of a model? Why is it often not done, anyway? Why should you never tune hyperparameters on the test set?

* best performance: α=0.1 with a bag-of-words and bigram vectorizer
* performance could be a lot worse with the "wrong" hyperparameters so you can't really draw conclusions about the performance before the hyperparameter tuning
* exhaustive grid search is however very computationally expensive and the model performance might not improve a lot which is why it is not often done
* the hyperparameters should never be tuned on the test set because then the model would be optimised on that set
  + expected error for the new data would be higher compared to the test data.

Riley’s comment: Always make sure that you pair a claim about "best performance" with the metric that you're using to compare. This is especially true since you spoke a lot about tradeoffs between accuracy and precision/recall, yet the cross validation library by default only compares accuracy.

# Lab 3a (Text clustering: Project reviews and Topic modelling: State of the Union)

## K-means clustering

* Tf-idf vectorizer and then k-means clustering with k=3

**Tip:** Training k-means models will take some time. To speed things up, you can use the n\_init parameter to control the number of times that the clustering is re-computed with different initial values. The default value for this parameter is 10; here and in the rest of this lab, you may want to set this to a lower value.

* Summaries for the clusters:
  + Cluster 1: Camera equipment
  + Cluster 2: Media
  + Cluster 3: Software (+very general terms)

## Rand Index

* Can be computed if you have gold-standard labels for at least a part of the data
* Approach in the Lab:
  + View as binary classifier on (unordered) pairs of documents
    - predicts ‘positive’ if and only if the two documents belong to the same cluster.
    - accuracy of this classifier relative to a reference in which a document pair belongs to the ‘positive’ class if and only if the two documents in the pair have the same gold-standard class label

Our values (not a huge difference):

Rand index for k = 1: 0.166

Rand index for k = 3: 0.673

Rand index for k = 5: 0.743

Rand index for k = 7: 0.756

Riley’s comment: Your scores here are a little bit higher than I was expecting; this is primarily due to the fact that you trained the k-means models only on the first 500 reviews instead of the full data (this is fairly unclear in the problem description). Normally, you see values closer to .16, .44, .67, and .76 for this problem, but k=3 is fairly unstable.

**RQ 3.1:** Based on your experiments in Problem 2 and Problem 3, what is the relation between the quality of a clustering and the number of clusters? What happens when the number of clusters is too low, or too high? For this particular data set, what would a ‘good’ number of clusters be?

* Observation from this lab: increasing number of clusters -> rand index increases
  + would expect that the rand index starts to decrease again at some point when increasing the number of clusters
* number of clusters too low: a lot of points will be clustered to the same cluster even though they might be a lot of different classes.
* number of clusters is too high: a lot of points will be in different clusters even though there might not be that many different classes
* 6 different classes in the categories, so 6 would be a logical choice for the number of classes

Riley’s comment: The main point to keep in mind with the Rand index is that it's best suited for relative comparisons, such as between the k-means clustering at various values of k. Since you do have a gold standard here, you can make claims about the quality of the clustering, but that is not generally the case. The score that you get depends, unfortunately, both on a bit of randomness in the clustering and on the distribution of labels in your gold data (which, here, was fairly even). For instance, k=1 would get a pretty good score even if you had 20 topics if one of those topics accounted for 90% of samples.

## Topic Modeling

Hier habe ich jetzt nicht wirklich viel mehr aufgeschrieben, weil das Thema irgendwie nie so wichtig wirkte und ich müde war :D Schaue ich mir aber ggf. nochmal kurz an

**RQ 3.2:** Explain why it is important to monitor an LDA model for convergence and not simply use, say, 1000 passes. How is the log likelihood used in this context? Were the topics from the multi-pass model ‘better’ than the topics from the 1-pass model?

* It takes a long time to train the model with 1000 passes even though the likelihood does not increase noticeably after some iterations.
  + lose a lot of time but not gain a lot of additional performance when adding more passes
* The model trained with 50 passes seems to produce topic summaries where the different topics are more contained within one field of politics compared to the 1-pass model

Riley’s comment: You don't clearly discuss how the log likelihood is used in this case; you more just describe it in abstract terms of performance. However, performance is usually relative to a metric, and there's not any gold standard here to derive such a metric from. Otherwise, still a good response!

* Ich verstehe seinen Kommentar nicht. Falls du ihn verstehst, kannst du es mir gerne erklären.

## Comparison

**RQ 3.3:** What are the differences between k-means and LDA? When would you use one, when the other?

* K-means does hard clusters: each of the data points is assigned to one distinctive cluster.
  + should be used when the task requires to produce hard clusters
  + Example: If you do sentiment analysis and want to actually sort all data points into certain sentiments (for example positive/negative)
* LDA produces soft clusters: each data point is assigned to all of the clusters up to a certain degree
  + should be used when it is beneficial to know the degree up to which a data point belongs to a certain cluster
  + Example: If you do sentiment analysis and want to know for each data point up to what degree this data point is positive

Riley’s comment: Good! Though I would have liked for you to also describe what it means for a task to require producing a hard cluster (what kind of task, what kind of labels, etc.).

* Examples added after this comment. Do they make sense in your opinion?

# 

# Lab 4 (Word embeddings)

* most\_similar uses cosine similarity
* Part of that: synonymy, antonymyn, hyperonoymy (more specific/less specific)

**RQ 4.1:** In Problem 3, you manipulated word vectors using addition and subtraction, getting intermediate vectors which are still valid embeddings. Consider the difference vector *Stockholm*−*Sweden*. What does that vector intuitively represent? What words do you think it should be most similar to?

The vector basically represents everything that is left if you take away Sweden from Stockholm. The word embedding that is closest to the resulting vector might be the embedding of "capital" or "city".

Riley’s comment: While true in a very literal sense, "everything that is left" might be better given a more descriptive label. Most often I talk about this representing the "relation between the two words", though to be fair it often doesn't represent anything in particular.

**RQ 4.2:** Manually engineering features is a fairly time-consuming task, but as shown in Problem 5, can result in systems which are on par with embedding-based systems. Conversely, embedding-based systems avoid manual feature engineering, but often require significantly more computational resources. Looking at your results from Problem 7, are embedding-based systems worth the extra resources in the context of this task?

This depends on the task at hand. If the goal is to achieve the highest possible accuracy, the results of this lab show that using embedding-based systems can improve the accuracy. However, in most real-world cases computational resources are limited and a trade-off between accuracy and computational resources needs to be made. A general statement about one optimal model can therefore not be made.

Riley’s comment: Good! Though I would have liked to see you point to a few more details from your results to the problems in this lab, such as what the difference in accuracy was, what the runtime of the two was, and so on. Those are both points to keep in mind when you update the lab (your scores and their differences will change), but you do not need to redo this question.

**RQ 4.3:** Throughout the lab, you have been using pre-trained word vectors from spaCy. In Problem 7, you used them to compute the *input* to a neural network. Another common pattern is to use them to initialize an *embedding layer* in a neural network. (Have a look at [this article](https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/) if you are unfamiliar with that pattern.) Explain the difference between the two usage patterns. What advantages/disadvantages do they have?

When using the pre-trained word vectors as input to the network, our hypothesis is that the model generalises better compared to a model where the pre-trained embeddings are used only to initialise an embedding-layer. During training, the embedding-layer can then be adapted to the task at hand which makes it more flexible, but also prone to overfitting.

# Lab 5 (Information Extraction)

**RQ 5.1:** In Problem 3, you did an error analysis on the task of recognizing text spans mentioning named entities. Summarize your results. Pick one type of error that you observed. How could you improve the model’s performance on this type of error? What resources (such as domain knowledge, data, compute) would you need to implement this improvement?

We improved the f1-score by removing certain named entities like dates, times, cardinals, ordinals and persons if the person name contained a number. This increased the f1-score by around 15%. However, one has to keep in mind that we decided on which entities to remove based on the errors our model made on the same data we used to calculate the f1-score. One type of error was that British town names were not correctly detected sometimes. This could be improved by incorporating domain knowledge about all town names in Great Britain.

Riley’s comment: Actually, you were supposed to "train" your preprocessing from that step by basing your choices on errors within the training data... I missed that on my first read-through, but I won't ask you to correct that, since you showed that you know the implications of doing it that way. Otherwise, good that you noticed the trend with British town names and used that as an example of where domain-specific information could be used!

**RQ 5.2:** Thinking back about Problem 6, explain what the word *context* refers to in the task addressed there, and how context can help to disambiguate between different entities. Suggest other types of context that you could use for disambiguation.

In problem 6, context was defined as the five tokens to the left and the five tokens to the right. However, it could also be defined as all words that occur in the same sentence or all words that occur in the same document as the term of interest.

**RQ 5.3:** One type of entity mentions that we did not cover explicitly in this lab are pronouns. As an example, consider the sentence pair *Ruth Bader Ginsburg was an American jurist*. *She served as an associate justice of the Supreme Court from 1993 until her death in 2020*. What facts would you want to extract from this sentence pair? How do pronouns make fact extraction hard?

Facts we would want to extract are that she was an American jurist, that she was part of the supreme court from 1993 to 2020 and that she died in 2020. It might be difficult to extract all of these facts because her name is not explicitly mentioned in the second sentence. Therefore, the system has to detect what the pronoun 'she' refers to. This task seems easy for humans, but is not trivial for computers.