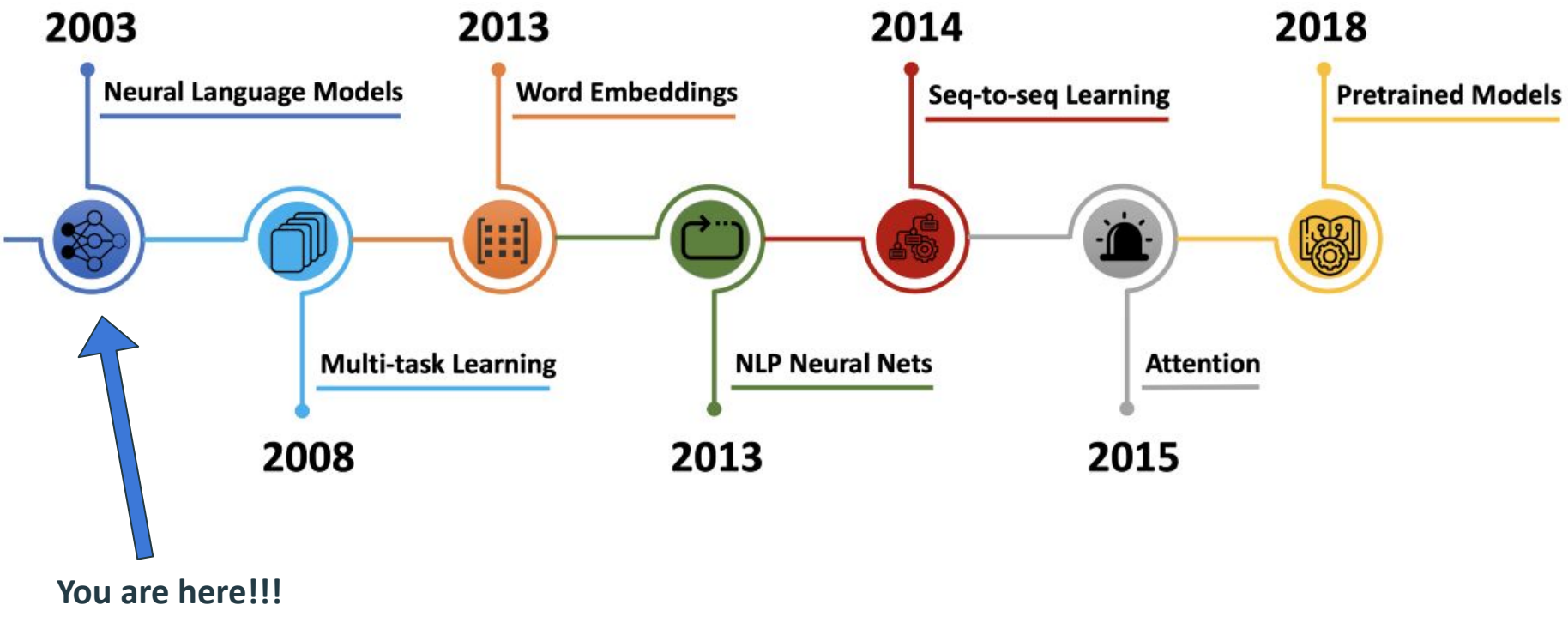
Practical Intro to NLP Workshop

Matei Bejan

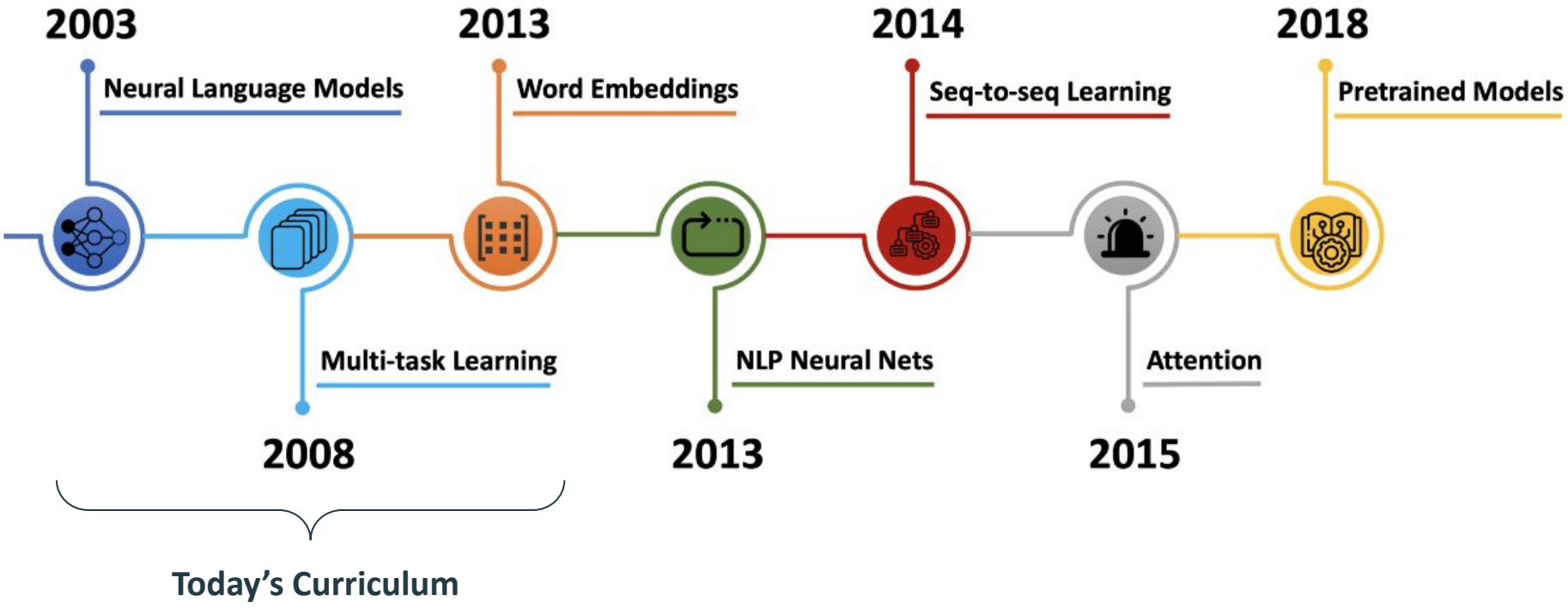
Summary:

1. A Brief Overview
2. Language Models
3. Preparing Your Data
4. Core NLP Libraries
    a. SpaCy
    b. Gensim
5. Evaluation Metrics and Methodology

A Brief Timelime of NLP



A Brief Timelime of NLP



A Note On Terminology

Token = An instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing: *a word is a token in a sentence*, and a sentence is a token in a paragraph etc.

Document = Some text.

Corpus = Collection of all documents in your dataset. Can span multiple languages.

Sample = Subset of a dataset. Typically (but not limited to) one data point.

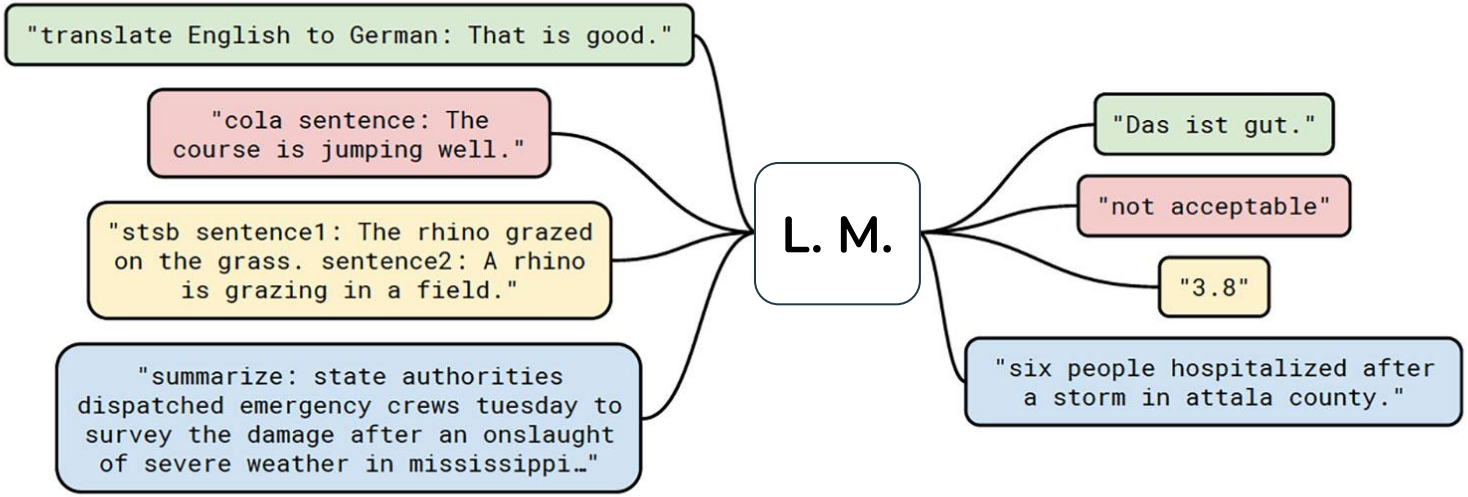
Batch = A set of samples.

Distribution = A function that shows the possible values a variable can take and how often they occur.

Model = Algorithm. The term is employed in the context of computational learning, either supervised, unsupervised or semi-supervised.

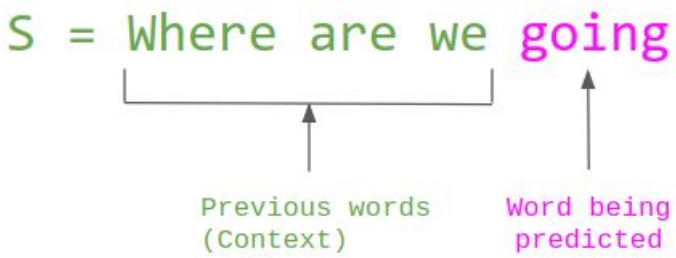
Language Models

Language Models



Language Models

Fancy Maths Formulation: A LM is a probability distribution over sets of words or word sequences.

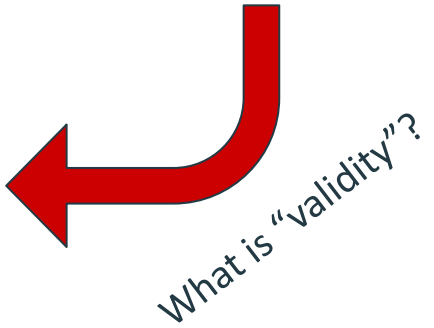


$$P(S) = P(\text{Where}) \times P(\text{are} \mid \text{Where}) \times P(\text{we} \mid \text{Where are}) \times P(\text{going} \mid \text{Where are we})$$

Language Models

Layman's Terms: A LM is an algorithm that gives the probability of a word to be **valid**.

Validity is a measure of how well has the model learned to resemble human writing (or speech) patterns, by looking at data.



P.S.: "Layman's terms" means a simple language that anyone can understand.
Source: https://www.merriam-webster.com/dictionary/layman%27s%20terms.

Language Models

1. **Statistical Language Models:** These models use traditional statistical techniques like N-grams, Hidden Markov Models (HMM) and certain linguistic rules to learn the probability distribution of words.

2. **Neural Language Models:** Recurrent Neural Networks, LSTMs, Attention.

Part of Speech Tagging

POS tagging is the process through which we categorize the tokens in a corpus in correspondence with a part of speech (noun, verb, attribute etc.).

This way we can capture syntactical information, which allows us overcome the shortcomings of some methods we will further discuss.

Lexical Term	Tag	Example
Noun	NN	Paris, France, Someone, Kurtis
Verb	VB	work, train, learn, run, skip
Determiner	DT	the, a
...	...	

Why	not	tell	someone	?
WRB	**RB**	**VB**	**NN**	**.**

Part of Speech Tagging

Several Ways to Achieve POS Tagging

- Lexical-Based Methods
  - Frequentist approach.
- Rule-Based Methods
  - Word rules, such as prefix/suffix occurrence.
- Probabilistic Methods
  - CRFs, HMMs.
- Deep Learning Methods
  - RNNs.

Several Useful Applications

- Named Entity Recognition.
  - Coming Soon.
- Coreference Resolution.
- Speech Recognition.
- *… and many more.*

Named Entity Recognition

Question: What is a named entity?

Answer: An object which has a unique identification and can be denoted with a proper name. It can be a place, person, organization, time, object or geographic entity.

Question: What is NER?

Answer: The process in NLP which deals with identifying and classifying named entities.

Named Entity Recognition

Why NER?

We can extract key information to understand the text, or merely use it to further infer useful information for our models.

Application of NER

- Recommendation Engines.
- Chatbots.
- Ticket Classification.
  - Parsing metadata.
- Various Types of Analytics.
  - Content, customer, etc.

N-grams

Rigorous definition: An N-gram language model predicts the probability of a given word within any sequence of words in a corpus.

With a good N-gram model, we can predict the probability of seeing a certain words after observing N - 1 previous words.

N-grams

Layman's Terms: Contiguous sequence of **n** tokens. E.g.:

- **unigrams (n=1):** *dog, that, barks, does, not, bite.*
- **bigrams (n=2):** *dog that, that barks, barks does, does not, not bite.*
- **trigrams (n=3):** *dog that barks, that barks does, barks does not, does not bite.*
- *… and so forth and so on.*

Practical Tip: Split your corpus into N-grams of your choice and calculate the co-occurrence frequency of each N-gram tuple. Use said frequencies on the test set.

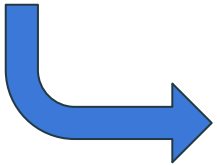
Bag-of-Words

Question: How do we represent large chunks of text to ML models?

Problems

1. ML models like fixed-length inputs, texts can vary in size.
2. Computational algorithms works with numbers, texts have characters/strings.

Answer: We use feature encoding.



Encoding: The process of turning categorical data in a dataset into numerical data.

Bag-of-Words

It’s an algorithm that transforms the text into fixed-length vectors.

It provides a representation of the text that describes the occurrence of words within a document, based on the following:

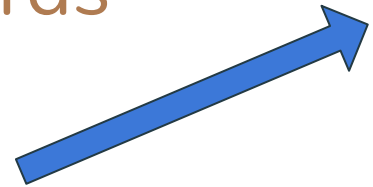
1. Vocabulary of known words.
2. A measure of the presence of known words.

We call it a “*bag*” because it discards structural information about the text, focusing solely on the semantic information that can be extracted.

Bag-of-Words

How it works:

1. Collect the data.



A collection of documents or texts, in other words your corpus.

2. Decide on a vocabulary.

An (ordered) array/list of unique words that appear in your corpus.

3. Create document vectors.

A binary array that indicates the presence of words as a boolean value (0 for absent, 1 for present) w.r.t. the vocabulary.

Bag-of-Words

Example: Consider a corpus of Netflix reviews.

Review 1: Game of Thrones is an amazing tv series!

Review 2: Game of Thrones is the best tv series!

Review 3: Game of Thrones is so great

All samples have a predetermined size so we solved the variating length text problem!

	amazing	an	best	game	great	is	of	series	so	the	thrones	tv
0	1	1	0	1	0	1	1	1	0	0	1	1
1	0	0	1	1	0	1	1	1	0	1	1	1
2	0	0	0	1	1	1	1	0	1	0	1	0

TF-IDF

Term Frequency - Inverse Document Frequency is an algorithm that associates each word in a document with a frequency that represents how relevant each word is in that document.

Documents with similar word distributions will have similar TF-IDF vectors. The applications of this measure of similarity are multiple: information retrieval, keyword extraction, text summarization etc.

We achieve TF-IDF is a composite representation achieved through multiplying the TF and IDF.

Okay, but what are these?

TF-IDF

Term Frequency (TF)

- Measure the frequency of a token in a document.
- We normalize the TF by the token count in the document in order to make the metric agnostic to the document length.

Inverse Document Frequency (IDF)

- Document Frequency = Measure the occurrence of a token in a corpus.
- IDF is computed by dividing the number of documents by the DF + 1.
- IDF will be low for high-frequency words, such as stopwords, and high for meaningful words.

Nota Bene: We use log(IDF) when computing the TF-IDF, in order to avoid extreme values when dealing with large corpuses.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

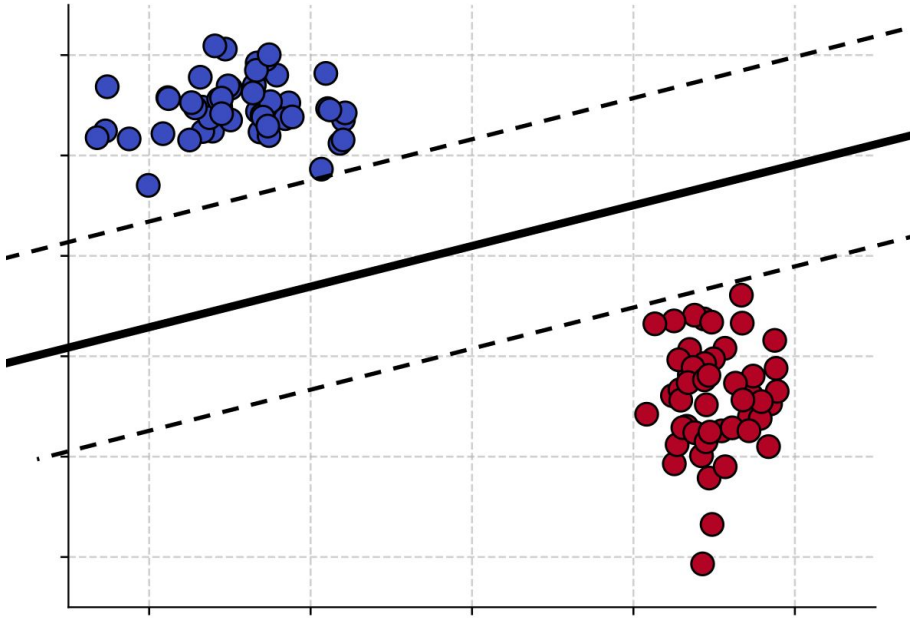
N = total number of documents

SVM: Short Revision

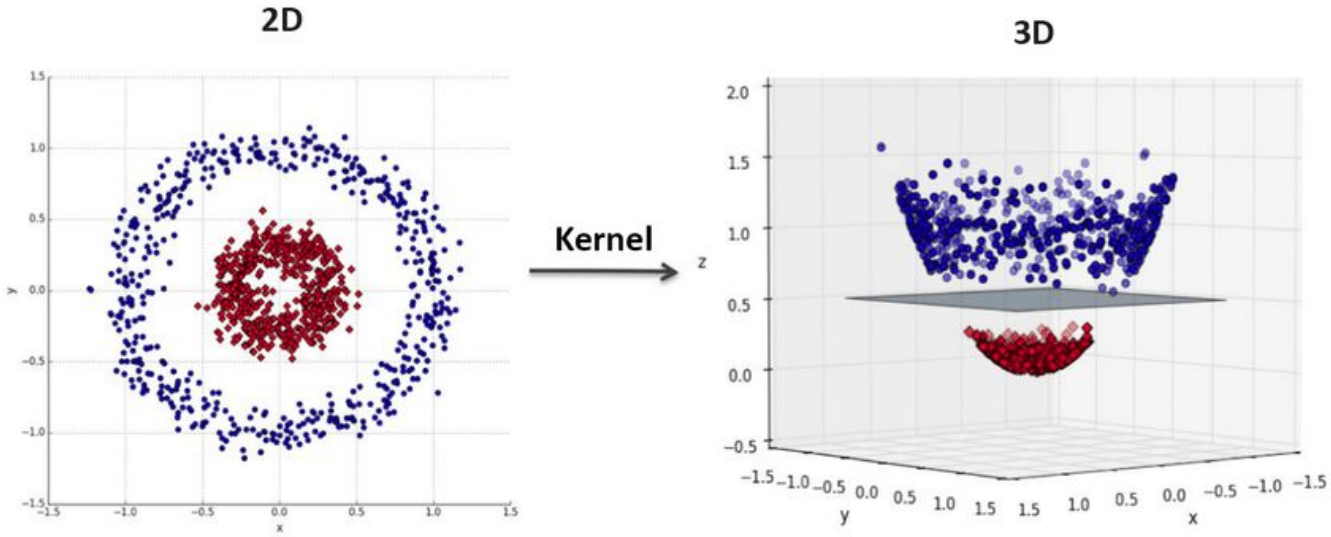
- Supervised ML model that tries to find a **maximum-margin hyperplane**.

  In layman's terms, it finds the best classifier that is as far away from the two classes as possible.

- Effective in high-dimensional spaces, thus used in text and hypertext classification.



SVM: Short Revision



A kernel function, or kernel method, is a mathematical operation which allows the model to learn in a high-dimensional feature space.

String Kernels

String kernels provide a measure of similarity between two series of tokens by counting the counting the common N-grams between the two series.

Main advantages of string kernels:

1. Language-agnostic.
2. Delimiter-agnostic.
3. Token-agnostic.
    - Characters can be interpreted as tokens, as well as words etc.
    - Due to this characteristic, they are widespread in genomics, given that they can be used on DNA or RNA sequences without the need for a preemptive split.

String Kernels

Example:

Given two sequences of characters “pineapple pi” and “apple pie”, we can build key-value pairs such as:

2-gram : sample count

pi	2
in	1
ne	1
ea	1
ap	1
pp	1
pl	1
le	1
e_	1
_p	1

ap	1
pp	1
pl	1
le	1
e_	1
_p	1
pi	1
ie	1

String Kernels

Some widely used string kernels are:

1. Linear String Kernel.
2. Polynomial String Kernel.
3. Gappy Kernel.
4. Mismatch Kernel.
5. Motif Kernel.

Useful resources for trying all of these methods:

- https://github.com/jakob-he/string-kernel
- https://pypi.org/project/string-kernels/

Input	Output	Task examples	Medical domain
Sequence of tokens	Class	**Sentiment analysis, Document Classification**	**Medical document classification**
Sequence of tokens	Sequence of classes for each token	**NER, POS tagging, Question Answering**	**Medical NER (De-identification, automatic disease and treatment labeling) Medical QA**
Sequence of tokens	Corresponding sequence of tokens	**Text summarization, Neural Translation**	**Patient notes summarization**
Sequence of tokens	Continuation of the sequence of tokens	**Text generation**	**Conditional patient notes generation**
Two Sequence of tokens	Class Example	**Relation classification**	**Gene disease relation classification**

An overview of some NLP tasks with their respective inputs and outputs.

… and more?

What else is there?

Find out in our next episode on Deep Learning models! ;)

Preparing Your Data

Data Clean-Up

Initial preprocessing steps:

- Removing punctuation.
- Removing URLs.
- Removing stopwords (“a”, “the”, “is”, “are” etc.).
- Lowercasing text.

Nota Bene: Research the task at hand before applying any of these steps. Some task (autorship and outlier detection) make use of punctuation and stopwords, while others (sentiment analysis) benefit from words such as “good”, “bad”, “don’t” etc.

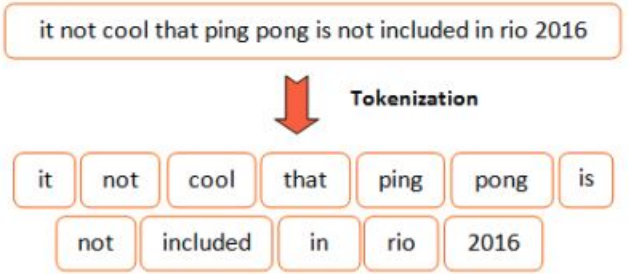
Tokenization

What is it?

Tokenization is the process of tokenizing or splitting a string or text into a list of tokens.

Why use it?

The reason we use tokenization is to access the atomic information of the text.



Tokenization

How do I apply tokenization to my corpus?

There are too many approaches to tokenization for us to discuss in detail, and they heavily depend on how you define your tokens and your purpose.

Your best bet is to do a little research on those methods and choose the one that suits the needs and scope of your project.

Lemmatization

What is it?

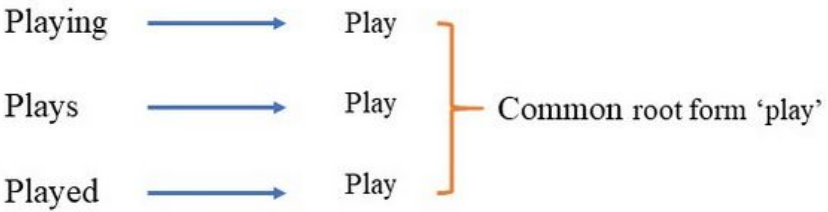
Lemmatization is the process of converting a word to its base form. That is, remove any inflectional endings and return the dictionary form of the word (known as *lemma*).

Lemmatization yields a real word, which bears semantic meaning.

Why use it?

We have previously seen how tokenization helps us access atomic information of tokens. In the case that our tokens are words, lemmatization brings helps us extract the semantic meaning of the word.

Lemmatization



am, are, is → be

Car cars, car's, cars' → car

Using above mapping a sentence could be normalized as follows:

the boy's cars are different colors → the boy car be differ color

Stemming

What is it?

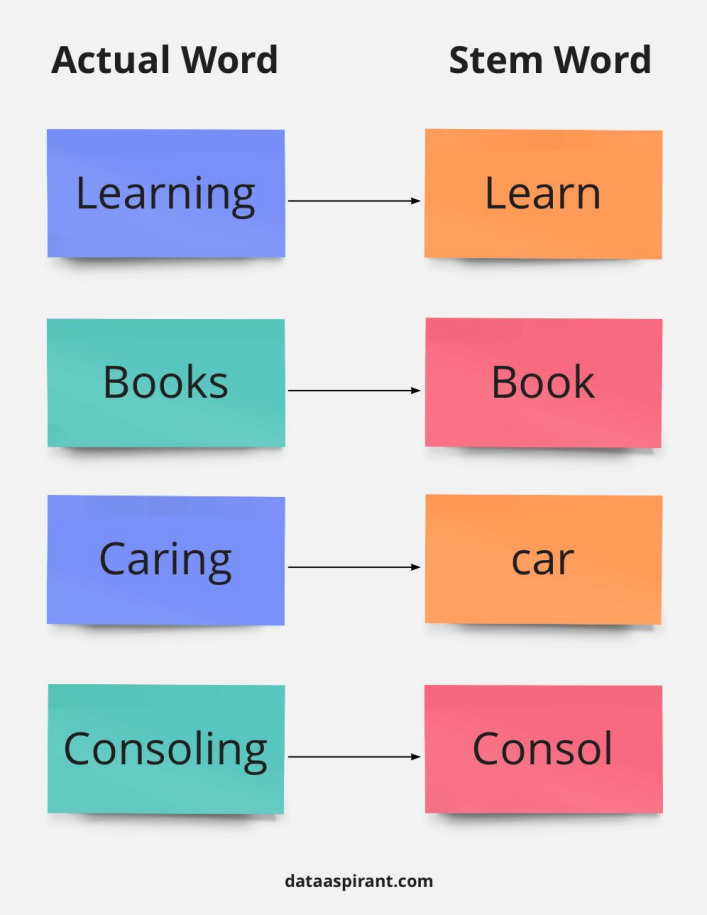
Stemming is a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

Computationally speaking, stemming returns the longest common substring for a family of words.

Why use it?

Some languages, such as English, have several variants of a single term, which add data redundancy when developing NLP applications. By applying stemming we transform those variants to their root form and.

Stemming


Actual Word
Stem Word
Learning
Learn
Books
Book
Caring
car
Consoling
Consol
dataaspirant.com

Stemming vs Lemmatization

Nota Bene: Stemming is a crude computation process that returns the longest common substring among a set of similar words, while lemmatization is a more sophisticated approach that yields a semantically significant result.

Core NLP Libraries
- spaCy, gensim and more -

spaCy

spaCy

Overview

What is spaCy?

spaCy is a an open-source, production-level, NLP library in Python.

Why use spaCy?

It provides the fastest and most accurate syntactic analysis of any NLP library released to date. While other libraries, such as NLTK or CoreNLP, might have a larger pool of algorithms, they are mostly implemented for educational purposes and not optimized for production environments.

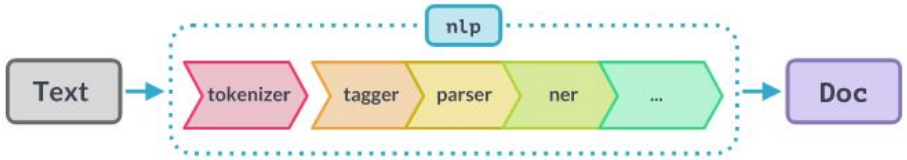
spaCy

Functionalities

NAME	DESCRIPTION
Tokenization	Segmenting text into words, punctuations marks etc.
Part-of-speech (POS) **Tagging**	Assigning word types to tokens, like verb or noun.
Dependency Parsing	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
Lemmatization	Assigning the base forms of words. For example, the lemma of “was” is “be”, and the lemma of “rats” is “rat”.
Sentence Boundary Detection (SBD)	Finding and segmenting individual sentences.
Named Entity Recognition (NER)	Labelling named “real-world” objects, like persons, companies or locations.
Entity Linking (EL)	Disambiguating textual entities to unique identifiers in a knowledge base.
Similarity	Comparing words, text spans and documents and how similar they are to each other.
Text Classification	Assigning categories or labels to a whole document, or parts of a document.
Rule-based Matching	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
Training	Updating and improving a statistical model’s predictions.
Serialization	Saving objects to files or byte strings.

spaCy

Pipelines



NAME	COMPONENT	CREATES	DESCRIPTION
tokenizer	`Tokenizer`	`Doc`	Segment text into tokens.
tagger	`Tagger`	`Token.tag`	Assign part-of-speech tags.
parser	`DependencyParser`	`Token.head`, `Token.dep`, `Doc.sents`, `Doc.noun_chunks`	Assign dependency labels.
ner	`EntityRecognizer`	`Doc.ents`, `Token.ent_iob`, `Token.ent_type`	Detect and label named entities.
lemmatizer	`Lemmatizer`	`Token.lemma`	Assign base forms.
textcat	`TextCategorizer`	`Doc.cats`	Assign document labels.
custom	custom components	`Doc._.xxx`, `Token._.xxx`, `Span._.xxx`	Assign custom attributes, methods or properties.

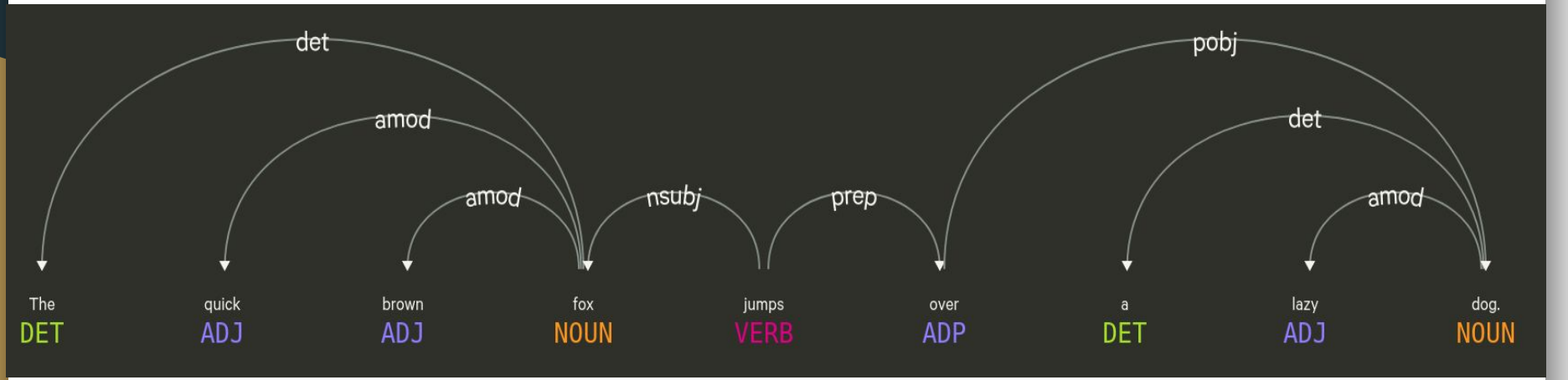
PROCESSING PIPELINE

spaCy

Visualizations:

POS Tagging

Sample text: The quick brown fox jumps over a lazy dog.



spaCy

Visualizations:

NER

```
import spacy
from spacy import displacy

text = "When Sebastian Thrun started working on self-driving cars at Google in 2007, few people outside of t

nlp = spacy.load("en_core_web_sm")
doc = nlp(text)
displacy.serve(doc, style="ent")
```

gensim

gensim

Overview

What is gensim?

gensim is an open-source NLP library in Python focused on language modelling, namely on extracting semantic topics from documents.

Why use gensim?

Optimized for handling a large corpus, while also providing multicore implementations of most of their algorithms.

... and others

Worthy Mentions

Scikit-learn:

- Has some feature encoding functionalities, as well as a wide range of ML models to choose from.

Polyglot:

- Massively multilingual NLP library, offers tokenization, NER, POS tagging, embeddings and analysis functionalities for tens, if not hundreds of languages.

Pattern:

- Good for web mining and extracting raw data. Not a full NLP library though.

NLTK:

- Provides a large variety of NLP algorithms, not necessarily optimized however.

Model Evaluation

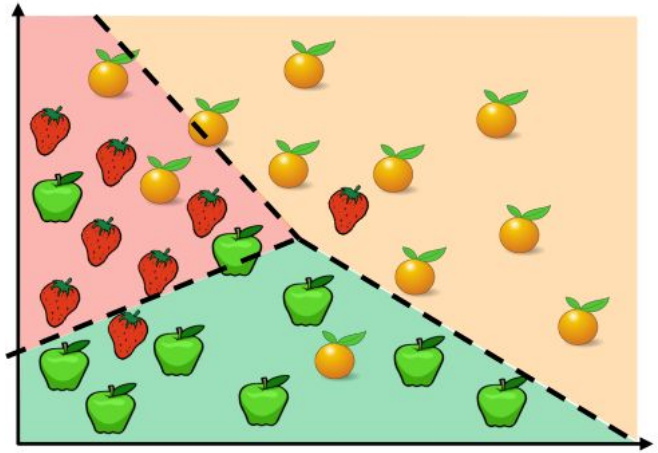
Model Evaluation

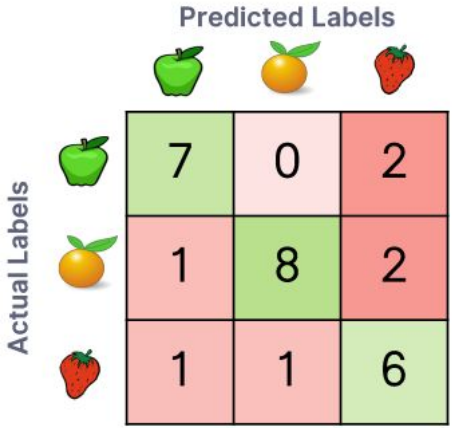
By and large there are two main approaches when assess your model's performance.

1. **Quantitative**
    - Confusion Matrix.
    - Metrics.
    - Cross-Validation.
2. **Qualitative**
    - Plots.
    - Diagrams.
    - Word distribution visualizations.

Quantitative: Confusion Matrix

A **confusion matrix** is a technique for summarizing the performance of a classification algorithm by visualizing the number of well-classified samples in opposition to those that were misclassified.



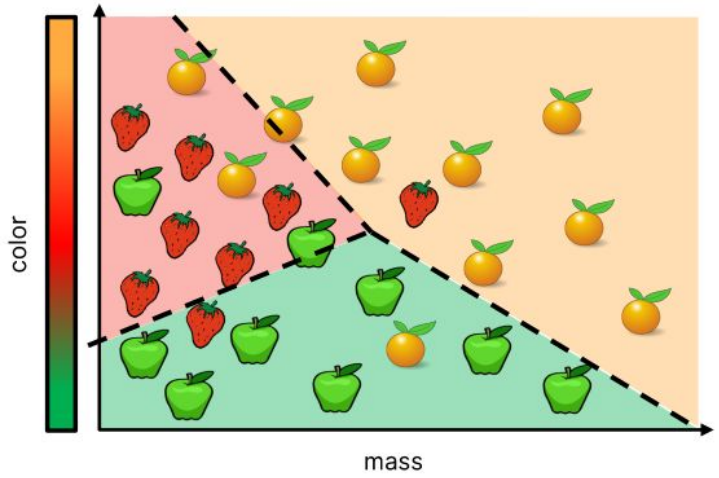


Quantitative: Metrics

1. Accuracy.
2. Precision.
3. Recall.
4. F1 Score.
5. ROC-AUC.

Metrics: Accuracy

Accuracy is the percentage of samples which were correctly classified by the ML model.



$$\text{Accuracy} = \frac{21}{28} = 0.75 = 75\%$$

Metrics: Accuracy

Pitfalls of Improper Scoring Rules

Consider the following scenario: 20 out of 1000 people have some disease. A classifier that predicts that no one has the disease has an accuracy of 98%. Efficient, right?

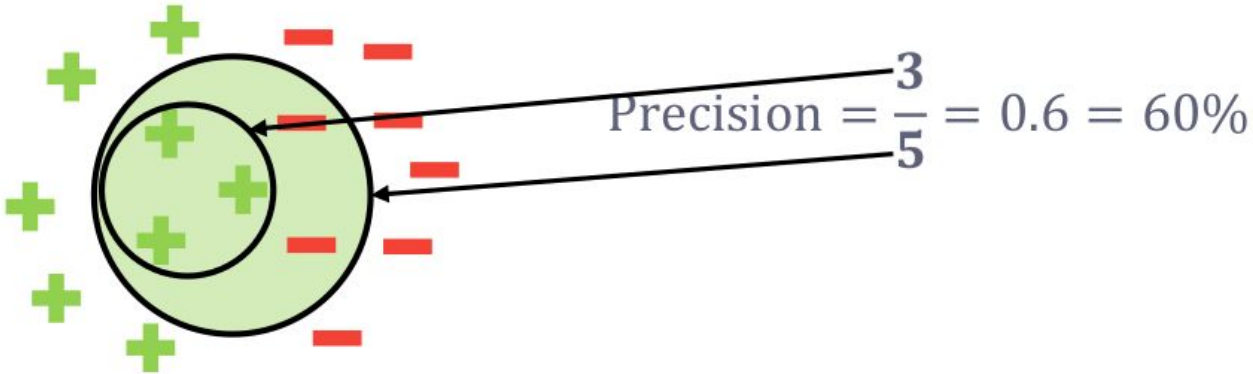
The accuracy metric has **3 major pitfalls**:

1. It fails to infer essential information once the data is unbalanced.
2. It fails to determine the conceptual difference between the two classes.
3. It fails to capture False Positive and False Negative errors.

If interested, you can read more on: https://www.fharrell.com/post/class-damage/.

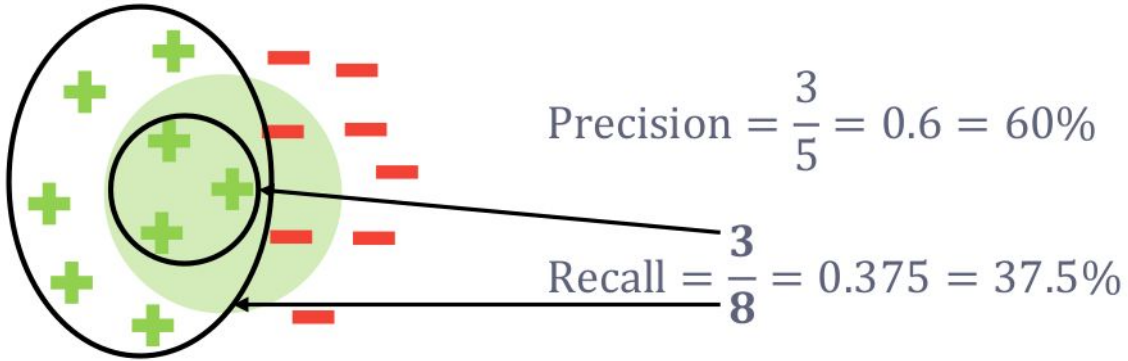
Metrics: Precision & Recall

Precision is the percentage of predicted samples that are positive.



Metrics: Precision & Recall

Recall is the percentage of positive samples that were predicted.



Metrics: F1 Score

The **F1 Score** is the harmonic mean of precision and recall.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Why F1?

False Positive and False Negatives are undesirable in most ML applications, reason for which we'd like to maximize both precision and recall. However, increasing precision decreases recall and vice-versa.

F1 provides a workaround to this issue: by combining both precision and recall, we take into account both our model's False Positive rates and False Negatives rates. Thus, optimizing the F1 is synonymous with decreasing those rates.

Metrics: ROC-AUC

ROC-AUC, or Receiver Operator Characteristic - Area Under the Curve, is a composite binary classification metric and is made up of two parts:

1. **ROC:** A probability curve that plots the **True Positive Rate** against **False Positive Rate** at various threshold values and essentially separates the ‘signal’ from the ‘noise’.

   A higher X-axis value indicates a higher number of False positives than True negatives. While a higher Y-axis value indicates a higher number of True positives than False negatives.

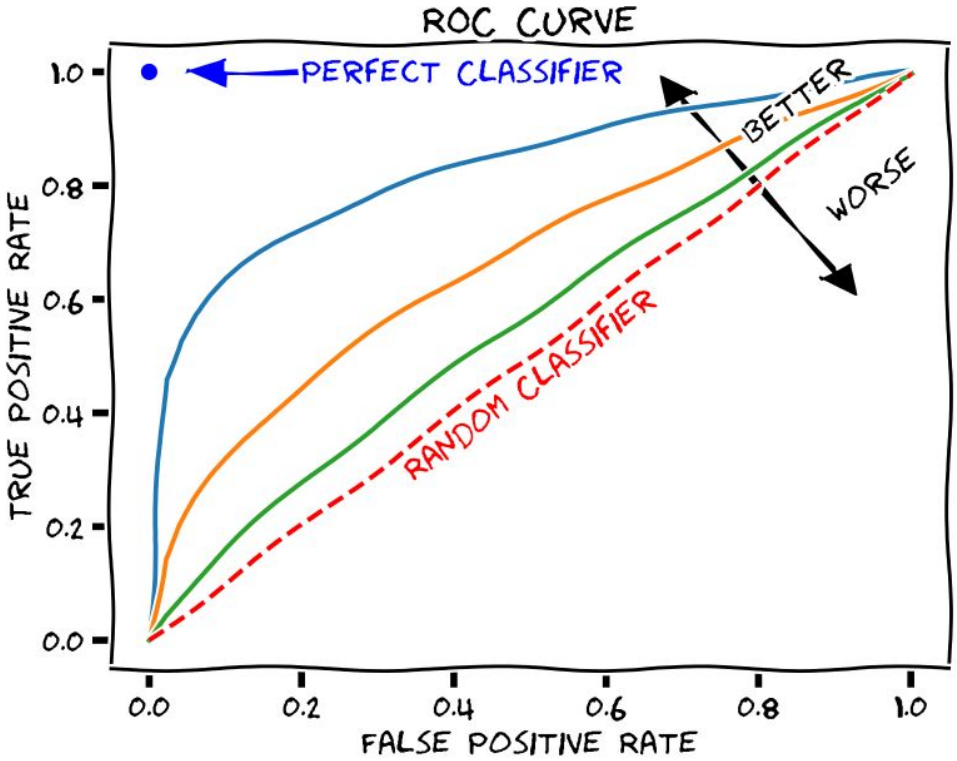
Metrics: ROC-AUC

ROC-AUC, or Receiver Operator Characteristic - Area Under the Curve, is a composite binary classification metric and is made up of two parts:

2. **AUC:** A measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

   The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

   When $0.5 < AUC < 1$, there is a high chance that the classifier will be able to distinguish between classes. If $AUC = 0.5$, the model is not able to differentiate between them, meaning either the predictions are random or constant.



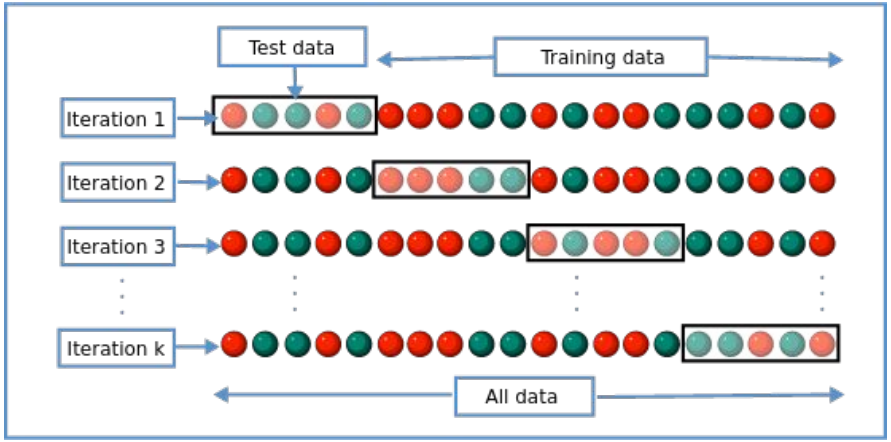
Simplified ROC-AUC visualisation: The model becomes better as the ROC curve moves upwards of the dashed line.

Quantitative: K-Fold Cross-Validation

Having too little data or using a predefined train-test set too many times can lead to overparametrization of the model.

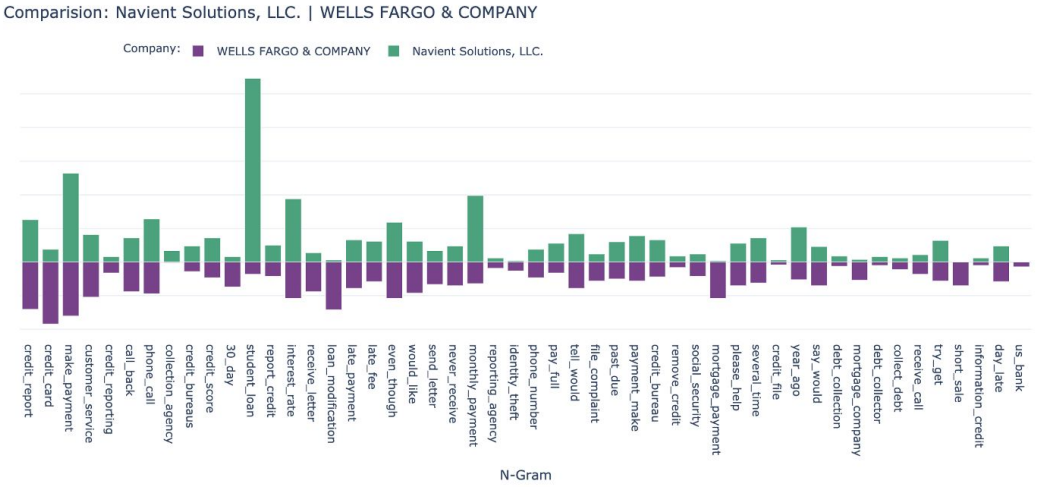
In order to avoid both these instances we use **K-Fold Cross-Validation**:

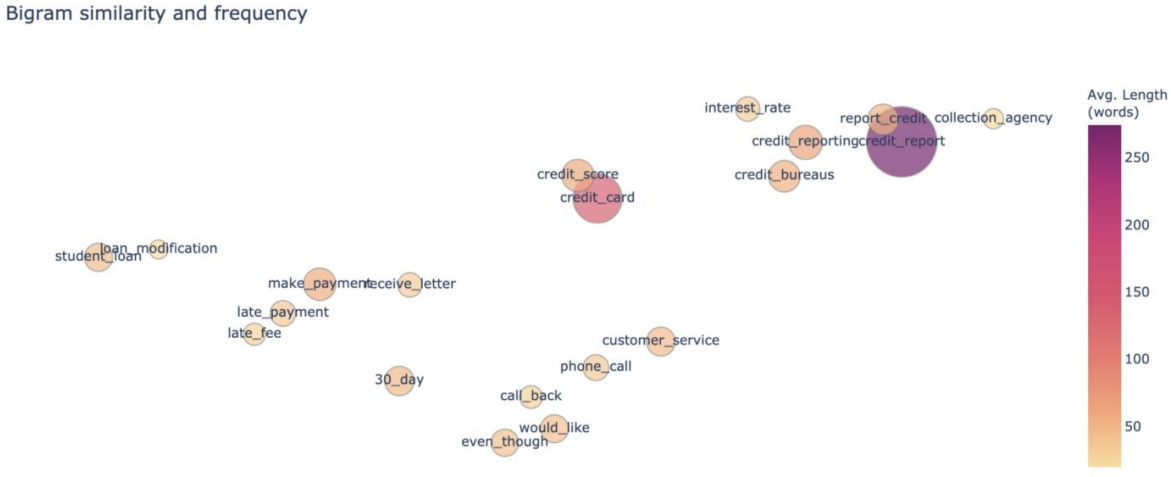
1. Split data into *k* equal parts (folds).
2. Repeat the train-test split k times, but use a different test fold each iteration.
3. Average out the errors.



Qualitative: Plots & Diagrams

Top Tip: Feel free to explore https://python-graph-gallery.com/ for all possible plots and graphs, source code included!





Qualitative: Word Distributions

Can provide insight to the way our model is perceiving and using the data.

for something useful and tell us which way to go ? " the cavern fork ##ed in a t - shape to the left and right . " uh . . . i don " t know . " drake admitted . " the only ones who have ever been in this place and made it out alive are the wizards , and they don " t exactly give us maps to be nice guys . " " you " re useless . " vlad sighed . " somebody pick one , then . " " right . " luke shrugged and walked ahead , then paused . " uh - oh . company . " two crawl were advancing , pale and gray colored , and with them was a monstrous green wolf , eyes glowing red and fangs dripping . " i " ll take the dog ! " yelling , the fighter engaged it while the other spread out . " watch out , its bite is poisonous ! " drake warned him as he and mary began stabbing and hacking at one crawl . " and these ones stu ##n , so be careful about them too ! " " these would be the stronger types , then . " vlad noted calmly , freezing the other crawl then blasting it apart with a lightning bolt . " still just as ugly . " mary smashed the crawl flat , and drake stabbed it a few more times to make

Open Discussion


01000110
01001101
01001001
1694
1864
UNIVERSITAS STVDIORVM BVCVRESTIENSIS

Thank you!


Virtute et Sapientia
1694
1694
UNIVERSITAS STVDIORVM BVCVRESTIENSIS