NLP Cheat Sheet - Introduction - Overview - Python - Starter Kit

Introduction to Natural Language Processing (NLP) tools, frameworks, concepts, resources for Python

Demo: launch binder

NLP Python Libraries

- spacy
- NLTK - similar to spacy, supports more models, simpl GUI model download `nltk.download()`
- gensim - topic modelling, accessing corpus, similarity calculations between query and indexed docs, SparseMatrixSimilarity, Latent Semantic Analysis
- lexnlp - information retrieval and extraction for real, unstructured legal text
- Holmes - information extraction, document classification, search in documents
- Pytorch-Transformers - includes BERT, GPT2, XLNet

Uncased model is better unless you know that case information is important for your task (e.g., Named Entity Recognition or Part-of-Speech tagging)

General

- PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing
- Tensorflow
- Keras

NLP Algortihms

- GPT-2 - generate fake news, text summaries
- BERT
- FinBERT - analyze sentiment of financial text
- XLnet
- ERNIE

Datasets

- Gutenberg Corpus - contains 25,000 free electronic books. `from nltk.corpus import gutenberg`
- OntoNotes 5 - corpus comprising various genres of text (news, conversational telephone speech, weblogs, usenet newsgroups, broadcast, talk shows) in three languages (English, Chinese, and Arabic) with structural information (syntax and predicate argument structure) and shallow semantics (word sense linked to an ontology and coreference).
- wiki_en_tfidf.mm in gensim 3.9M documents, 100K features (distinct tokens) and 0.76G non-zero entries in the sparse TF-IDF matrix. The Wikipedia corpus contains about 2.24 billion tokens in total.
- GPT-2 Dataset
- Brown corpus - contains text from 500 sources, and the sources have been categorized by genre, such as news, editorial, and so on.
- Reuters Corpus - 10,788 news documents totaling 1.3 million words
- Newsfilter.io stock market news corpus - contains over 4 million press releases, earnings reports, FDA drug approvals, analyst ratings, merger agreements and many more covering all US companies listed on NASDAQ, NYSE, AMEX
- Kaggle - All the news, 143K articles
- Kaggle - Daily news for stock market prediction
- CNN News
- AG News - PyTorch integrated

Installation:

spacy (good for beginners; use NLTK for bigger projects)

```
pip install spacy
python -m spacy download en
# python -m spacy download en_core_web_lg
```

LexNLP (good for dealing with legal and financial documents; installation guide here)

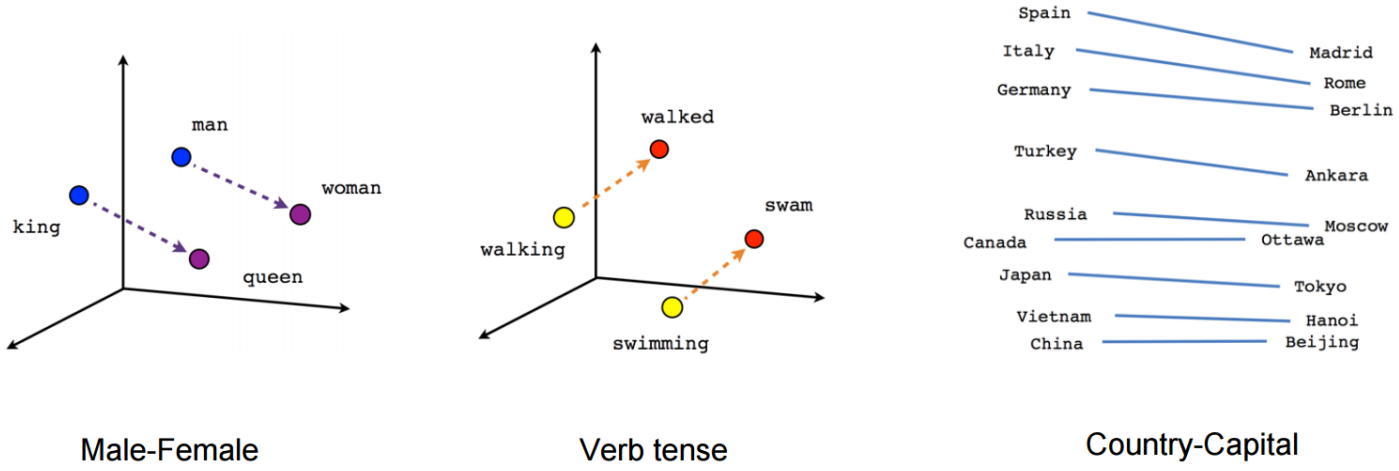
```
pip install https://github.com/LexPredict/lexpredict-lexnlp/archive/master.zip
python # to open REPL console
>>> import nltk
>>> nltk.download() # download all packages
```

Double-click (or enter) to edit

Concepts

Word embeddings (=word vectors)

Visualizing word vectors using PCA. Paper: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf



Male-Female Verb tense Country-Capital

- Word embeddings are vector representation of words.
- Example sentence: word embeddings are words converted into numbers.
- A word in this sentence may be “Embeddings” or “numbers ” etc.
- A dictionary may be the list of all unique words in the sentence, eg [‘Word’,’Embeddings’,’are’,’Converted’,’into’,’numbers’]
- A vector representation of a word may be a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else.

Example

- numbers = [0,0,0,0,0,1]
- converted = [0,0,0,1,0,0]

** Either use pre-trained word vectors or train our own**

Pre-trained word embeddings:

- Word2Vec (Google, 2013), uses Skip Gram and CBOW
- Vectors trained on Google News (1.5GB) - vocabulary of 3 million words trained on around 100 billion words from the google news dataset
- GloVe (Stanford)
- Stanford Named Entity Recognizer (NER)
- LexPredict: pre-trained word embedding models for legal or regulatory text
- LexNLP legal models - US GAAP, finaical common terms, US federal regulators, common law

Create word vectors yourself

```
import gensim
word2vev_model = gensim.models.word2vec.Word2Vec(sentence_list)
```

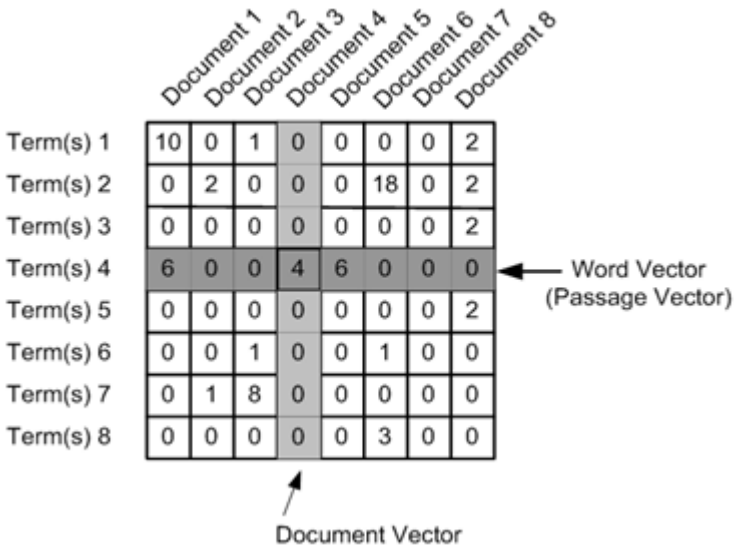
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

How to create word vectors?

- Count-based methods compute the statistics of how often some word co-occurs with its neighbor words in a large text corpus, and then map these count-statistics down to a small, dense vector for each word.
- Predictive models directly try to predict a word from its neighbors in terms of learned small, dense embedding vectors (considered parameters of the model).
    - Example: Word2vec (Google)

1. Count based word embeddings

Count Vector (= Document Term Matrix)



TF-IDF

Term Frequency - Inverse Document Frequency

- Term frequency equals the number of times a word appears in a document divided by the total number of words in the document.
- Inverse document frequency calculates the weight of rare words in all documents in the corpus, with rare words having a high IDF score, and words that are present in all documents in a corpus having IDF close to zero.

(sklearn) in Python has a function TfidfVectorizer() that will compute the TF-IDF values for you

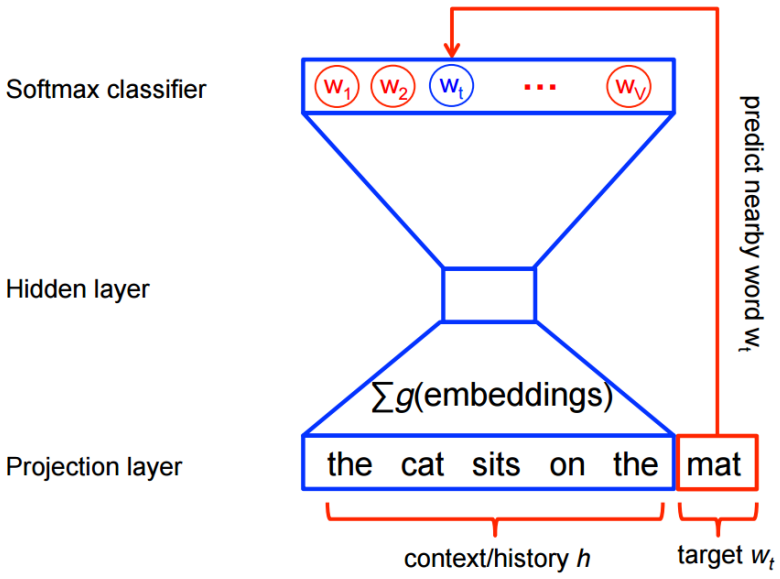
```
from sklearn.feature_extraction.text import TfidfVectorizer

# Write a function for cleaning strings and returning an array of ngrams
def ngrams_analyzer(string):
    string = re.sub(r'[,-./]', r'', string)
    ngrams = zip(*[string[i:] for i in range(5)])  # N-Gram length is 5
    return [''.join(ngram) for ngram in ngrams]

# Construct your vectorizer for building the TF-IDF matrix
vectorizer = TfidfVectorizer(analyzer=ngrams_analyzer)

# Credits: https://towardsdatascience.com/group-thousands-of-similar-spreadsheet-text-cell
```

Co-Occurrence Vector

2. Prediction based word embeddings

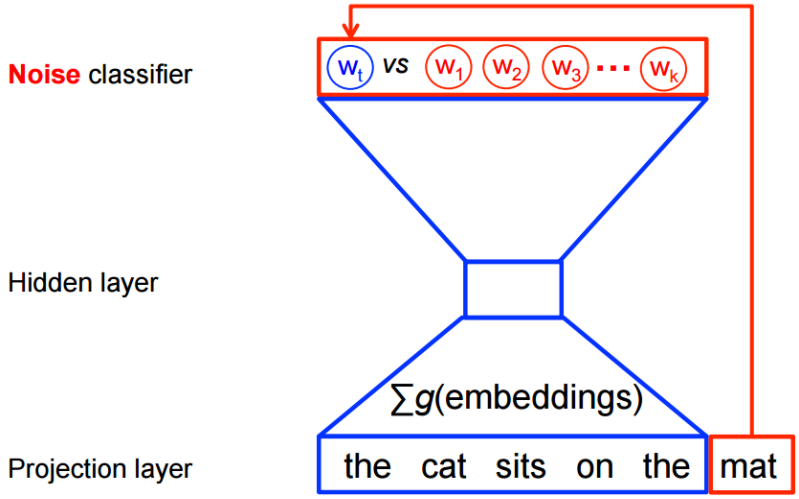
- Uses Neural Networks
- CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the')
- Skip-gram does the inverse and predicts source context-words from the target words

CBOW (Continuous Bag of words)

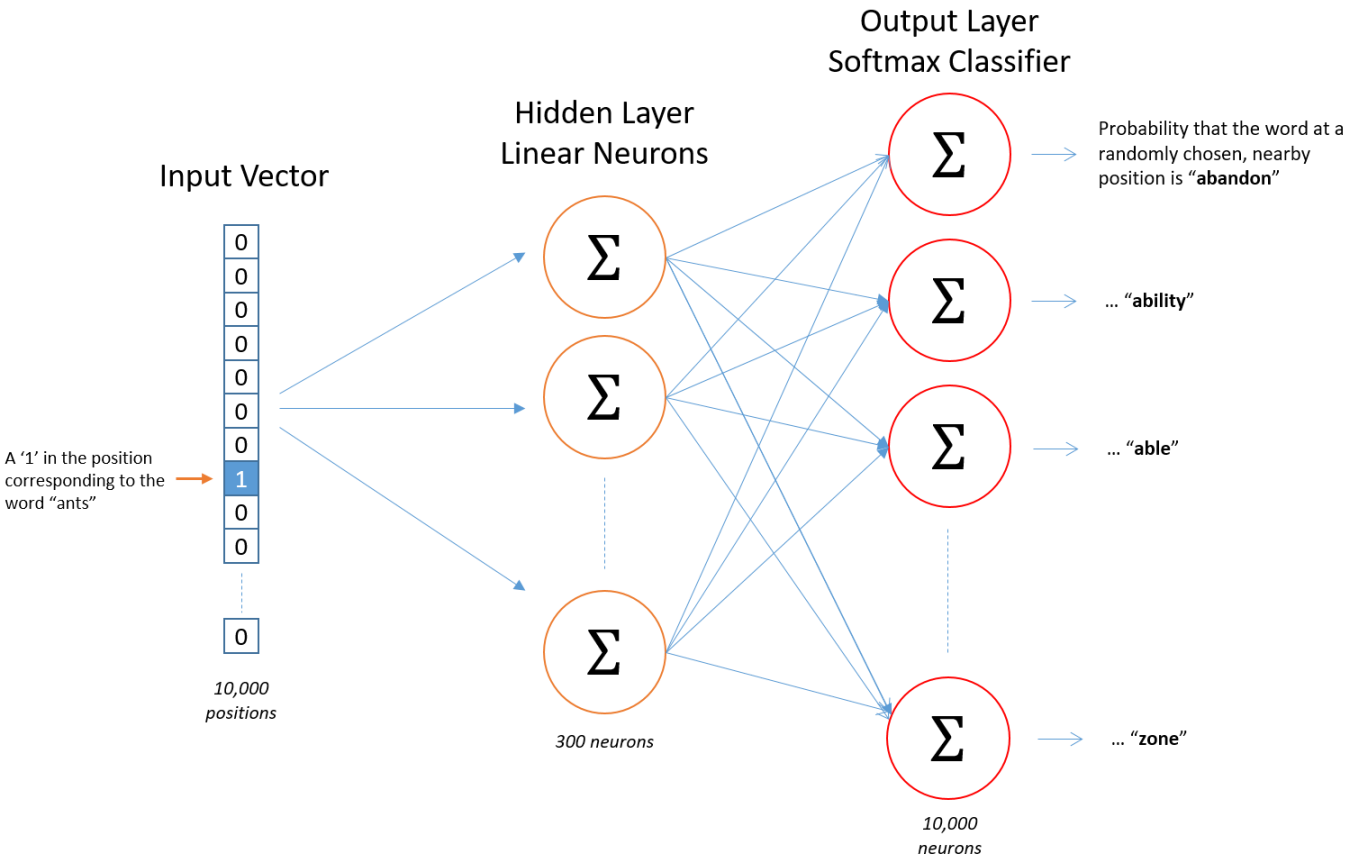


Skip Gram

Skip – gram follows the same topology as of CBOW. It just flips CBOW's architecture on its head. The aim of skip-gram is to predict the context given a word



Outcome



Bag of Words

```
1 # John likes to watch movies. Mary likes movies too.
2 BoW1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
```

spacy

```
import spacy
```

```
# Import dataset
nlp = spacy.load("en")
# Import large dataset. Needs to be downloaded first.
# nlp = spacy.load("en_core_web_lg")
```

Stop Words

Stop words are the very common words like ‘if’, ‘but’, ‘we’, ‘he’, ‘she’, and ‘they’. We can usually remove these words without changing the semantics of a text and doing so often (but not always) improves the performance of a model.

```
# spacy: Removing stop words
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS

print('spacy: Number of stop words: %d' % len(spacy_stopwords))
```

```
spacy: Number of stop words: 326
```

```
# nltk: Removing stop words
from nltk.corpus import stopwords
english_stop_words = stopwords.words('english')

print('ntlk: Number of stop words: %d' % len(english_stop_words))
```

```
ntlk: Number of stop words: 179
```

```
text = 'Larry Page founded Google in early 1990.'
doc = nlp(text)
tokens = [token.text for token in doc if not token.is_stop]
print('Original text: %s' % (text))
print()
print(tokens)
```

```
Original text: Larry Page founded Google in early 1990.

['Larry', 'Page', 'founded', 'Google', 'early', '1990', '.']
```

Spans

Part of a given text. So doc[2:4] is a span starting at token 2, up to – but not including! – token 4.

Docs: https://spacy.io/api/span

[ ] ↳ 2 cells hidden

Token and Tokenization

Segmenting text into words, punctuation etc.

- Sentence tokenization
- Word tokenization

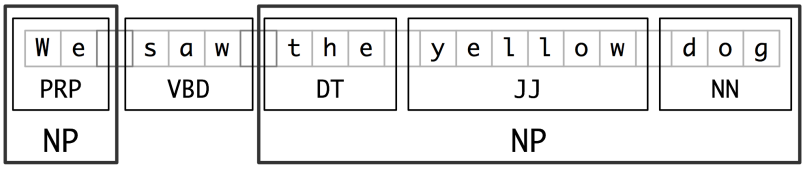
Docs: https://spacy.io/api/token

[ ] ↳ 2 cells hidden

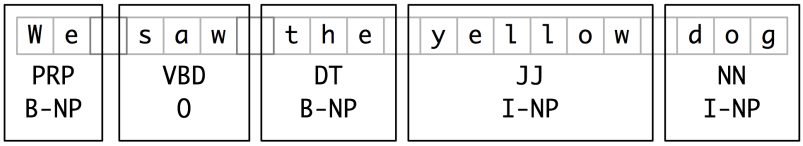
Chunks and Chunking

Segments and labels multi-token sequences.

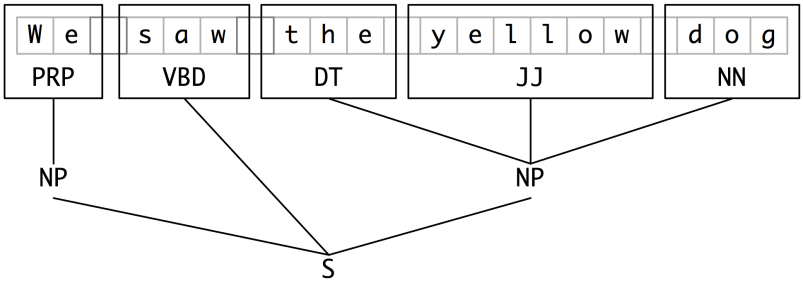
- Each of these larger boxes is called a chunk.
- Like tokenization, which omits whitespace, chunking usually selects a subset of the tokens.
- The pieces produced by a chunker do not overlap in the source text.



Segmentation and Labeling at both the Token and Chunk Levels



Tag Representation of Chunk Structures



Tree Representation of Chunk Structures

Credits: https://www.nltk.org/book/ch07.html

Chinks and Chinking

Chink is a sequence of tokens that is not included in a chunk.

Credits: https://www.nltk.org/book/ch07.html

Part-of-speech (POS) Tagging

Assigning word types to tokens like verb or noun.

POS tagging should be done straight after tokenization and before any words are removed so that sentence structure is preserved and it is more obvious what part of speech the word belongs to.

[ ] *5 cells hidden*

Stemming

Stemming is the process of reducing words to their root form.

Examples:

- cats, catlike, catty → cat
- fishing, fished, fisher → fish

There are two types of stemmers in NLTK: Porter Stemmer and Snowball stemmers

Credits

[ ] *1 cell hidden*

Lemmatization

Assigning the base form of word, for example:

- "was" → "be"
- "rats" → "rat"

[ ] *1 cell hidden*

Sentence Detection

Finding and segmenting individual sentences.

[ ] *1 cell hidden*

Dependency Parsing

Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.

[ ] *2 cells hidden*

Base noun phrases

[ ] *1 cell hidden*

Named Entity Recognition (NER)

What is NER? Labeling "real-world" objects, like persons, companies or locations.

2 popular approaches:

- Rule-based
- ML-based:
  - Multi-class classification
  - Conditional Random Field (probabilistic graphical model)

Datasets:

- Kaggle, IOB, POS tags

Credits: https://medium.com/@yingbiao/ner-with-bert-in-action-936ff275bc73

Entities supported by spacy:

- PERSON People, including fictional.
- NORP Nationalities or religious or political groups.
- FAC Buildings, airports, highways, bridges, etc.
- ORG Companies, agencies, institutions, etc.
- GPE Countries, cities, states.
- LOC Non-GPE locations, mountain ranges, bodies of water.
- PRODUCT Objects, vehicles, foods, etc. (Not services.)
- EVENT Named hurricanes, battles, wars, sports events, etc.
- WORK_OF_ART Titles of books, songs, etc.

- LAW Named documents made into laws.
- LANGUAGE Any named language.
- DATE Absolute or relative dates or periods.
- TIME Times smaller than a day.
- PERCENT Percentage, including ”%“.
- MONEY Monetary values, including unit.
- QUANTITY Measurements, as of weight or distance.
- ORDINAL “first”, “second”, etc.
- CARDINAL Numerals that do not fall under another type.

Alternatives to spacy

LexNLP entities:

- acts, e.g., “section 1 of the Advancing Hope Act, 1986”
- amounts, e.g., “ten pounds” or “5.8 megawatts”
- citations, e.g., “10 U.S. 100” or “1998 S. Ct. 1”
- companies, e.g., “Lexpredict LLC”
- conditions, e.g., “subject to …” or “unless and until …”
- constraints, e.g., “no more than” or “
- copyright, e.g., “(C) Copyright 2000 Acme”
- courts, e.g., “Supreme Court of New York”
- CUSIP, e.g., “392690QT3”
- dates, e.g., “June 1, 2017” or “2018-01-01”
- definitions, e.g., “Term shall mean …”
- distances, e.g., “fifteen miles”
- durations, e.g., “ten years” or “thirty days”
- geographic and geopolitical entities, e.g., “New York” or “Norway”
- money and currency usages, e.g., “$5” or “10 Euro”
- percents and rates, e.g., “10%” or “50 bps”
- PII, e.g., “212-212-2121” or “999-999-9999”
- ratios, e.g.,” 3:1” or “four to three”
- regulations, e.g., “32 CFR 170”
- trademarks, e.g., “MyApp (TM)”
- URLs, e.g., “http://acme.com/”

Stanford NER entities:

- Location, Person, Organization, Money, Percent, Date, Time

NLTK

- NLTK maximum entropy classifier

[ ] ↳ *8 cells hidden*

Text Classification

Two types:

- binary classification (text only belongs to one class)
- multi-class classification (text can belong to multiple classes)

Assigning categories or labels to a whole document, or parts of a document.

Approach:

- calculate document vectors for each document
- use kNN to calculate clusters based on document vectors
- each cluster represents a class of documents that are similar to each other

[ ] ↳ *2 cells hidden*

Similarity

How similar are two documents, sentences, token or spans?

Cosine similarity (also known as: L2-normalized dot product of vectors) is a formula used to calculate how similar two given word vectors are.

How to calculate Cosine similarity?

- spacy (see example below)
- scikit: [sklearn.metrics.pairwise.cosine_similarity]

[ ] ↳ *9 cells hidden*

n-grams: Unigram, bigrams, trigrams

- Unigram = one word, eg the, and, of, hotel
- Bigrams = two consecutive words, eg the hotel, in seattle, the city
- Trigrams = three consecutive words, eg easy access to, high speed internet, the heart of

Credits: [https://towardsdatascience.com/building-a-content-based-recommender-system-for-hotels-in-seattle-d724f0a32070]

[ ] ↳ *6 cells hidden*

Visualization

[ ] *4 cells hidden*

Kernels

Used by

- Support Vector Machines (SVMs)
- Principal Component Analysis (PCA)

Useful for

- classification tasks

Also known as

- kernel function
- similarity function

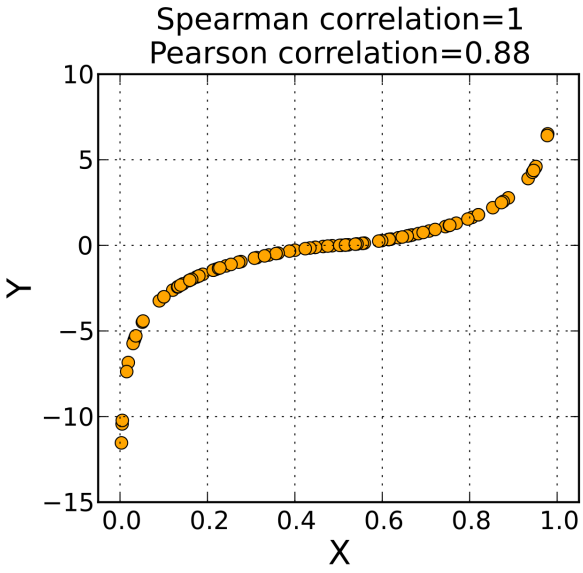
Opposite of kernels: vectors

Source:

- Wikipedia

2 cells hidden

Spearman's Rank Correlation Coefficient



Credits: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

kNN

k-nearest neighbors algoritm

Useful for

- classification

Text Summarization

- How to Make a Text Summarizer
- How to Prepare News Articles for Text Summarization

Sentiment Analysis

Is text fact or opinion? Only perform sentiment analysis on opinion, not facts.

Sentiments:

- positive
- neutral
- negative

2 ways:

- rule-based uses lexicon with polarity score per word. Count positive and negative words. Doesn't provide training data.
- automatic using machine learning (=classification problem). Needs training data.

Sentiment analysis can be performed with ntlk's `SentimentIntensityAnalyzer`

See: https://www.nltk.org/api/nltk.sentiment.html#module-nltk.sentiment.vader

Learning resources:

- https://www.youtube.com/watch?v=3Pzni2yfGUQ
- https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184

[ ] *1 cell hidden*

Logistic Regression

A classification model that uses a sigmoid function to convert a linear model's raw prediction () into a value between 0 and 1. You can interpret the value between 0 and 1 in either of the following two ways:

- As a probability that the example belongs to the positive class in a binary classification problem.
- As a value to be compared against a classification threshold. If the value is equal to or above the classification threshold, the system classifies the example as the positive class. Conversely, if the value is below the given threshold, the system classifies the example as the negative class.

https://developers.google.com/machine-learning/glossary/#logistic-regression

RNN

Recurrent neural networks

- Size changes depending on input/output (in contrast to neural network like CNN)

LSTM

Long Short-Term Mermoy

ToDo

[ ] *1 cell hidden*

Levenshtein distance

[ ] *1 cell hidden*

Regularization

Markov Decision Process

- State -> action -> state -> action ...
- Agent
- Set of actions

- Transitions
- Discount factor
- Reward

Probability to discard words to reduce noise

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

Credits: https://towardsdatascience.com/how-to-train-custom-word-embeddings-using-gpu-on-aws-f62727a1e3f6

Loss functions

A measure of how far a model's predictions are from its label.

In contrast to:

- reward function

SSE (sum of squared of the errors)

Mean Squared Errors (MSE)

Mean Squared Error (MSE) is a common loss function used for regression problems.

Mean squared error of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

Can be used for regression problems (say, to predict the price of a house).

Alternatives:

- Binary Crossentropy Loss (is better for dealing with probabilities)

Binary Crossentropy Loss

Used in binary classification tasks, ie model outputs a probability (a single-unit layer with a sigmoid activation), we'll use the binary_crossentropy loss function.

Cross-entropy loss

Sparse Categorical Crossentropy

Used in image classification task

Log loss

Used in logistic regression tasks

Optimizer

This is how the model is updated based on the data it sees and its loss function.

Gradient Descent

Optimization algorithm for finding the minimum of a function.

Stochastic Gradient Descent (SGD)

Adam

AdaBoost

AdaGrad

NN Frameworks

- Keras (best learning tool for beginners)
- PyTorch (dynamic)
- Tensorflow (declerative programming, can run on Apache Spark)

Classification

- Binary
- Not binary

Activation function

A function (for example, ReLU or sigmoid) that takes in the weighted sum of all of the inputs from the previous layer and then generates and passes an output value (typically nonlinear) to the next layer.

https://developers.google.com/machine-learning/glossary/#activation_function

Softmax Function

A function that provides probabilities for each possible class in a multi-class classification model. The probabilities add up to exactly 1.0. For example, softmax might determine that the probability of a particular image being a dog at 0.9, a cat at 0.08, and a horse at 0.02.

Example: last layer is a 10-node softmax layer—this returns an array of 10 probability scores that sum to 1.

Sigmoid

A function that maps logistic or multinomial regression output (log odds) to probabilities, returning a value between 0 and 1

Sigmoid function converts /sigma into a probability between 0 and 1.

ReLU (Rectified Linear Unit)

- If input is negative or zero, output is 0.
- If input is positive, output is equal to input.

Performance measure

Accuracy

Used when taining a neural network.

- training loss decreases with each epoch
- training accuracy increases with each epoch

Precision

TP/(TP+FP)

- TP=true positive
- FP=false positive

Recall

TP/(TP+FN)

F1 score

(2 × Precision × Recall) / (Precision + Recall)

Mean Absolute Error

A common regression metric is Mean Absolute Error (MAE).

Mean Squared Error

Early stopping

Early stopping is a useful technique to prevent overfitting.

Regularization

L1 Regularization

penalizes weights in proportion to the sum of the absolute values of the weights

https://developers.google.com/machine-learning/glossary/#L1_regularization

L2 Regularization

penalizes weights in proportion to the sum of the squares of the weights

Sparsity

The number of elements set to zero (or null) in a vector or matrix divided by the total number of entries in that vector or matrix.

Ranking

Wilson-Score Interval

Used by Reddit to rank comments.

Euclidean Ranking

Cosine Ranking

XLNet + BERT in spacy

https://spacy.io/models/en#en_pytt_xlnetbasecased_lg

Latent Dirichlet Allocation

Confusion Matrix

A confusion matrix is a table where each cell `[i,j]` indicates how often label `j` was predicted when the correct label was `i`.

Naive Bayes Classifiers

- Every feature gets a say in determining which label should be assigned to a given input value.
- To choose a label for an input value, the naive Bayes classifier begins by calculating the prior probability of each label, which is determined by checking frequency of each label in the training set.

Credits: https://www.nltk.org/book/ch06.html

[ ] *1 cell hidden*