Intro to NLP: Text Categorization and Topic Modeling

Sanghamitra Deb

Lead Data Scientist

Chegg Inc.

Outline

Topic Modeling

- Introduction & Examples
- Latent Dirichlet Allocation
- Application using genism

- Dataset --- Recipes

Text Categorization

- Data Centric AI
  - Rule Based Learning
- TfIdf features
- Simple feedforward NN classifier
- CNN
  - Word based
  - Character based

- Dataset : Spam

Metrics

- Precision-Recall
- Thresholding
- Confusion Matrix

Topic Modeling

Topic Modeling

Text mining technique that can be used to understand , organize and summarize and large corpus of text into recognizable themes or topics. It’s a method to discover patterns in your data.

Example :

'ham bone onion celery carrot water large stock pot bring boil reduce heat simmer skim foam occasionally meat start separate bone remove bone allow cool add use ham flavor well garlic salt pepper add bean kind like tomato continue simmer remove ham bone return ham soup mixture cubed ham point recommend taste test soup seasoning careful bite tongue grab throat’,

'cut apple small slice about slice apple minuscule mush cook shred baby carrot bowl bowl mix sugar cardamon cinnamon whisk together unroll crescent_roll slightly floured_surface silpat separate triangle brush dough melt butter divide apple evenly dough sprinkle carrot shredding apple drizzle tsp honey apple carrot mixture sprinkle sugar mixture apple carrot roll fat side narrow side encompass apple carrot mixture place sheet cook min remove oven let cool drizzle remain honey roll enjoy’,

'add grenadine orgeat bottom glass mound ice glass add white rum add sweet sour pineapple juice float bacardi dark rum rum top go glass aprox’,

'preheat oven cut potato inch chunk toss bowl potato oil rosemary lemon juice lemon zest garlic chuck lot baking tray forget scrape herb oil bowl bake minute listen funky tune enjoy yumminess come check potato firm yet give fork squish little bit fork put back in oven minute gas_mark’,

'place bean large bowl sit aside heat oil large fry pan add onion garlic corn cumin saut about minute medium heat add green chili sauce saut minute place bowl bean let cool mash blender okay piece mash add salsa chili powder flour breadcrumb stir blend ingredient rubber_spatula form patty place wax_paper store tupperware ready cook cook spray side saut medium heat brown repeat other side'

Topic Modeling

Text mining technique that can be used to understand , organize and summarize and large corpus of text into recognizable themes or topics. It’s a method to discover patterns in your data.

Example :

'ham bone onion celery carrot water large stock pot bring boil reduce heat simmer skim foam occasionally meat start separate bone remove bone allow cool add use ham flavor well garlic salt pepper add bean kind like tomato continue simmer remove ham bone return ham soup mixture cubed ham point recommend taste test soup seasoning careful bite tongue grab throat’,

'cut apple small slice about slice apple minuscule mush cook shred baby carrot bowl bowl mix sugar cardamon cinnamon whisk together unroll crescent_roll slightly floured_surface silpat separate triangle brush dough melt butter divide apple evenly dough sprinkle carrot shredding apple drizzle tsp honey apple carrot mixture sprinkle sugar mixture apple carrot roll fat side narrow side encompass apple carrot mixture place sheet cook min remove oven let cool drizzle remain honey roll enjoy’,

'add grenadine orgeat bottom glass mound ice glass add white rum add sweet sour pineapple juice float bacardi dark rum rum top go glass aprox’,

'preheat oven cut potato inch chunk toss bowl potato oil rosemary lemon juice lemon zest garlic chuck lot baking tray forget scrape herb oil bowl bake minute listen funky tune enjoy yumminess come check potato firm yet give fork squish little bit fork put back in oven minute gas_mark’,

'place bean large bowl sit aside heat oil large fry pan add onion garlic corn cumin saut about minute medium heat add green chili sauce saut minute place bowl bean let cool mash blender okay piece mash add salsa chili powder flour breadcrumb stir blend ingredient rubber_spatula form patty place wax_paper store tupperware ready cook cook spray side saut medium heat brown repeat other side'

Does this corpus contain anything about baking?

Let’s do topic modeling!

- 'preheat oven degree grease round cake pan oven preheat begin make cake batter dissolve cocoa espresso stir buttermilk cocoa mixture set aside cool whisk together flour bake soda salt bowl beat butter medium speed about minute gradually add sugar butter mixture light fluffy slowly add egg vanilla butter continue beat sure scrape side bowl go so ingredient incorporate fold flour cocoa mixture butter mixture alternate flour cocoa follow order flour mixture cocoa mixture flour mixture cocoa mixture flour mixture overmix’,

- 'pancake large bowl mix flour bake powder bake soda sugar salt set aside separate egg yolk egg white beat egg white cream tartar high speed high speak form set aside beat milk egg yolk vanilla melt butter together slow speed mixed add milk mixture flour mixture slowly beat medium speed completely blend add egg white batter mix together fold rest egg white carefully mix’,

- 'large bowl cream butter minute add sugar beat minute long light fluffy beat egg separate bowl whisk together juice third bowl whisk flour bake soda bake powder salt spice stir third flour mixture butter mixture mix pumpkin mixture add third flour mixture remain pumpkin mixture sprinkle remain flour mixture raisin nuts mix spoon batter grease’,

- 'mix vinegar milk make sour milk add vanilla sour milk set aside large mixing bowl cream together sugar shortening light fluffy add egg time separate bowl blend together bake soda salt flour bake powder add flour mixture alternately sour milk mixture cream sugar egg mixture cover refrigerate overnight roll dough desire thickness’,

- 'cream together oil sugar well blend add egg beat minute beat vanilla rind sift together flour soda salt beat flour mixture orange egg mixture fold uncle fruit mix'

What else can I cook?

- 'ratio rice water quantity rice increase cup rice cup water cup rice cup water method put water boil meanwhile wash rice time water start boil add rice salt check min see rice cook grain still firm raw break grain finger check drain cover fitting flat lid invert sink then leave inverted vessel good minute make sure water drain also drain colander drain pasta straighten’,

- 'large saucepan bring salt water boil simmer lentil low minute tender drain water lentil set aside pan saute onion garlic oil add raisin date spice mix set aside cook rice like sure fire way use rice cooker rinse rice cold water water runs_clear add rice cooker cover cup water add teaspoon salt drizzle oil cook rice do transfer cooked rice large bowl same pot add oil just cover bottom surface add potato slice add layer rice add layer lentil raisin mixture continue layer end final layer rice cover cook minute drizzle melt butter saffron water rice cover top pot rice cooker tea towel prevent steam escape top cook low minute want make potato crispy so cook slowly rice cooker continuously hit cook button process happen totally worth’,

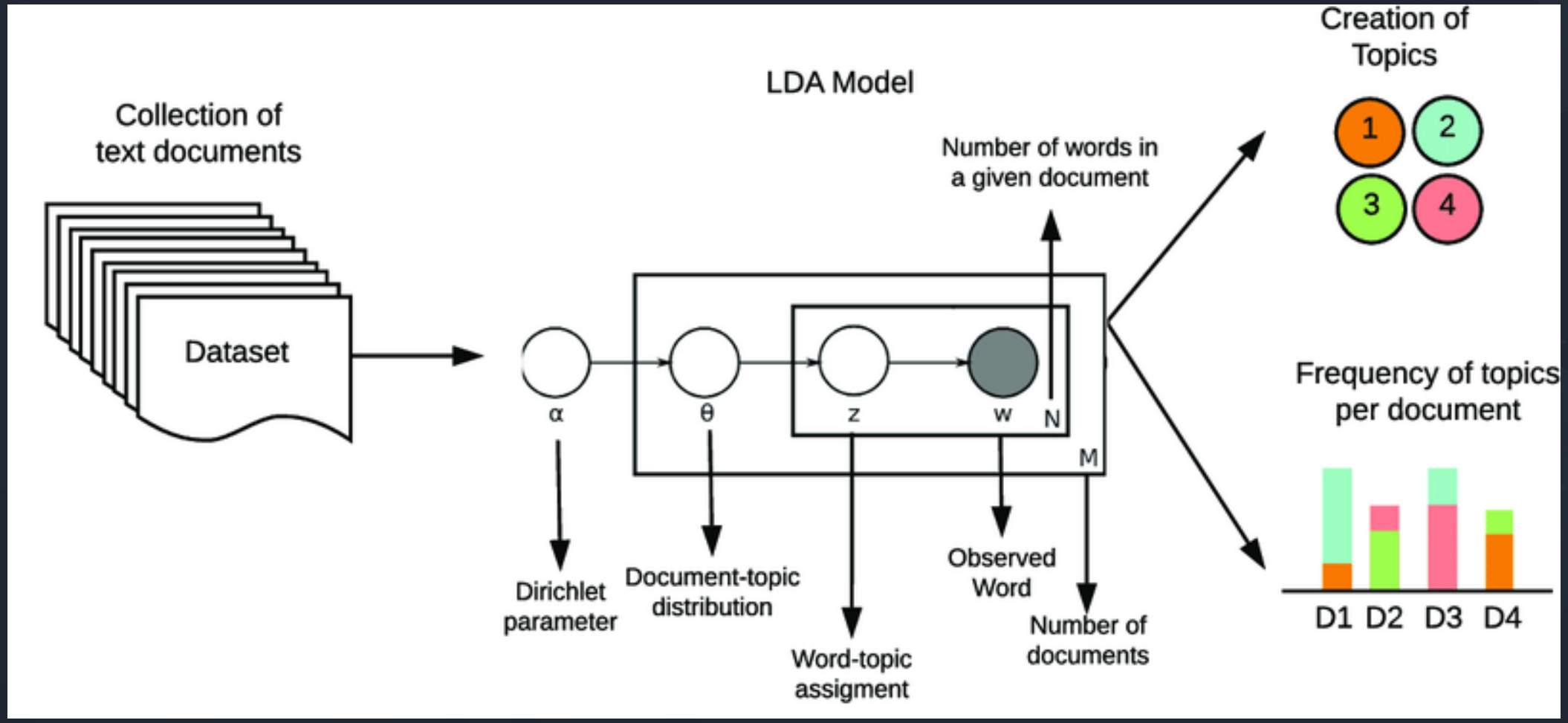
- 'wash rice several_time water runs_clear soak rice salt taste at least hour bring pot water boil add salt taste add drain rice once boil cook minute rice cook gently scoop rice bottom pot bring surface release step several_time place colander sink want choose colander hole small rice wo_nt escape check rice make sure cook soft cook mushy drain rice colander rinse cold water stop cooking process pick barberry remove stone find then soak water minute rinse barberry soak skillet melt’,

- 'cook rice saucepan place cup cold water cup bring boil reduce heat cover simmer minute lift lid cook prepare seasoning filling rice cook aim ingredient prepare place separate plate bowl ready roll rice cook cool small bowl mix together tablespoon rice vinegar tablespoon sugar teaspoon salt use season rice slightly large bowl mix cup water tablespoon rice vinegar use keep hand equipment moisten thus keep rice stick make roll sprinkle large tray platter water rice vinegar solution use cool rice peel slice avocado squeeze juice lemon slice prevent brown slice carrot cucumber matchstick slice mushroom season cool rice rice cook add sugar vinegar salt mixture saucepan rice gently stir buy special rice paddle purpose spoon fork suffice just careful break grain place rice moistened tray platter spread take magazine piece cardboard similar fan rice turn rice occasionally continue fan rice cool room_temperature make roll position bamboo roll mat roll away place sheet nori mat side dip hand water vinegar solution take ball rice place nori spread rice pat nori close body rice thick edge rice see roll finish nori rice section seal roll centre rice place filling now start roll lift edge mat close’,

- 'place rice rice cooker need water rinse rice first pour cup cold water right top rice separate container dissolve bouillon cup warm water bouillon completely dissolve mix cooker rest water rice put rice cooker cook do'

Latent Dirichlet Allocation

- LDA makes two key assumptions:
- Documents are a mixture of topics, and
- Topics are a mixture of tokens (or words)



Another view.

Document Term Matrix (DTM)

	W1	W2	W3	W4	W5	W6	W7	W8
D1	0	1	1	0	1	1	0	1
D2	1	1	1	1	0	1	1	0
D3	1	0	0	0	1	0	0	1
D4	1	1	0	1	0	0	1	0
D5	0	1	0	1	0	0	1	0

Shape: 5 * 8

Document Topic Matrix

	K1	K2	K3	K4	K5	K6
D1	1	0	0	0	0	0
D2	0	1	0	0	1	1
D3	1	1	0	0	0	0
D4	1	0	0	1	0	1
D5	0	0	1	1	0	0

Shape: 5 * 6

Topic Word Matrix

	W1	W2	W3	W4	W5	W6	W7	W8
K1	0	1	1	0	1	0	1	0
K2	1	1	1	1	0	1	1	1
K3	1	0	0	0	0	1	0	0
K4	1	1	0	1	1	0	0	1
K5	0	0	1	1	0	1	1	1
K6	1	0	1	1	1	0	0	1

Shape: 6 * 8

Text Preprocessing

- Removing special characters.
- Cleaning numbers.
- Removing misspellings
- Peter Norvig's spell checker. https://norvig.com/spell-correct.html
- Using Google word2vec vocabulary to identify misspelled words. https://mlwhiz.com/blog/2019/01/17/deeplearning_nlp_preprocess/

- Removing contracted words --- contraction_dict = {"ain't": "is not", "aren't": "are not","can't": "cannot", …}

Topic Modeling Dataset and Notebook

Dataset : https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions

Notebook: https://github.com/sangha123/Intro-to-NLP-Topic-Modeling-and-Text-Categorization/blob/main/notebooks/topic_modeling_recipes.ipynb

Requirements

```
import pandas as pd
import numpy as np
import json
import pickle
```

```
import re
from pprint import pprint

# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# spacy for lemmatization
import spacy
from spacy.lang.en import English
# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
import matplotlib.pyplot as plt
%matplotlib inline

###
import nltk
#nltk.download('stopwords')
from nltk.corpus import stopwords

### models
from gensim.models import Phrases
from gensim.models.phrases import Phraser
from gensim import models
```

```
df_recipes = pd.read_csv('../data/RAW_recipes.csv')
```

```
df_recipes.head()
```

	name	id	minutes	contributor_id	submitted	tags	nutrition	n_steps	steps	description	ingredients
0	arriba baked winter squash mexican style	137739	55	47892	2005-09-16	['60-minutes-or-less', 'time-to-make', 'course...	[51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]	11	['make a choice and proceed with recipe', 'dep...	autumn is my favorite time of year to cook! th...	['winter squash', 'mexican seasoning', 'mixed ...
1	a bit different breakfast pizza	31490	30	26278	2002-06-17	['30-minutes-or-less', 'time-to-make', 'course...	[173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0]	9	['preheat oven to 425 degrees f', 'press dough...	this recipe calls for the crust to be prebaked...	['prepared pizza crust', 'sausage patty', 'egg...
2	all in the kitchen chili	112140	130	196586	2005-02-25	['time-to-make', 'course', 'preparation', 'mai...	[269.8, 22.0, 32.0, 48.0, 39.0, 27.0, 5.0]	6	['brown ground beef in large pot', 'add choppe...	this modified version of 'mom's' chili was a h...	['ground beef', 'yellow onions', 'diced tomato...
3	alouette potatoes	59389	45	68585	2003-04-14	['60-minutes-or-less', 'time-to-make', 'course...	[368.1, 17.0, 10.0, 2.0, 14.0, 8.0, 20.0]	11	['place potatoes in a large pot of lightly sal...	this is a super easy, great tasting, make ahea...	['spreadable cheese with garlic and herbs', 'n...

```
#### text preprocessing


stop_words = stopwords.words('english')
stop_words.extend(['from', 'to','in','a','the', 'and'])

nlp = spacy.load("en_core_web_sm")
config = {"punct_chars": [".","?"]}
nlp.add_pipe("sentencizer", config=config)

def sent_to_words(text):

    docs = nlp(text)
    sentences = [token.sent for token in docs.sents]
    words = [[token.text for token in x] for x in sentences]
    return words

def remove_stopwords(words):
    return [x for x in words if x not in stop_words]

def process_text(text):

    words = sent_to_words(text)
    words_clean = remove_stopwords(words)
    return words_clean
```

```
steps_processed = [process_text(".".join(x)) for x in steps]
```

Bigram & Trigram

```
# higher threshold fewer phrases.
bigram = gensim.models.Phrases(steps_processed, min_count=5, threshold=100)
trigram = gensim.models.Phrases(steps_processed, threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)
```

```
def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    """https://spacy.io/api/annotation"""
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out
```

```
data_words_bigrams = make_bigrams(steps_processed)
# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = lemmatization(data_words_bigrams, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])
```

Stemming vs Lemmatization

```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:1])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10, 1), (1
1, 1), (12, 1), (13, 2), (14, 1), (15, 1), (16, 1), (17, 1), (18, 1), (19, 1), (20, 1), (21,
1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1)]]
```

Doc2bow --- Converts *document* into the bag-of-words (BoW) format = list of *(token_id, token_count)* tuples.

```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=30,
                                            random_state=100,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
```

```
# Compute Perplexity
# a measure of how good the model is. lower the better.

print('\nPerplexity: ', lda_model.log_perplexity(corpus))

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, \
                                     texts=data_lemmatized, \
                                     dictionary=id2word, \
                                     coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -10.614173278238972

Coherence Score:  0.444294803807023
```

```
def lda_to_df(model,corpus):
    '''This function takes a gensim lda model as input, and outputs a df with topics probs by document'''
    topic_probs = model.get_document_topics(corpus) #get the list of topic probabilities by doc
    topic_dict = [dict(x) for x in topic_probs] #convert to dictionary to convert to data frame
    df = pd.DataFrame(topic_dict).fillna(0) #convert to data frame
    df['docs'] = df.index.values #create column with document indices (correspond to indices of dataframe)
    df.columns = df.columns.astype(str) #convert to string to make indexing easier
    return df

def get_best_docs(df, n, k, texts):
    '''Return the index of the n most representative documents from a list of topic responsibilities for each topic'''
    '''n is the number of douments you want, k is the number of topics in the model, the texts are the FULL texts used
    #create column list to iterate over
    k_cols = range(0, k)

    #intialize empty list to hold results
    n_rep_docs_dict = defaultdict(list)

    #loop to extract documents for each topic
    for i in k_cols:
        if str(i) in df.columns:
            inds = df.nlargest(n = n, columns = str(i))['docs'].astype(int).tolist()
            #use list comprehension to extract documents
            n_rep_docs_dict[i] +=[texts[ind] for ind in inds]

    return n_rep_docs_dict
```

df_topics_docs = lda_to_df(lda_model,corpus)
docs_dict = get_best_docs(df_topics_docs, 5, 100, texts)

```
[" ".join(x) for x in docs_dict[16]]
```

```
['ratio rice water quantity rice increase cup rice cup water cup rice cup water method put water boil meanwhile wash
rice time water start boil add rice salt check min see rice cook grain still firm raw break grain finger check drain
cover fitting flat lid invert sink then leave inverted vessel good minute make sure water drain also drain colander d
rain pasta straighten',
 'large saucepan bring salt water boil simmer lentil low minute tender drain water lentil set aside pan saute onion g
arlic oil add raisin date spice mix set aside cook rice like sure fire way use rice cooker rinse rice cold water wate
r runs_clear add rice cooker cover cup water add teaspoon salt drizzle oil cook rice do transfer cooked rice large bo
wl same pot add oil just cover bottom surface add potato slice add layer rice add layer lentil raisin mixture continu
e layer end final layer rice cover cook minute drizzle melt butter saffron water rice cover top pot rice cooker tea t
owel prevent steam escape top cook low minute want make potato crispy so cook slowly rice cooker continuously hit coo
k button process happen totally worth',
```

```
for i in range(0, lda_model.num_topics-1):
    print(lda_model.print_topic(i))
    print("topic %s"%str(i))
```

```
0.281*"be" + 0.213*"favorite" + 0.149*"equal" + 0.110*"gentle" + 0.058*"peppercorn" + 0.043*"crab" + 0.041*"left" +
0.000*"forum" + 0.000*"doughy" + 0.000*"obscure"
topic 0
0.499*"oven" + 0.254*"preheat" + 0.183*"degree" + 0.040*"breast" + 0.015*"pizza" + 0.000*"doughy" + 0.000*"adequatel
y" + 0.000*"forum" + 0.000*"obscure" + 0.000*"vert"
topic 1
0.419*"remain" + 0.159*"transfer" + 0.136*"meanwhile" + 0.069*"scoop" + 0.058*"over" + 0.034*"rolling_pin" + 0.028*"h
eated" + 0.019*"firmly" + 0.012*"puff" + 0.012*"applesauce"
topic 2
0.209*"sugar" + 0.173*"flour" + 0.097*"beat" + 0.079*"bake" + 0.058*"powder" + 0.055*"vanilla" + 0.050*"batter" + 0.0
27*"light" + 0.024*"speed" + 0.023*"soda"
topic 3
0.000*"obscure" + 0.000*"grease"ooze" + 0.000*"arugula" + 0.000*"haricot" + 0.000*"vert" + 0.000*"adequately" + 0.000
*"doughy" + 0.000*"forum" + 0.000*"pic" + 0.000*"gazpacho"
```

Text Categorization

Dataset

```
#### data I/O
import pandas as pd
import numpy as np
import json
```

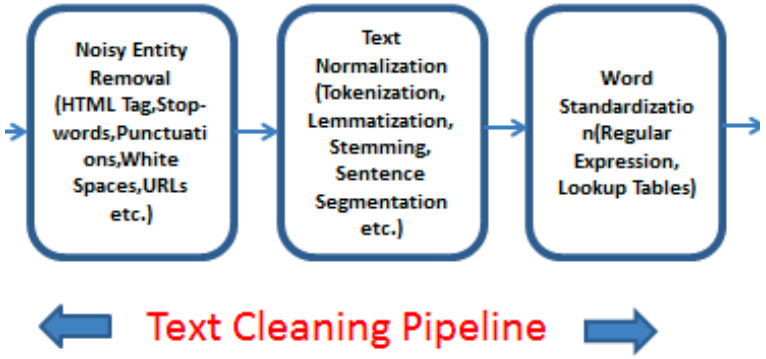
```
df_spam_0 = pd.read_csv('../data/spam_ham_dataset.csv')
```

```
df_spam_0.head()
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

Text Classification

Text Pre - processing



- Reduces noise
- Ensures quality
- Improves overall performance

Collecting Training Data

SME

- Training Data Collection / Examples of classes that we are trying to model
- Model performance is directly correlated with quality of training data
- Use rule based systems to generate training data.

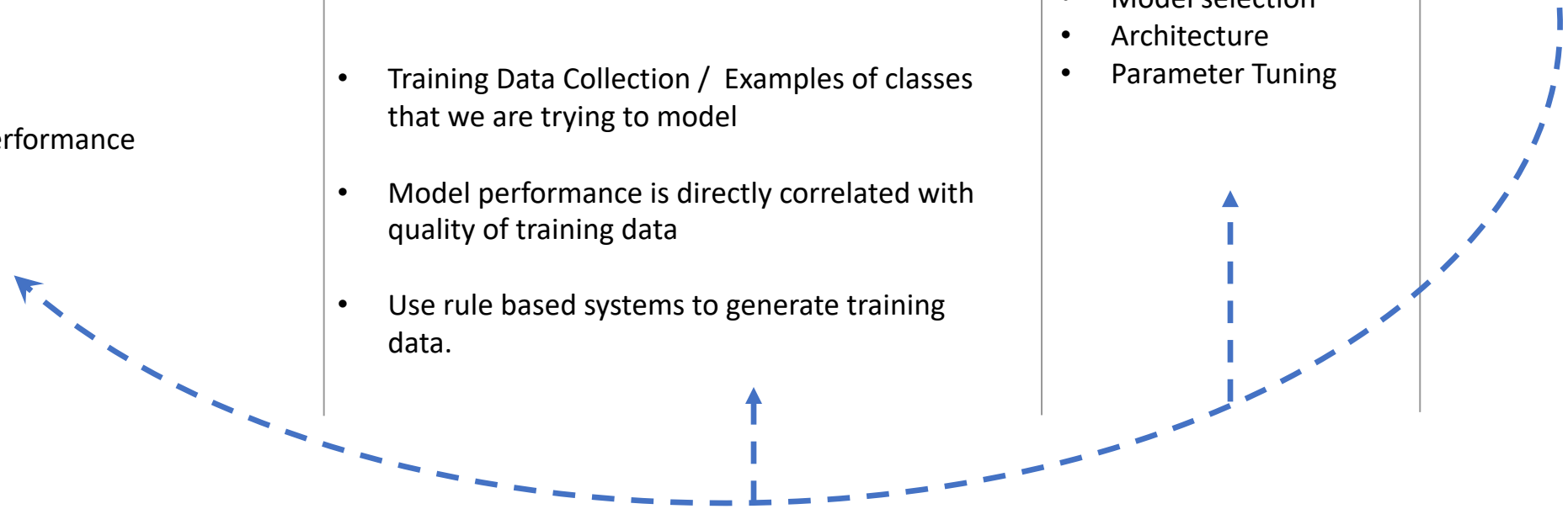
Model Building

- Model selection
- Architecture
- Parameter Tuning

Model Evaluation

Offline

Online



Data centric vs Model Centric AI

Data beats models

- Better models improve performance by tenths of a percentage point
- Better data improves performance by tens of percentage points

We no longer really understand our models

- In DNNs, domain knowledge is encoded in the data
- Previously, much was encoded in features, model architecture, rules

Data is not gospel-truth

- Contains both noise & systemic flaws:
  spurious correlations, unobserved variables, domain shift

Data centric approach

Trying to improve the data while keeping the model same

Model centric approach

Trying to improve the code while keeping the data same

Weak Supervision : Rule Based data generation

Using noisy sources of truth to generate training data.

Constraints: “Entites will always occur more than once in your dataset”
“Only 10% of the data can belong to this class”

Using **cheaper sources of labels**.

- Non experts
- crowdsourcing

Heuristics: ”a company name typically ends with an inc”

Pretrained model:

- train a high level model and then impose rules on the output probability of the classifier.
- Topic modeling

Existing knowledgebase: In spam detection known emails coming from known Ips.

Data --- Train, Test and Validation

- *Train – 90% , Test –5 %, Validation –5% ---- the percentages can vary depending on the total size of the dataset.*

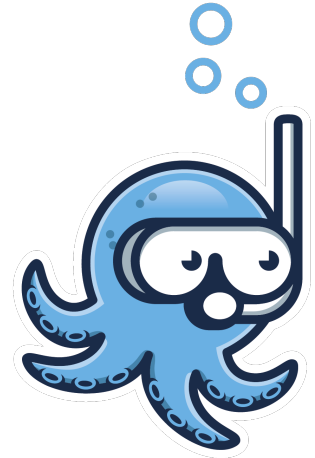
- Train --- data used to train the model

- Validation --- data that the model has not seen but is used for parameter tuning, i.e the model is optimized based on performance on this set.

- Test --- model has not seen this data, this data is not used in any part of the computation. Final performance metrics are reported on this data.

Dataset: https://www.kaggle.com/datasets/venky73/spam-mails-dataset

Rule based Learning using snorkel



snorkel

Snorkel is a data programing paradigm that started at Stanford in 2016.
It helps programmatically label, build, and manage training data.

There is an open source version of the snorkel.
https://www.snorkel.org/

It is part of DAWN Labs.
https://dawn.cs.stanford.edu/projects/

The creators of snorkel founded a company that provides a platform for this service.

https://snorkel.ai/platform/

STANFORD
DAWN

Setting up your Directory

```
git clone https://github.com/sangha123/Intro-to-NLP-Topic-Modeling-and-Text-Categorization.git
cd Intro-to-NLP-Topic-Modeling-and-Text-Categorization.git
git clone https://github.com/snorkel-team/snorkel-tutorials.git
cd snorkel-tutorials
mv ../notebooks/01_spam_tutorial_sdeb.ipynb .
```

Writing rules or Labeling functions

Labeling functions (LFs) help users encode domain knowledge and other supervision sources programmatically.

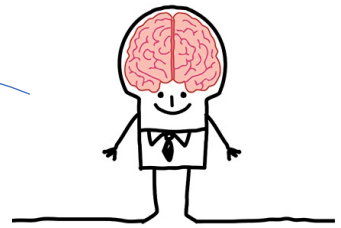
Keyword searches: looking for specific words in a sentence
Pattern matching: looking for specific syntactical patterns
Third-party models: using an pre-trained model (usually a model for a different task than the one at hand)
Distant supervision: using external knowledge base
Crowdworker labels: treating each crowdworker as a black-box function that assigns labels to subsets of the data

```
@labeling_function()
def check(x):
    return SPAM if "check" in x.text.lower() else ABSTAIN


@labeling_function()
def check_out(x):
    return SPAM if "check out" in x.text.lower() else ABSTAIN
```

LF Iterations

SME



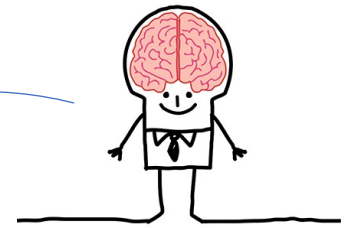
Knowledge transfer

What is important to product and business

Product/Businesss

Iterate

Language, business needs and teams evolve.



Knowledge transfer

Data Scientist

What is important to product and business

Product/Businesss

- Create Filters
- Create rules

Replaces manual generation of labelled data

- Redefine Filters
- Redefine rules

Metrics to determine performance for LFs

- **Polarity**: The set of unique labels this LF outputs (excluding abstains)
- **Coverage**: The fraction of the dataset the LF labels
- **Overlaps**: The fraction of the dataset where this LF and at least one other LF label
- **Conflicts**: The fraction of the dataset where this LF and at least one other LF label and disagree
- **Correct**: The number of data points this LF labels correctly (if gold labels are provided)
- **Incorrect**: The number of data points this LF labels incorrectly (if gold labels are provided)
- **Empirical Accuracy**: The empirical accuracy of this LF (if gold labels are provided)

LF performance evolution

Rules for You tube comments

```
Majority Vote Accuracy:    42.8%
Label Model Accuracy:      37.4%
```

Customized rules from topic modeling

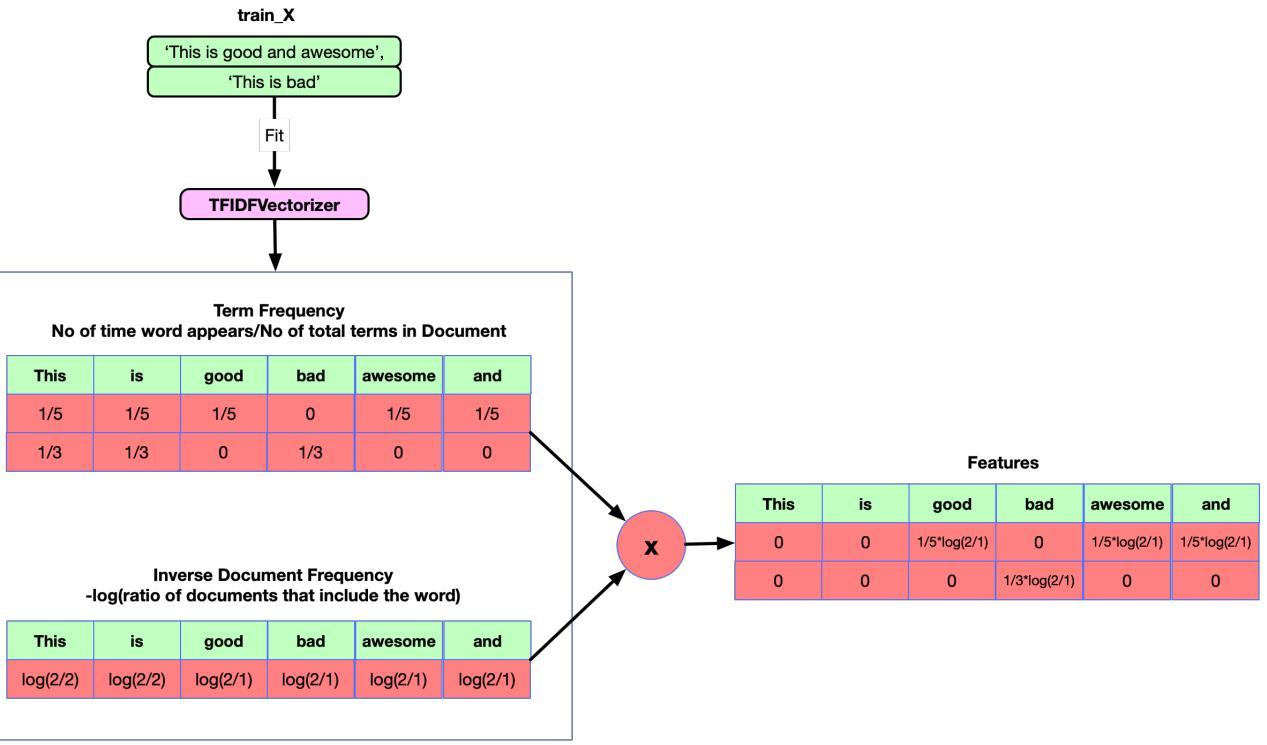
Spam --- 'phaermacy',
'harmacie',
'pill',
'overnite delivery',
'overnight delivery'

Ham --- 'enron', 'buyback', 'out of the office', 'goodbye', 'mandate', 'getting back', 'spreadsheet'

```
Majority Vote Accuracy:    56.0%
Label Model Accuracy:      58.8%
```

TFIDF Features

- **ngram_range:** (1,3) --- implies unigrams, bigrams, and trigrams will be taken into account while creating features.
- **min_df:** Minimum no of time an ngram should appear in a corpus to be used as a feature.

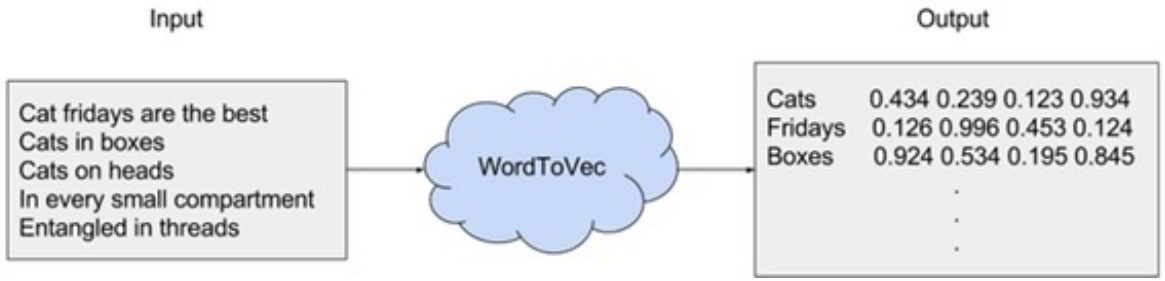



```
# Always start with these features. They work (almost) everytime!
tfv = TfidfVectorizer(dtype=np.float32, min_df=3,  max_features=None,
            strip_accents='unicode', analyzer='word',token_pattern=r'\w{1,}',
            ngram_range=(1, 3), use_idf=1,smooth_idf=1,sublinear_tf=1,
            stop_words = 'english')

# Fitting TF-IDF to both training and test sets (semi-supervised learning)
tfv.fit(list(train_df.cleaned_text.values) + list(test_df.cleaned_text.values))
xtrain_tfv =  tfv.transform(train_df.cleaned_text.values)
xvalid_tfv = tfv.transform(test_df.cleaned_text.values)
```

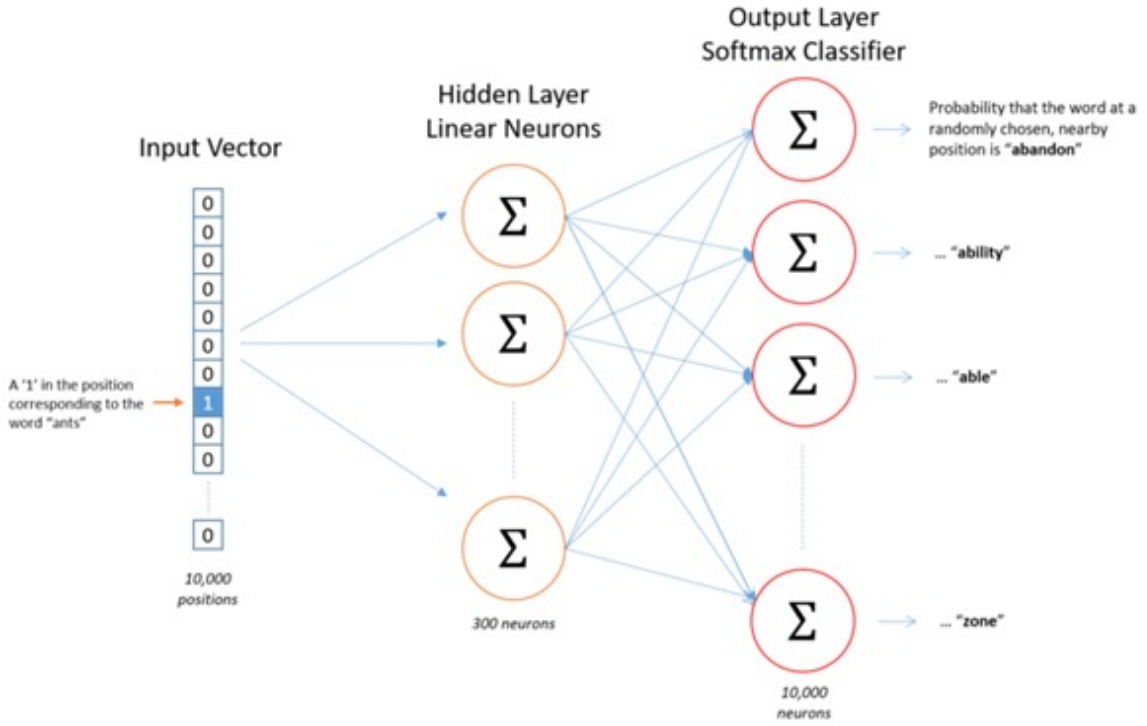
Tfidf features can be used with any ML classifier such as LR
When using LR for NLP tasks L1 regularization performs better since tfidf features are sparse.

Transfer Learning – word2vec features



https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1

either using context to predict a target word (a method known as continuous bag of words, or CBOW), or using a word to predict a target context, which is called skip-gram



Applying tfidf weighting to word vectors boosts overall model performance
https://towardsdatascience.com/supercharging-word-vectors-be80ee5513d

Text Pre-processing with Keras

Tokenizing

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(df_spam_0.text)
```

```
vocab_size = len(tokenizer.word_index) + 1
```

```
X_train = tokenizer.texts_to_sequences(df_train.text.values)
X_val = tokenizer.texts_to_sequences(df_val.text.values)
X_test = tokenizer.texts_to_sequences(df_test.text.values)
```

```
print(df_train.text.values[2])
print(X_train[2])
```

```
Subject: hpl nom for july 27 , 2000
( see attached file : hplo 727 . xls )
- hplo 727 . xls
[9, 48, 148, 6, 208, 250, 92, 93, 98, 169, 590, 121, 1, 590, 121]
```

Padding

Python >>>

```
>>> from keras.preprocessing.sequence import pad_sequences

>>> maxlen = 100

>>> X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
>>> X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)

>>> print(X_train[0, :])
[  1  10   3 282 739  25   8 208  30  64 459 230  13   1 124   5 231   8
  58   5  67   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
```

Start with an Embedding Layer

- Embedding Layer of Keras which takes the previously calculated integers and maps them to a dense vector of the embedding.
  - **Parameters**
    - **input_dim:** the size of the vocabulary
    - **output_dim:** the size of the dense vector
    - **input_length:** the length of the sequence

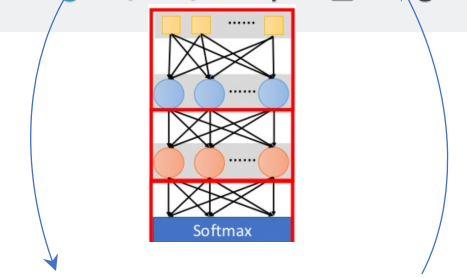
Hope to see you soon
Nice to see you again

```
[0, 1, 2, 3, 4]

[5, 1, 2, 3, 6]
```

https://stats.stackexchange.com/questions/270546/how-does-keras-embedding-layer-work

```
Embedding(7, 2, input_length=5)
```



After training

index	Embedding
0	[1.2, 3.1]
1	[0.1, 4.2]
2	[1.0, 3.1]
3	[0.3, 2.1]
4	[2.2, 1.4]
5	[0.7, 1.7]
6	[4.1, 2.0]

Add a pooling layer

- MaxPooling1D/AveragePooling1D or a GlobalMaxPooling1D/GlobalAveragePooling1D layer
- Way to downsample (a way to reduce the size of) the incoming feature vectors.
- Global max/average pooling takes the maximum/average of all features whereas in the other case you have to define the pool size.

Measuring performance per epoch.

```
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]
```

Definition of the entire model

```
class_weight = {0: 1.,
                1: 2.5
                }


from keras.models import Sequential
from keras import layers

embedding_dim = 50

model = Sequential()
model.add(layers.Embedding(input_dim=vocab_size,
                           output_dim=embedding_dim,
                           input_length=maxlen))
model.add(layers.Flatten())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=METRICS)
model.summary()
```

Fitting the model

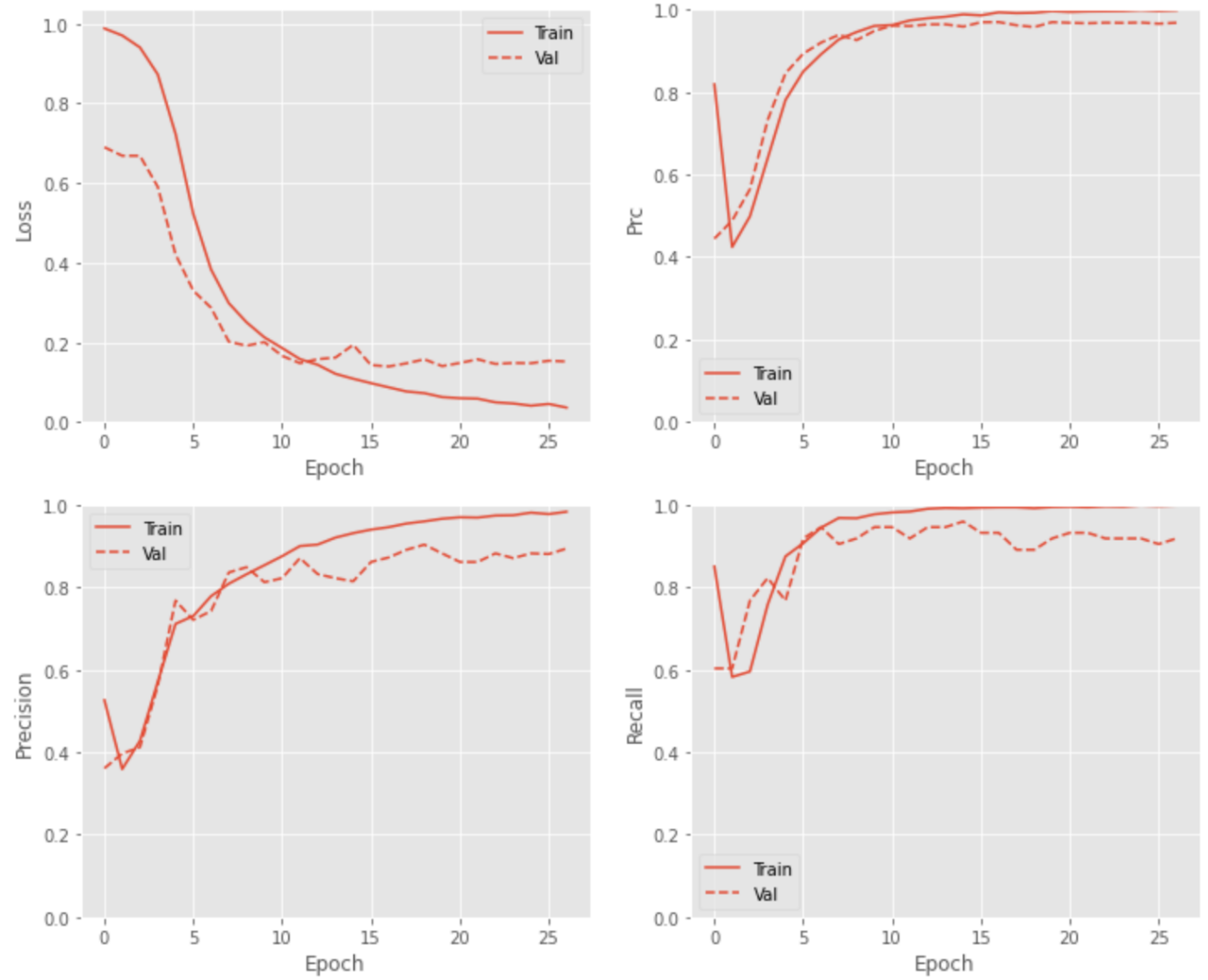
```
early_stopping = tf.keras.callbacks.EarlyStopping(
    monitor='val_prc',
    verbose=1,
    patience=10,
    mode='max',
    restore_best_weights=True)
```

```
history = model.fit(X_train, y_train,
                    epochs=100,
                    verbose=False,
                    validation_data=(X_val, y_val),
                    batch_size=10,
                   class_weight=class_weight,
                    callbacks=[early_stopping]
                   )
```

https://realpython.com/python-keras-text-classification

Performance

	precision	recall	f1-score	support
0	0.95	0.98	0.96	173
1	0.94	0.88	0.91	77
accuracy			0.95	250
macro avg	0.95	0.93	0.94	250
weighted avg	0.95	0.95	0.95	250

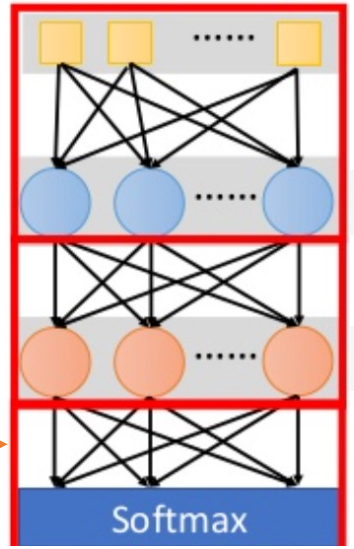


Convolution Neural Network

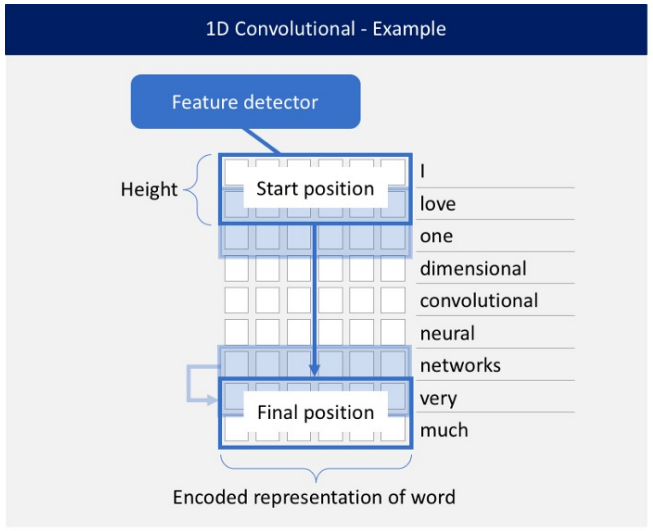
Detect features ! Downsample.

What is a CNN?

In a traditional feedforward neural network we connect each input neuron to each output neuron in the next layer. That's also called a fully connected layer, or affine layer.




```
model = Sequential()
model.add(layers.Dense(10, input_dim=in
model.add(layers.Dense(10, input_dim=in
model.add(layers.Dense(1, activation='
```



In this example for natural language processing, a sentence is made up of 9 words. Each word is a vector that represents a word as a low dimensional representation. The feature detector will always cover the whole word. The height determines how many words are considered when training the feature detector. In our example, the height is two. In this example the feature detector will iterate through the data 8 times.

- We use convolutions over the input layer to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters and combines the result
- During the training phase, **a CNN automatically learns the values of its filters** based on the task you want to perform.
- Inputs --- n_filters, kernel size (=2)

Model definition

```
def create_model(num_filters, kernel_size, vocab_size, embedding_dim, maxlen):
    model = Sequential()
    model.add(layers.Embedding(vocab_size, embedding_dim, input_length=maxlen))
    model.add(layers.Conv1D(num_filters, kernel_size, activation='relu'))
    model.add(layers.GlobalMaxPooling1D())
    model.add(layers.Dense(10, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=METRICS),

    return model
```

Hyper Parameter Tuning & model fitting

```
param_grid = dict(num_filters=[32, 64, 128],
                  kernel_size=[3, 5, 7],
                  vocab_size=[5000],
                  embedding_dim=[50],
                  maxlen=[100])
```

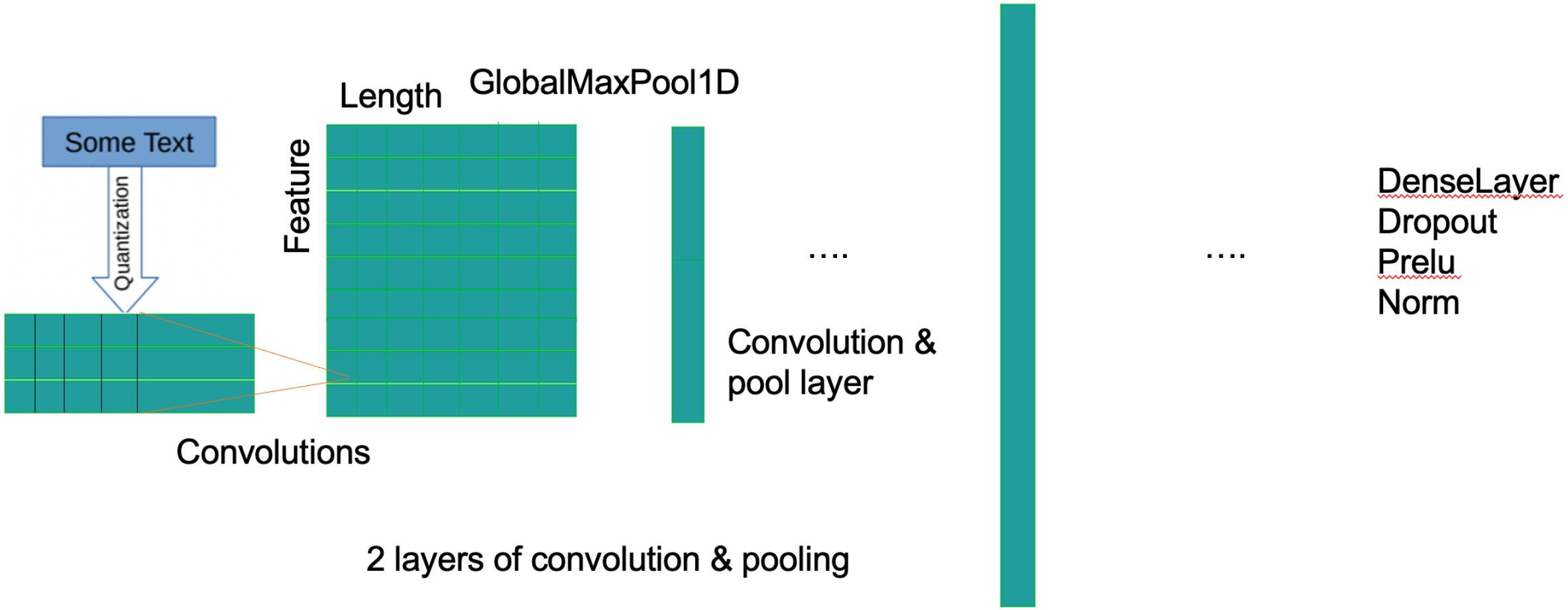
```
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import RandomizedSearchCV
```

```
epochs = 30
```

```
model = KerasClassifier(build_fn=create_model,
                        epochs=epochs, batch_size=10,validation_data=(X_val, y_val),
                        verbose=False,callbacks=[early_stopping])
grid = RandomizedSearchCV(estimator=model, param_distributions=param_grid,
                          cv=4, verbose=1, n_iter=5)
grid_result = grid.fit(X_train, y_train)
```

	precision	recall	f1-score	support
0	0.99	0.98	0.98	180
1	0.94	0.97	0.96	70
accuracy			0.98	250
macro avg	0.97	0.97	0.97	250
weighted avg	0.98	0.98	0.98	250

Character based CNN



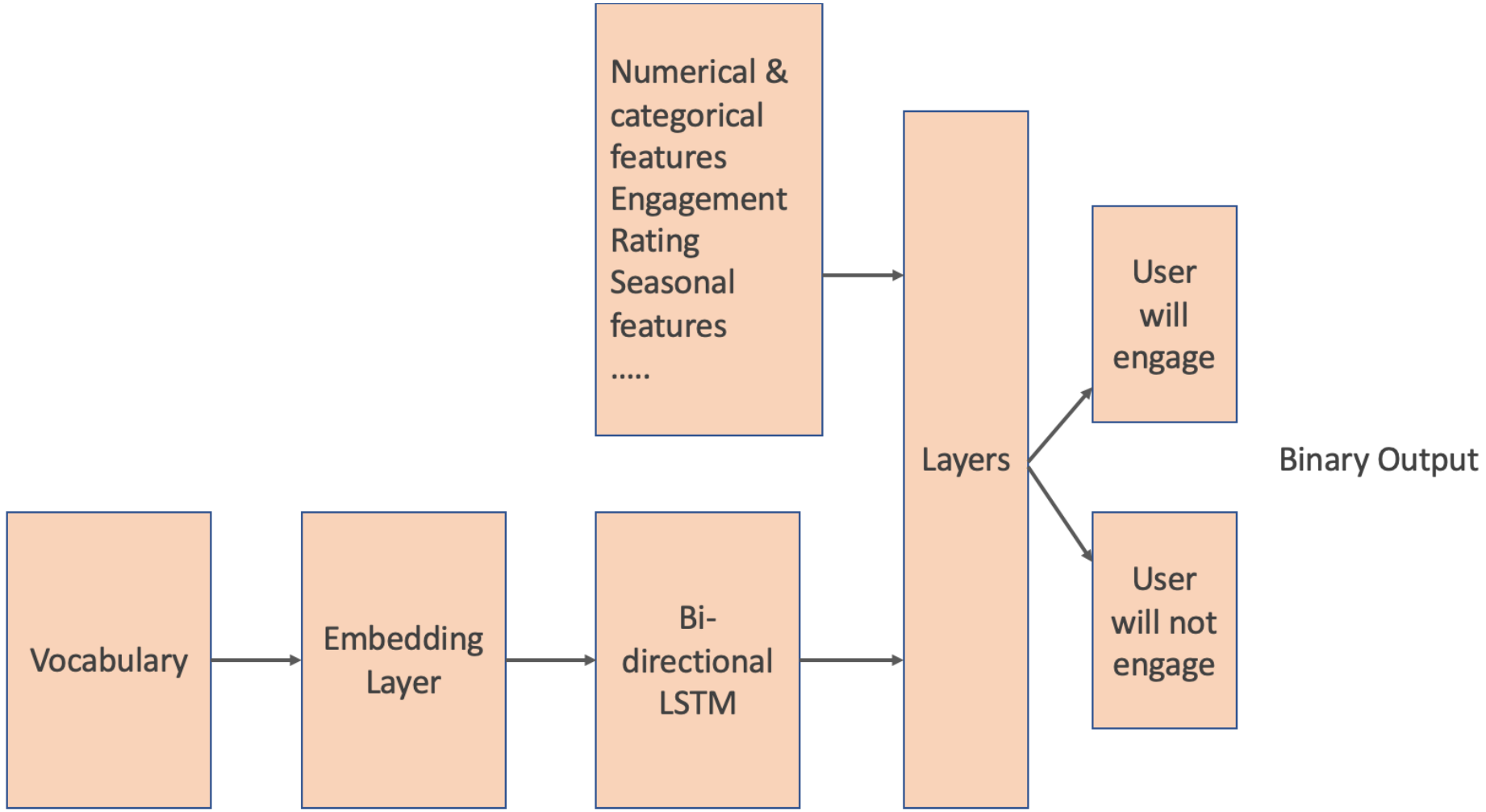
https://towardsdatascience.com/character-level-cnn-with-keras-50391c3adf33

Advantages of CNN

- **Character Based CNN**
  - **Has the ability to deal with out of vocabulary words. This makes it particularly suitable for user generated raw text.**
  - **Works for multiple languages.**
  - **Model size is small since the tokens are limited to the number of characters ~ 70. This makes real life deployments easier and faster.**
  - **Does not need a lot of data cleaning**
- **Networks with convolutional and pooling layers are useful for classification tasks in which we expect to find strong local clues regarding class membership.**

https://machinelearningmastery.com/best-practices-document-classification-deep-learning/

Text features + Structured features.



https://towardsdatascience.com/how-to-combine-textual-and-numerical-features-for-machine-learning-in-python-dc1526ca94d9

Performance Metrics

Measuring performance per epoch.

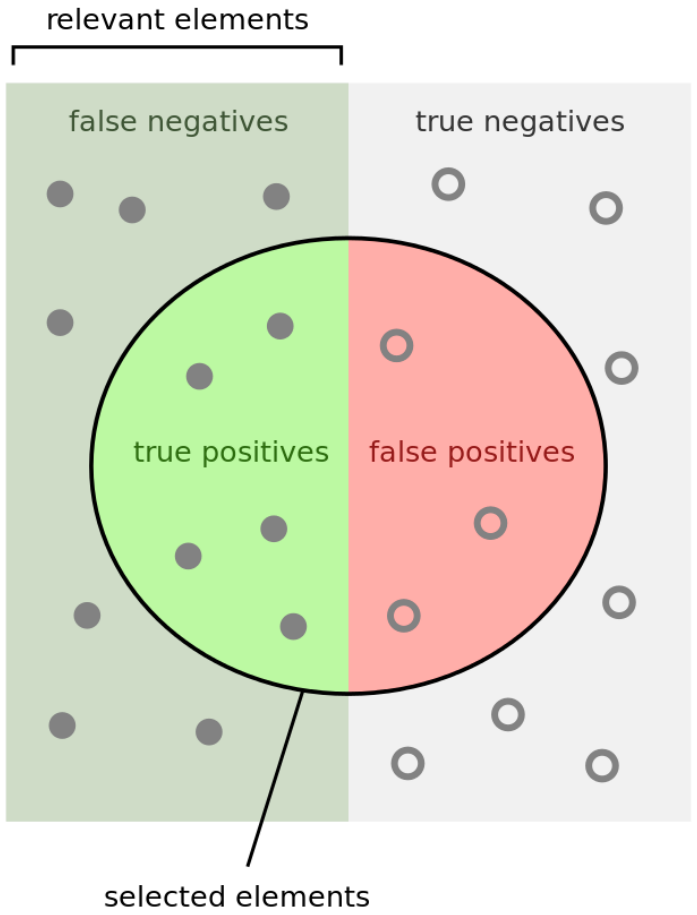
```
METRICS = [
      keras.metrics.TruePositives(name='tp'),
      keras.metrics.FalsePositives(name='fp'),
      keras.metrics.TrueNegatives(name='tn'),
      keras.metrics.FalseNegatives(name='fn'),
keras.metrics.BinaryAccuracy(name='accuracy'),
      keras.metrics.Precision(name='precision'),
      keras.metrics.Recall(name='recall'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'), # precision-recall curve
]
```

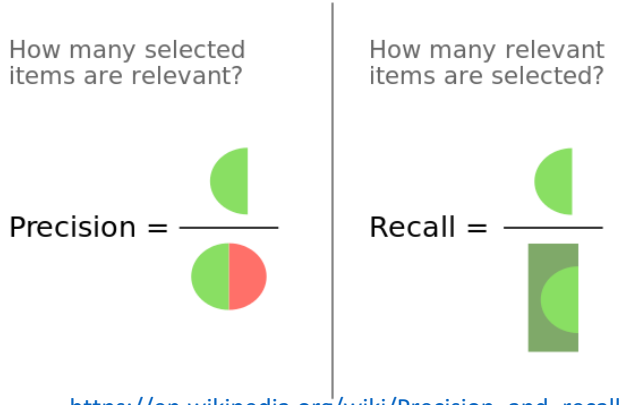
Classification

Precision : TP/(TP+FP) --- what percentage of the positive class is actually positive?

Recall : TP/(TP+FN) --- what percentage of the positive class gets captured by the model?

Accuracy --- (TP+TN)/(TP+FP+TN+FN) --- what percentage of predictions are correct?

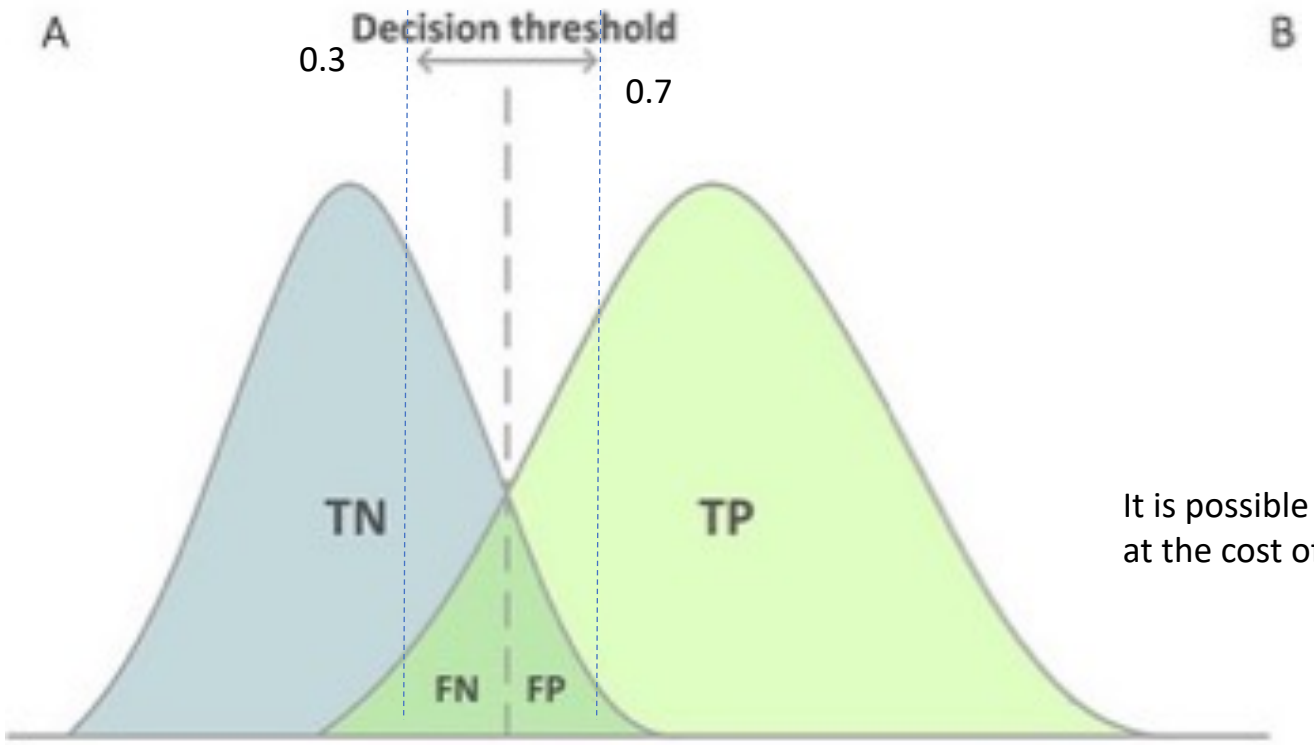




https://en.wikipedia.org/wiki/Precision_and_recall

Thresholding --- Coverage

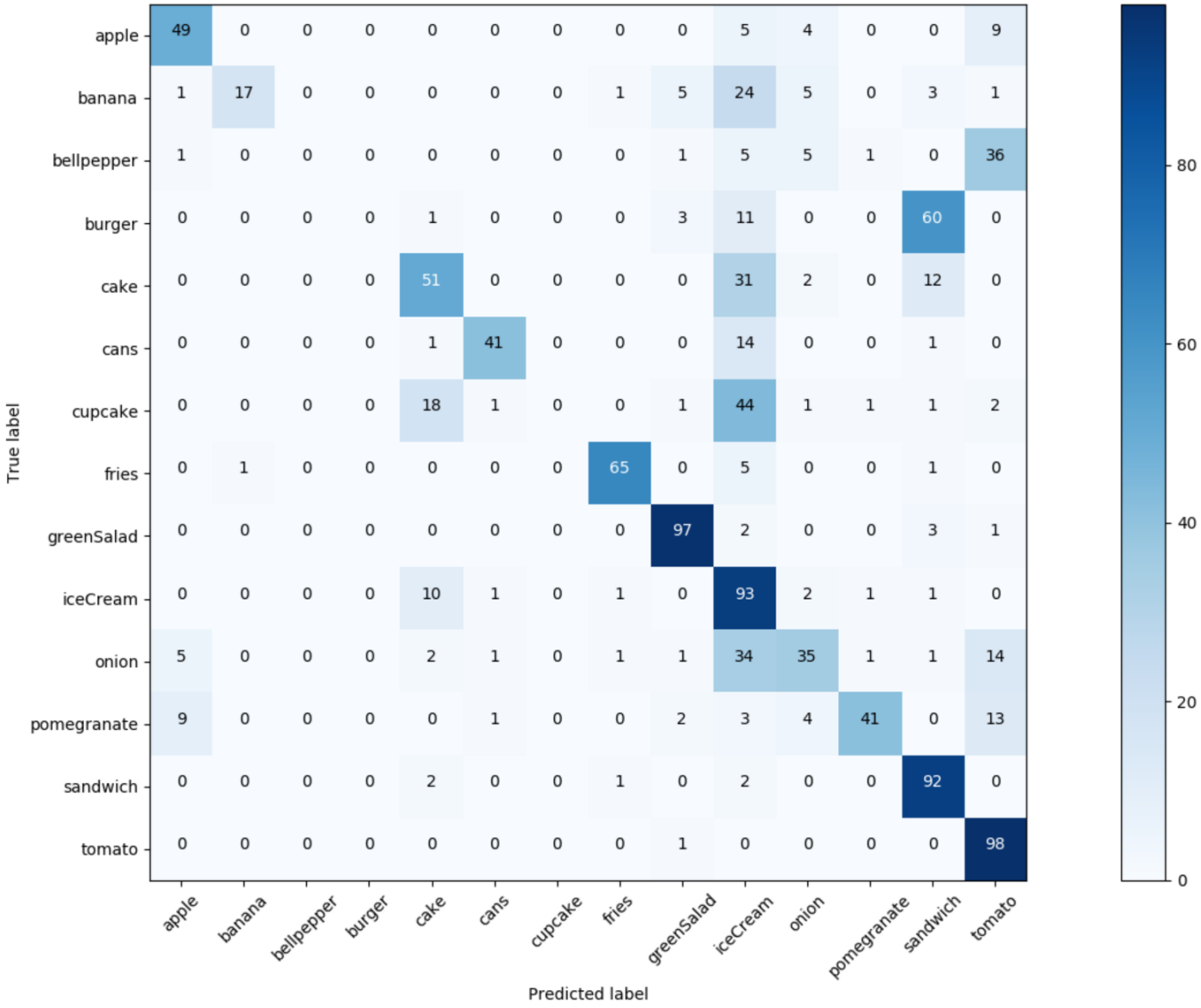
In a binary classification if you choose randomly the probability of belonging to a class is 0.5



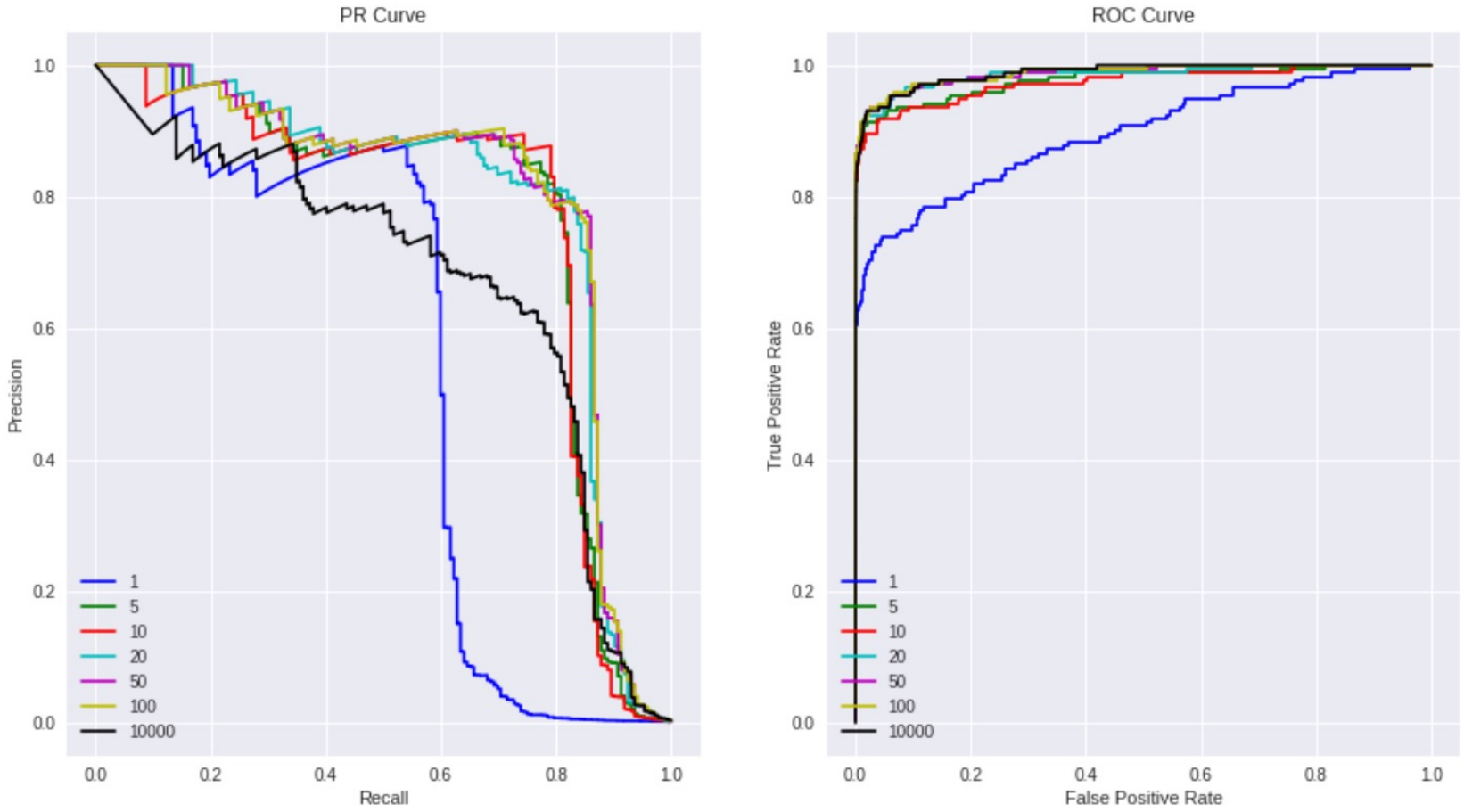
It is possible improve the percentage of correct results at the cost of coverage.

Confusion Matrix

- Good for checking where your model is incorrect
- For multi-class classification it reflects which classes are correlated



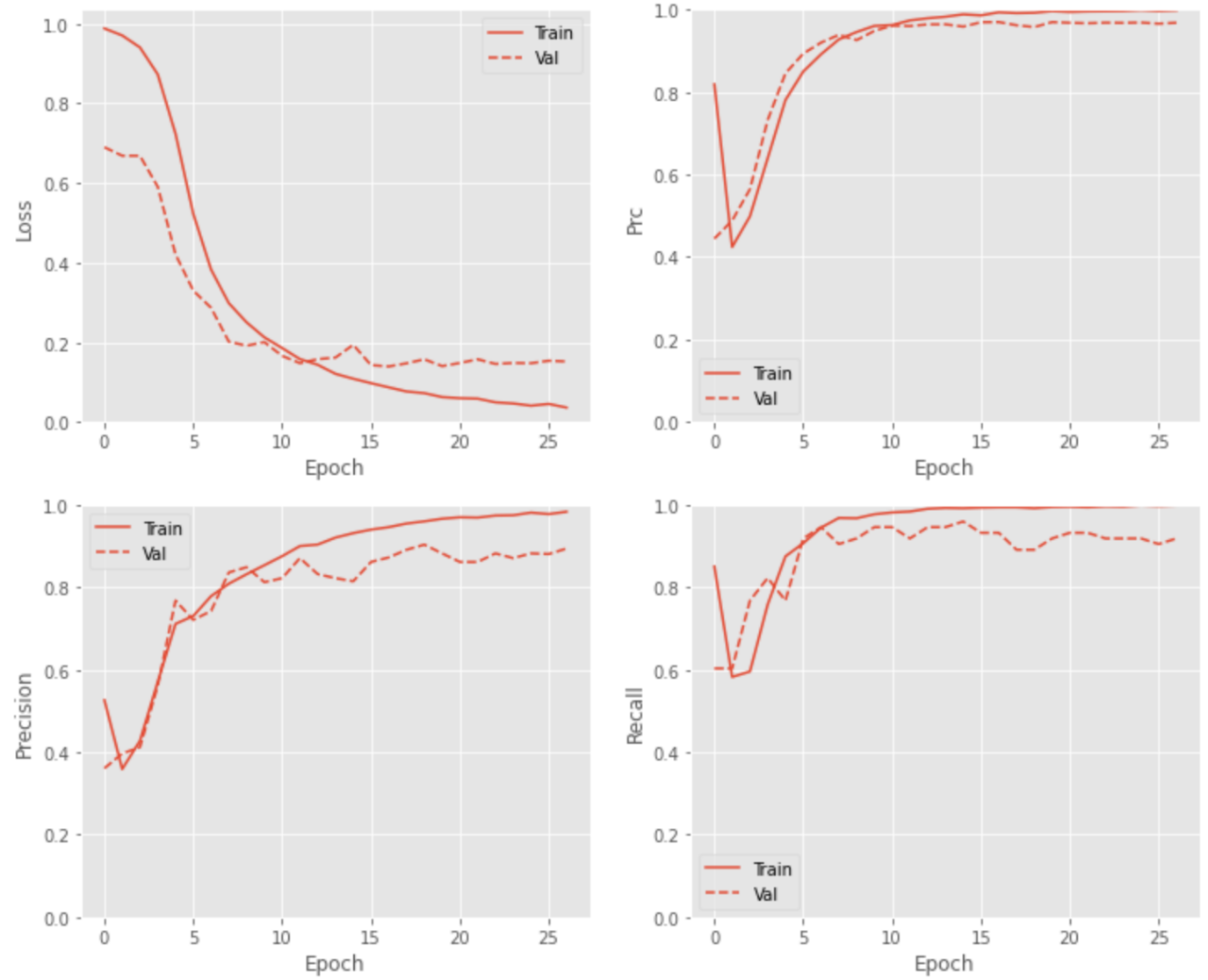
ROC vs Precision Recall Curves



https://www.kaggle.com/code/lct14558/imbalanced-data-why-you-should-not-use-roc-curve/notebook

Performance

	precision	recall	f1-score	support
0	0.95	0.98	0.96	173
1	0.94	0.88	0.91	77
accuracy			0.95	250
macro avg	0.95	0.93	0.94	250
weighted avg	0.95	0.95	0.95	250



Thank You

 @sangha_deb
sangha123@gmail.com