Natural Language Processing

#Regular expression

^ → starts with

$ → ends with

[] → Find within range of characters

\ → Signals a special sequence (numbers, \d word\space etc)
  ↳ can also be used to escape special characters.

. → Any character (except newline)

* → zero or more occurrences

+ → One or more occurrences

? → zero or One occurrences

{} → Exactly the specified number of occurrences

| → Either or

() → capture and group

Use of several functions
findall, search, split, sub, span, group, match, string ... etc

#collected from W3schools.com

Do follow official Documentation of Python.

#Why NLP is booming

- Freely available + Pretrained models
- Open source Ecosystem (spacy, Gensim, NLTK)
- Cheap Hardware, Cloud Resources
- Learning Resources
- Huge investment by Big Tech

#Learn Regular expression (must)

Go to Link: regex101.com to use chatbot for Regex

#"import re" to import re module:

prequisites

1. Python
2. Machine Learning
3. Deep Learning (RNN, CV, LSTM, Word to Vec)

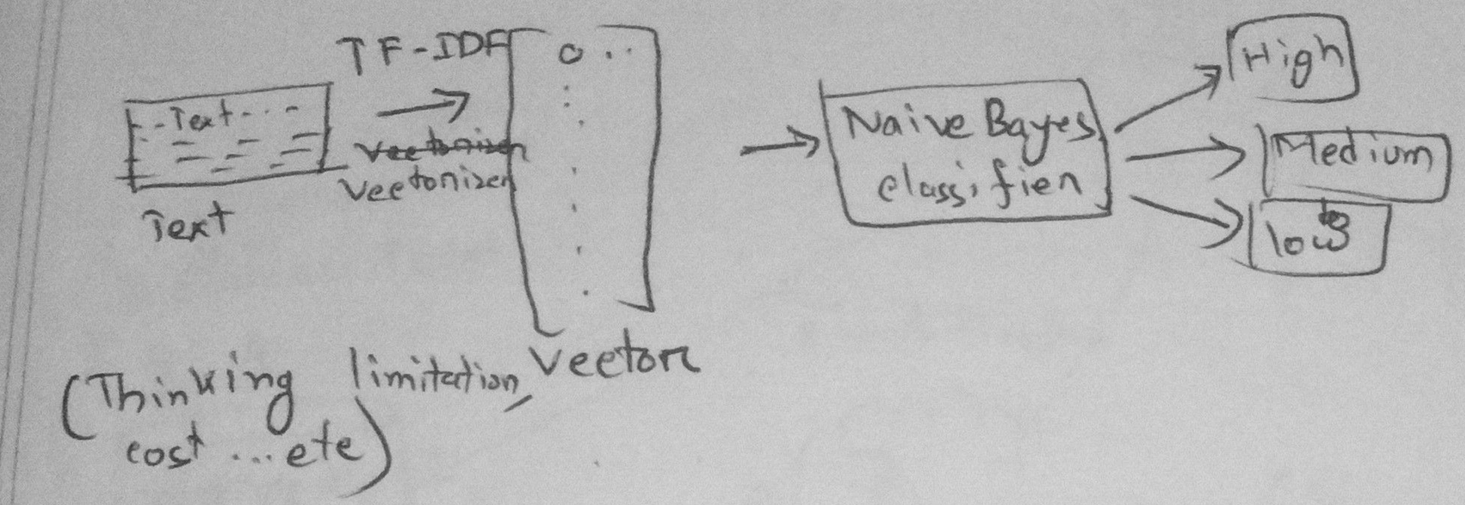
* Information Extraction using Regular Expression

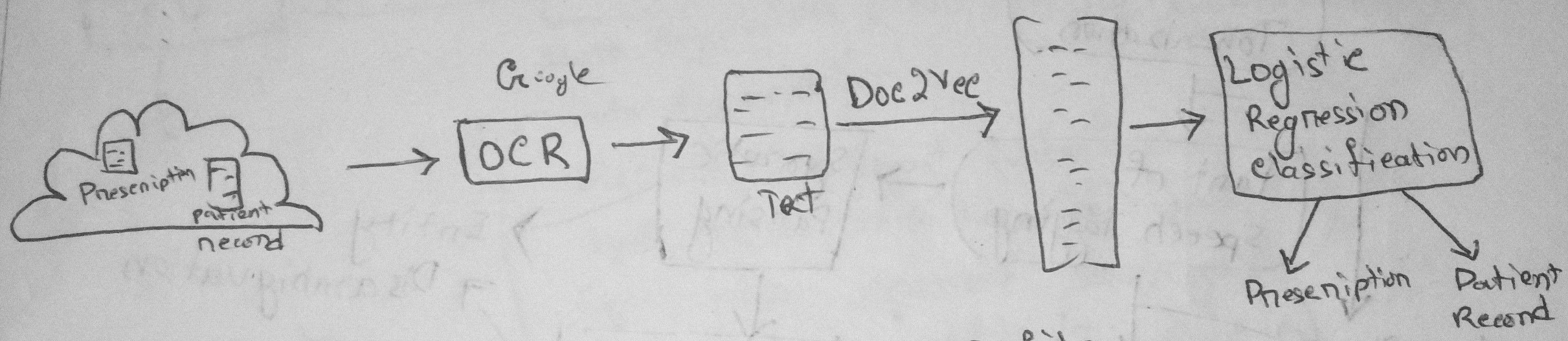
NLP Techniques

1. Rules & Heuristics (Use Regular Expression) → To extract information
2. NLP pipeline (Raw Text → Number Vector → Statistical Machine Learning)
3. Sentence Embedding / Word Embedding (Cosine Similarity based, Using count vectorizing approach)

NLP Tasks

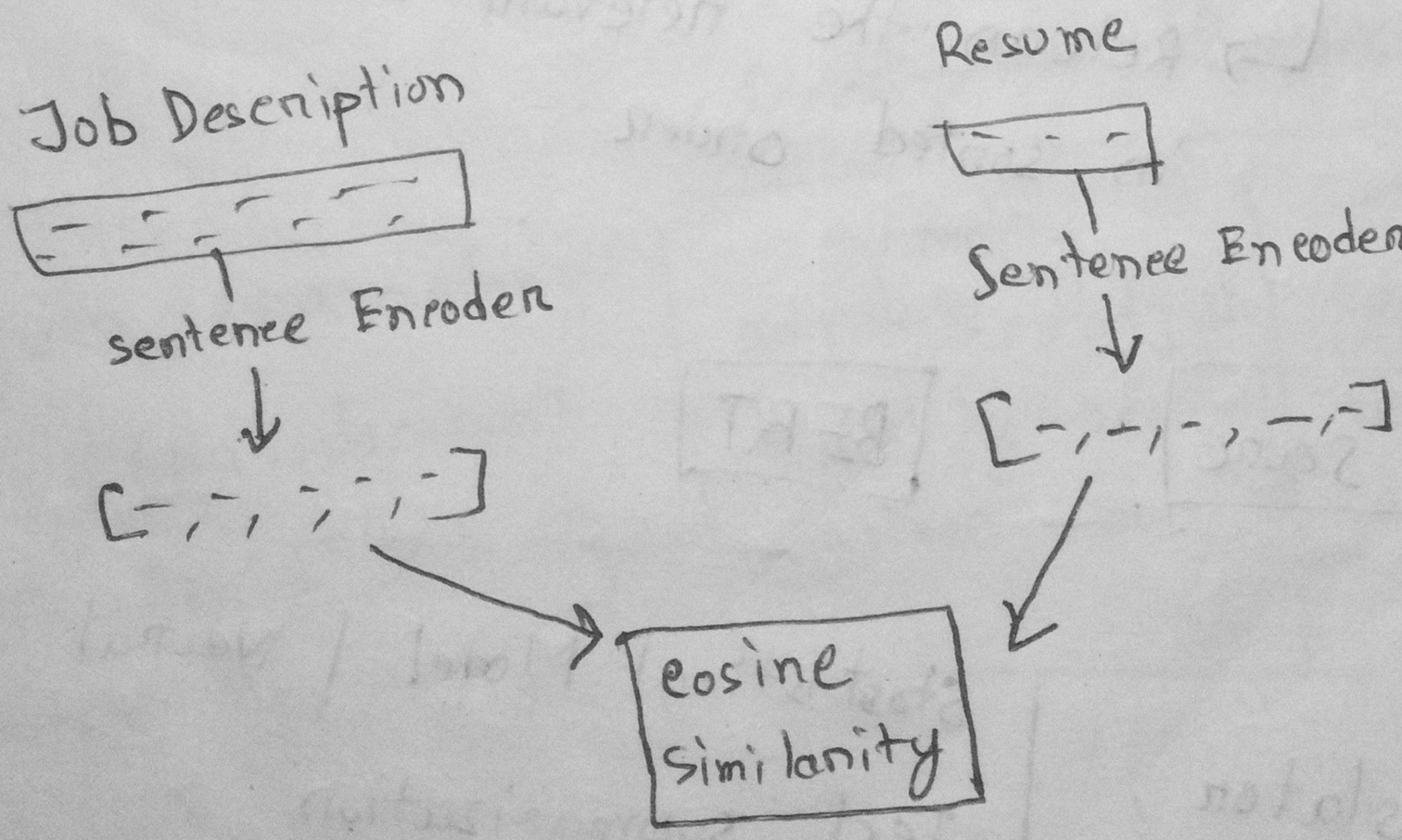
Text classification



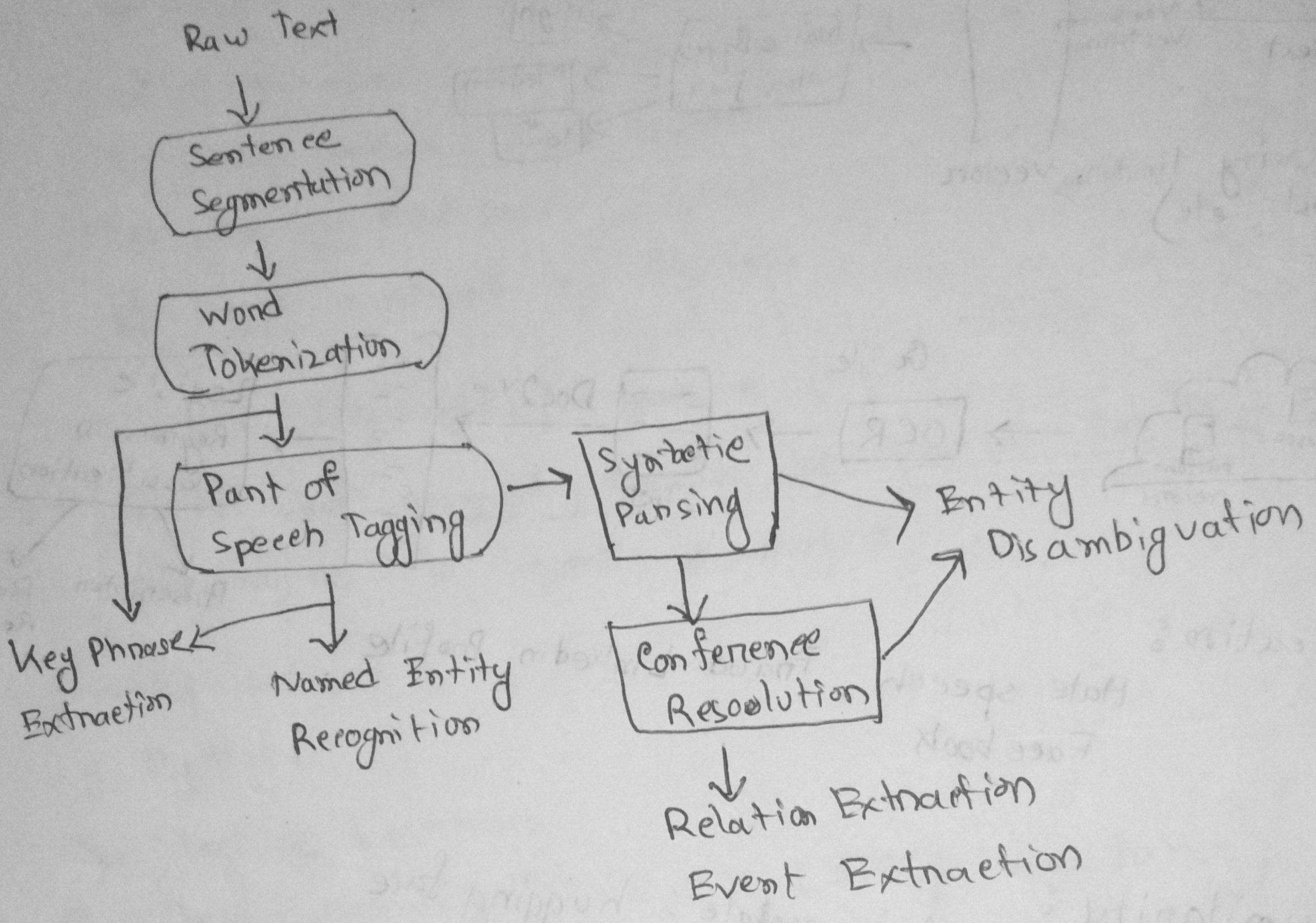


Application: Hate speech - Facebook, Fraud LinkedIn Profile.

Text similarity: search in google: hugging face sentence transformer



Information Extraction

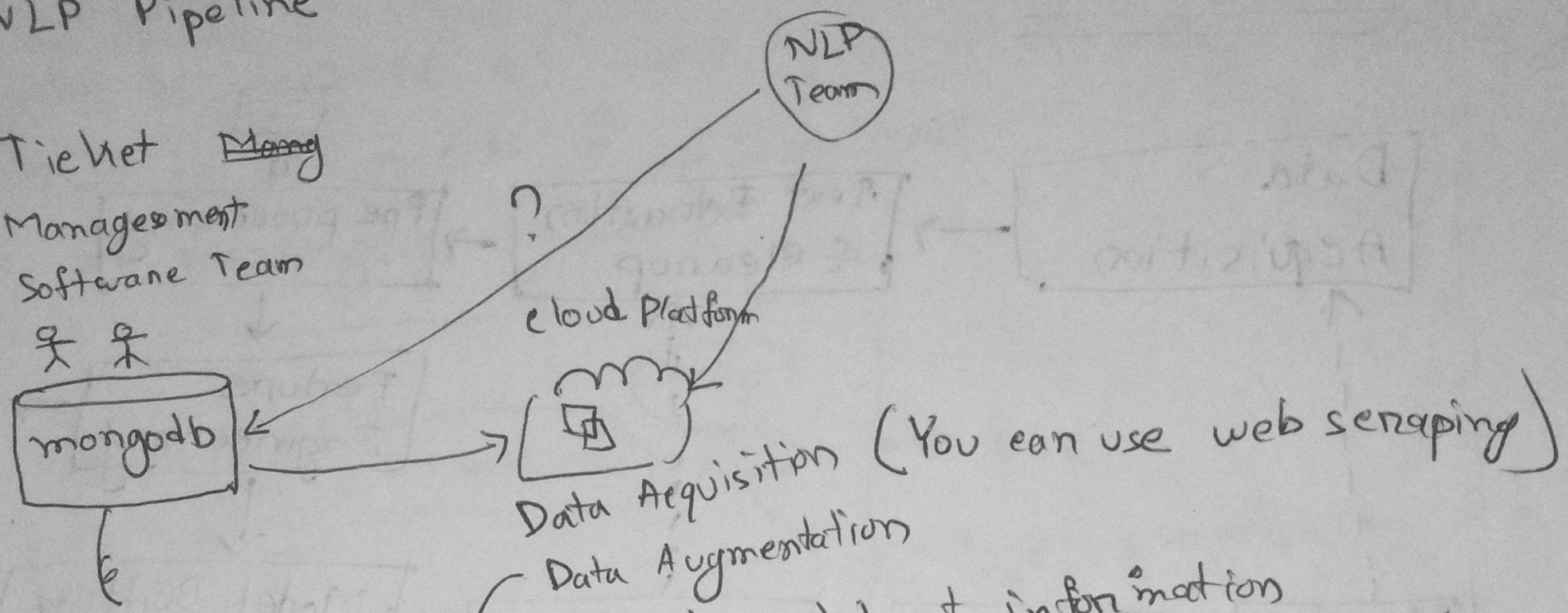


Information Retrieval

↳ Return the relevant website in sorted order

TF-IDF score	BERT
chatbots, Google Translator, Language Modelling,	Statistical Model | Neural Model, Text summarization, Topic Modelling, Voice Assistants,

NLP Pipeline



Feature Engineering
CTF-IDF Vectorizer
One Hot Encoding
Word Embedding)

~~Build Naive Bayes Models~~ ML Models

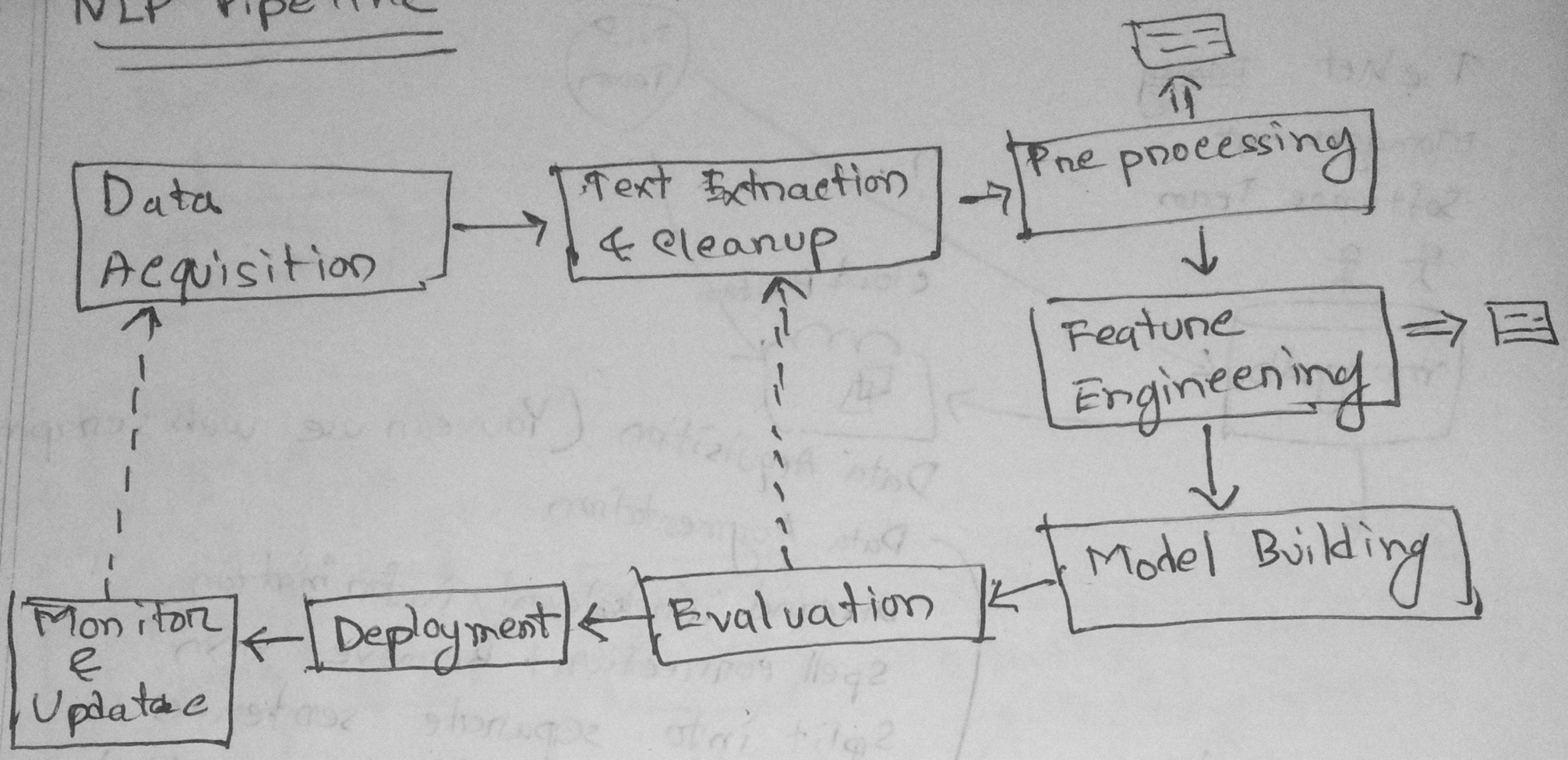
- Hyperparameters Tuning (Randomized search, Grid search)
- Evaluate the Model

Stemming	Lemmatization
adjustable → adjust	was → (to) be
formality → formal	better → good
airliner → airline	meeting → meeting

Lastly, Deploy it on Google cloud / AWS

Maintain and update the model

NLP Pipeline



Spacy

→ Spacy is Object Oriented

→ Provides most effieient NLP algorithms for a given tasks. Hence if you eare about the end result, go with spacy

→ User friendly

→ Perfect for app developers

→ New library, very active user community

NLTK

→ NLTK is mainly a string processing library.

→ Provides access to many algorithms. If you eare about specific algo and customization go with NLTK

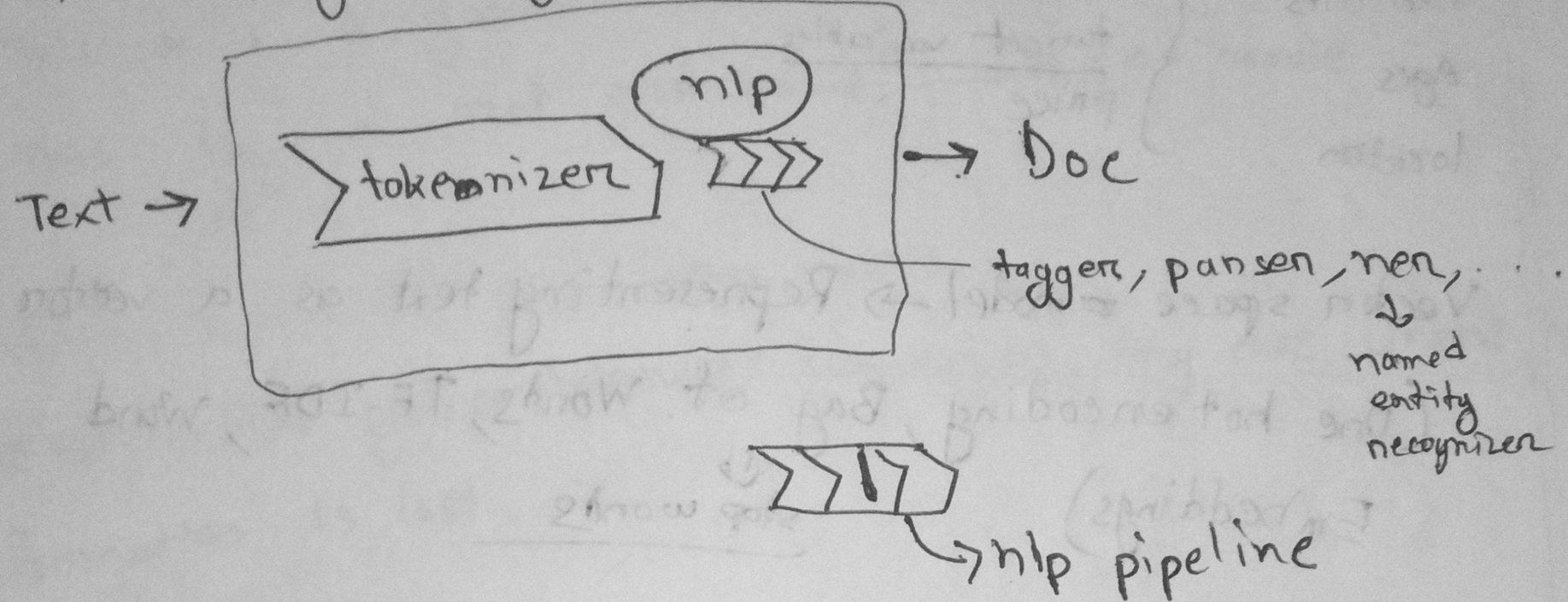
→ less user friendly

→ Perfect for Reseanchen's

→ Old library, Not ~~much~~ much active as Spacy

Tokenization

- can be sentence or word type → stemming, Lemmatization
- It is a process of splitting text into meaningful segments



~~nlp~~

```
nlp.pipe_names
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
                                                    ↳ named Entity recognition
```

fixed rules / heuristics to derive the base word.

↳ stemming

lemmatization → (linguistic knowledge) to derive the base word.

stemming and lemmatization → both are important

stemming → NLTK (spacy does not support)

lemmatization → Both (spacy, NLTK)

Label Encoding	One Hot Encoding	Bag of Words / Bag of n-grams	TF-IDF	Word Embeddings

[8]

Feature Engineering → Extracting features from raw data.

In NLP, Text Representation

Features

- Area
- Facilities
- Ages
- location

} target variables

price

Vector space model → Representing text as a vector

(One hot encoding, Bag of Words, TF-IDF, Word Embeddings)

↓↓

stop words

Label encoding, One Hot encoding → In NLP, You can not use it.

#Disadvantages

similar words does not have similar representation

Consume too much memory and compute resources

Out of vocabulary (OOV) problem.

No Fixed length representation.

#Bag of words → Sparse representation

- lot of memories, computer resources
- Does not capture the meaning of the sentence

↳ Count vectorizers

	musk	tesla	iphone	ipad
article1	14	0	1	5
article2	0	5	0	2

Stop words in BOW (To remove extra words)

When should I not remove stop words (losing important Information)
(Does not need to translate like name)

{ This is a good movie ⟶ good movie
This is a not good movie ⟶ good movie

where should not use stop words
{ chat bot, Q&A system, Language Translation, Any case where valuable information is lost. }

Bag of n-grams → order of ~~and~~ words is important, capturing the relationship with words

(One) 1-grams	Dhaval	sat	on	a	sofa	and ate
(bi) 2-grams						
(Tri) 3-grams						

several n grams combinedly

	Thon eat	eat pizza	loki fall	loki eat
Doc1	1	1	0	0
Doc2	0	0	1	0
Doc3	0	1	0	1

limitation of bag of n-grams model

⇒ As n increases, dimensionally spansity increases
⇒ Does not address out of vocabulary (OOV) problem

TF-IDF

		musk	that	price	market	investor	iphone	itores	
Apple	article-1	0	32	45	48	26	7	3	Apple
	article-2	0	4	3	7	8	6	3	
Tesla	article-3	15	31	44	43	25	0	0	Tesla
	article-4	93	0	0	6	0	0	0	

that → 3
iphone → 2

Document frequency (DF) = Number of times term t is present in all docs

*log is used to damped the importance of term that has high frequency

Inverse Document frequency $(IDF)_{(t)} = \log\left(\frac{\text{Total Documents}}{\text{Document frequency}}\right)$

that → $\frac{4\ (\text{Total})}{3}$ iphon → $\frac{4\ (\text{Total})}{2}$

To normalize thing (to avoid biusness, be fair)

To reduce the effect $TF(t,d) = \left(\frac{\text{Total number of time term t is present in doc A}}{\text{Total number of tokens in doc A}}\right)$

$TF\text{-}IDF = TF(t,d) * IDF(t)$

multiplication.