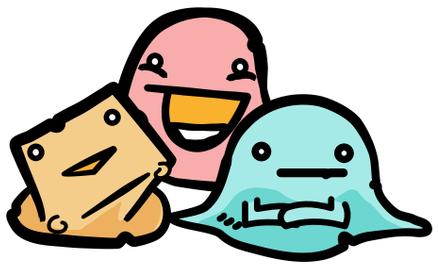
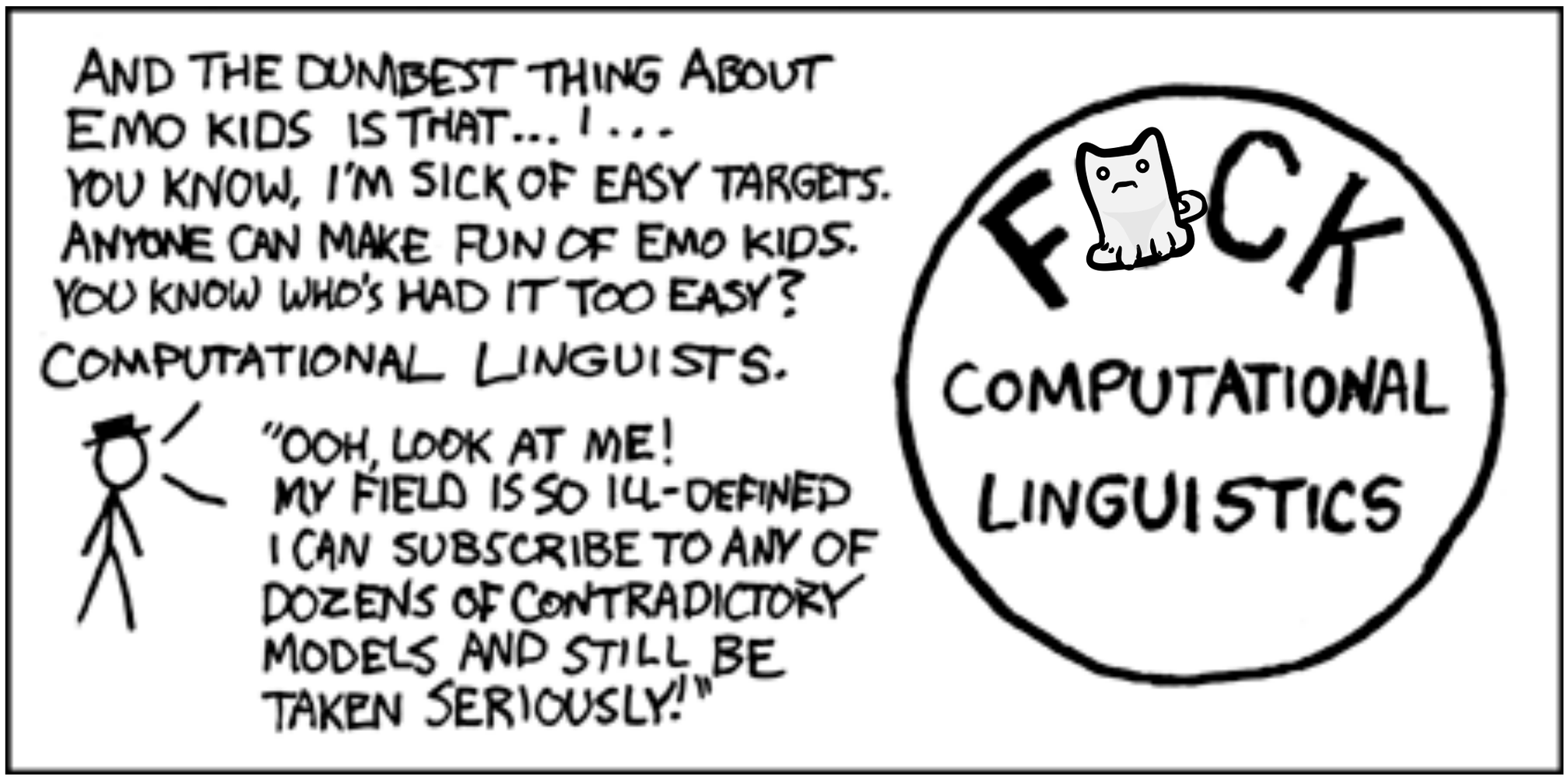
YOUR FIRST NLP PROJECT: PEAKS AND PITFALLS OF UNSTRUCTURED DATA



ML4ALL

Portland, May 27-29, 2018

ABOUT ME: COMPUTATIONAL LINGUIST

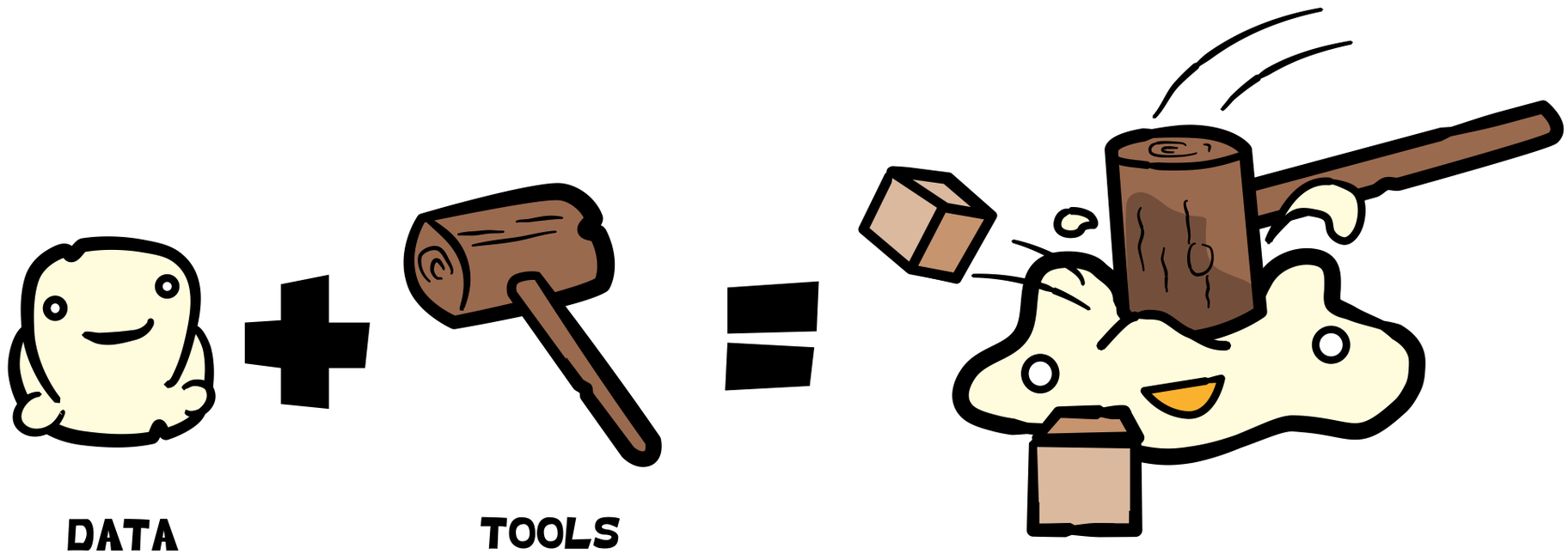


@widiger_anna

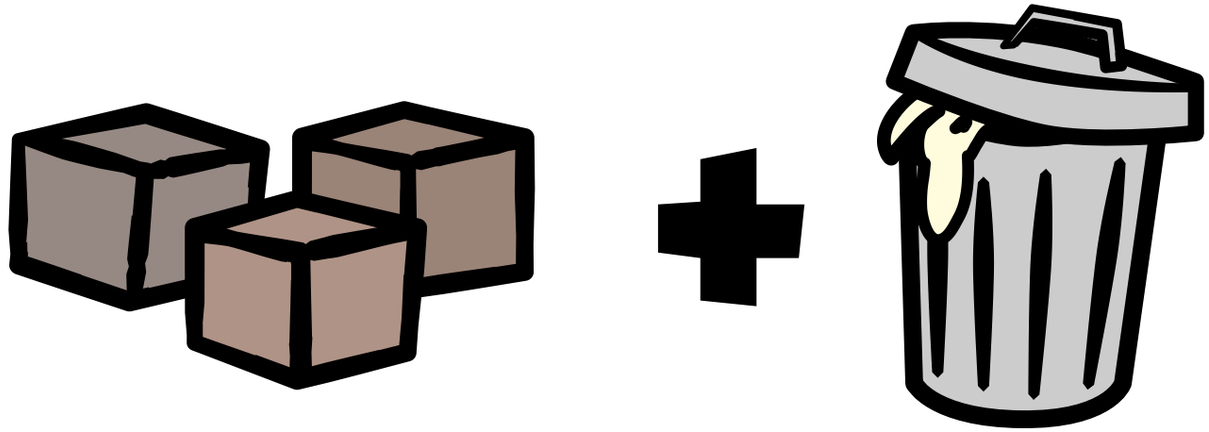
OVERVIEW

1. Intro to NLP
2. Recipes with spaCy:
    a. Emoji and syntax in a tweet
    b. Named entities in a Yelp review
    c. Multilingual topic modeling
3. Takeaways

NLP AND ML:
FROM TEXT TO TOKENS



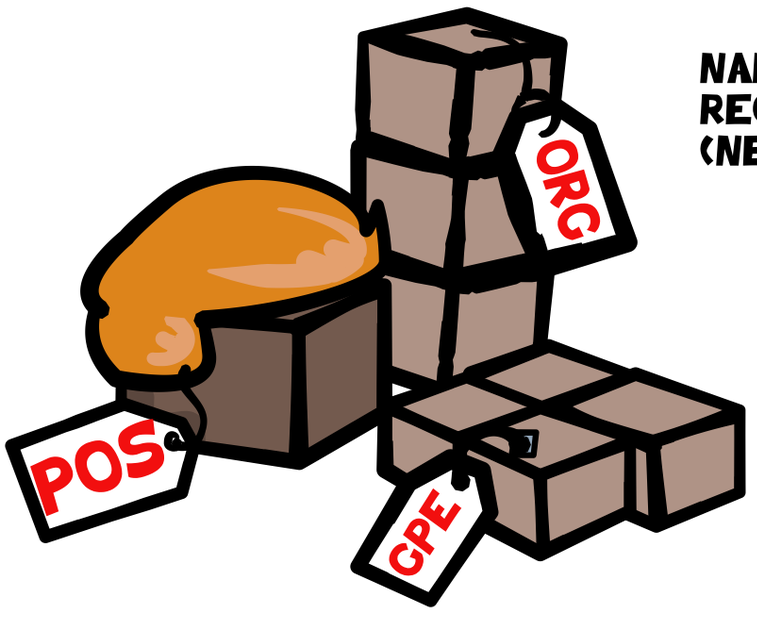
NLP AND ML:
FROM TEXT TO TOKENS



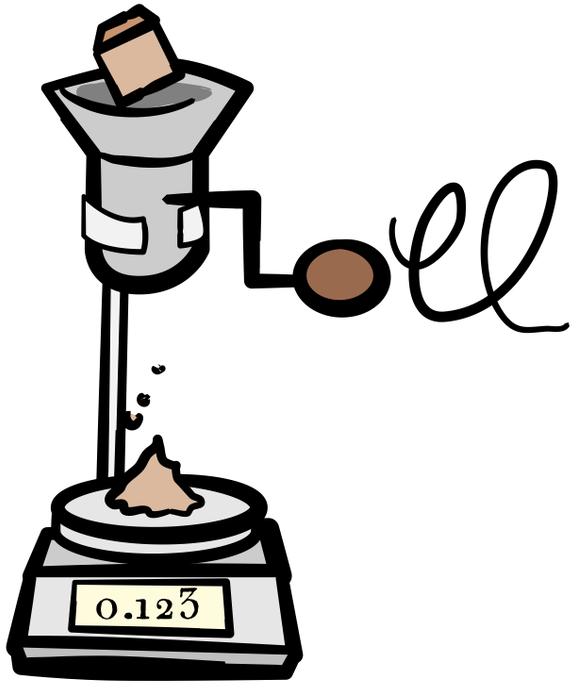
WHAT CAN YOU DO WITH TOKENS?

label

vectorize



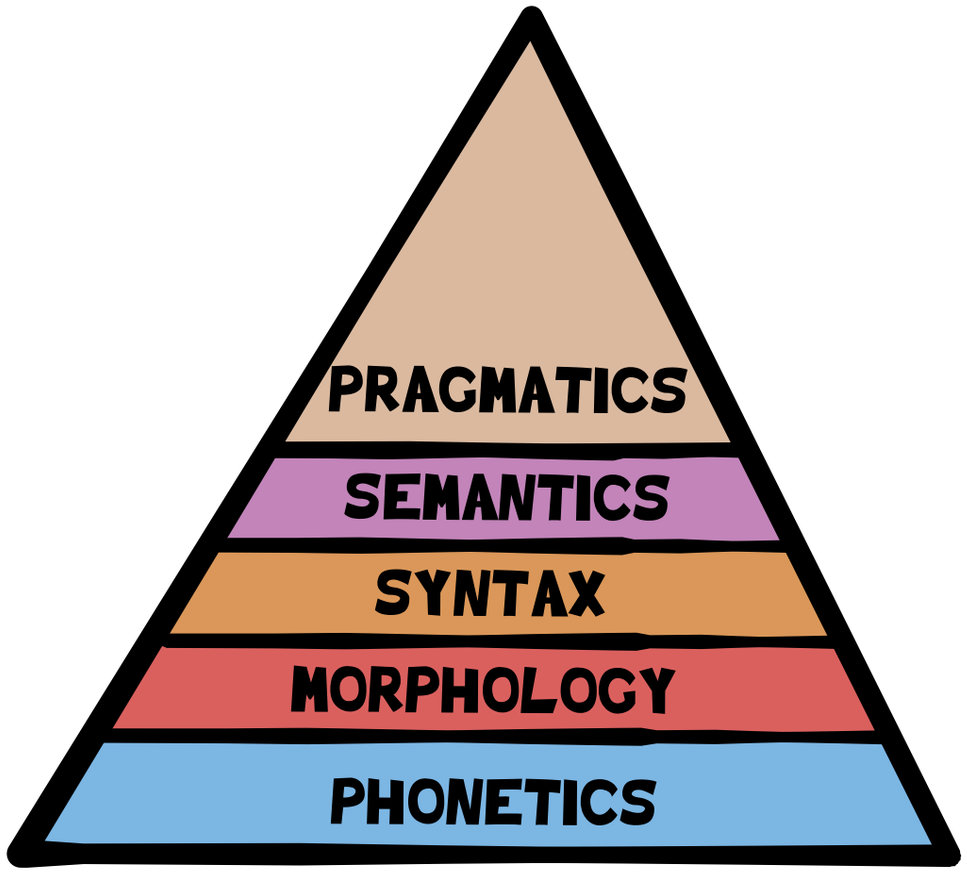
NAMED ENTITY RECOGNITION (NER)



NLP AND DL

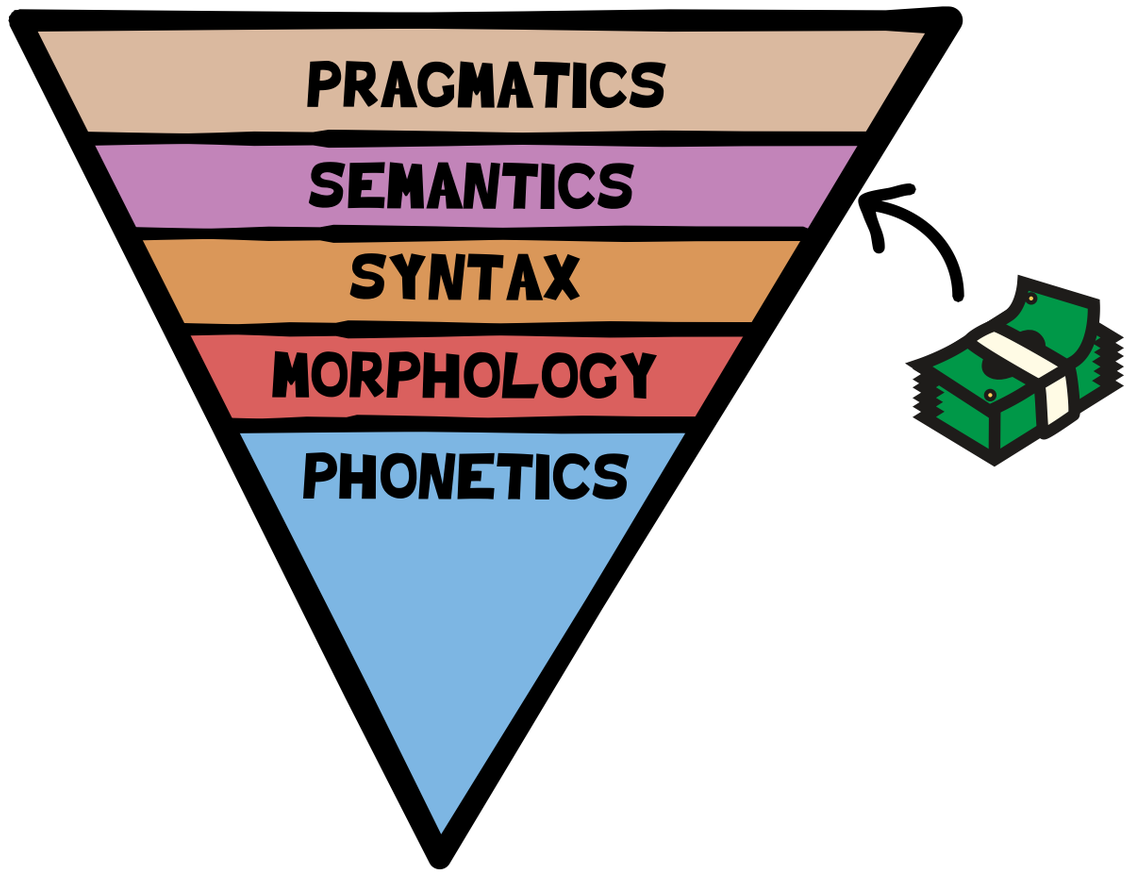
NLP PYRAMID OF SADNESS

LINGUISTS VIEW OF THE WORLD



EACH LAYER BUILT ON THE PREVIOUS

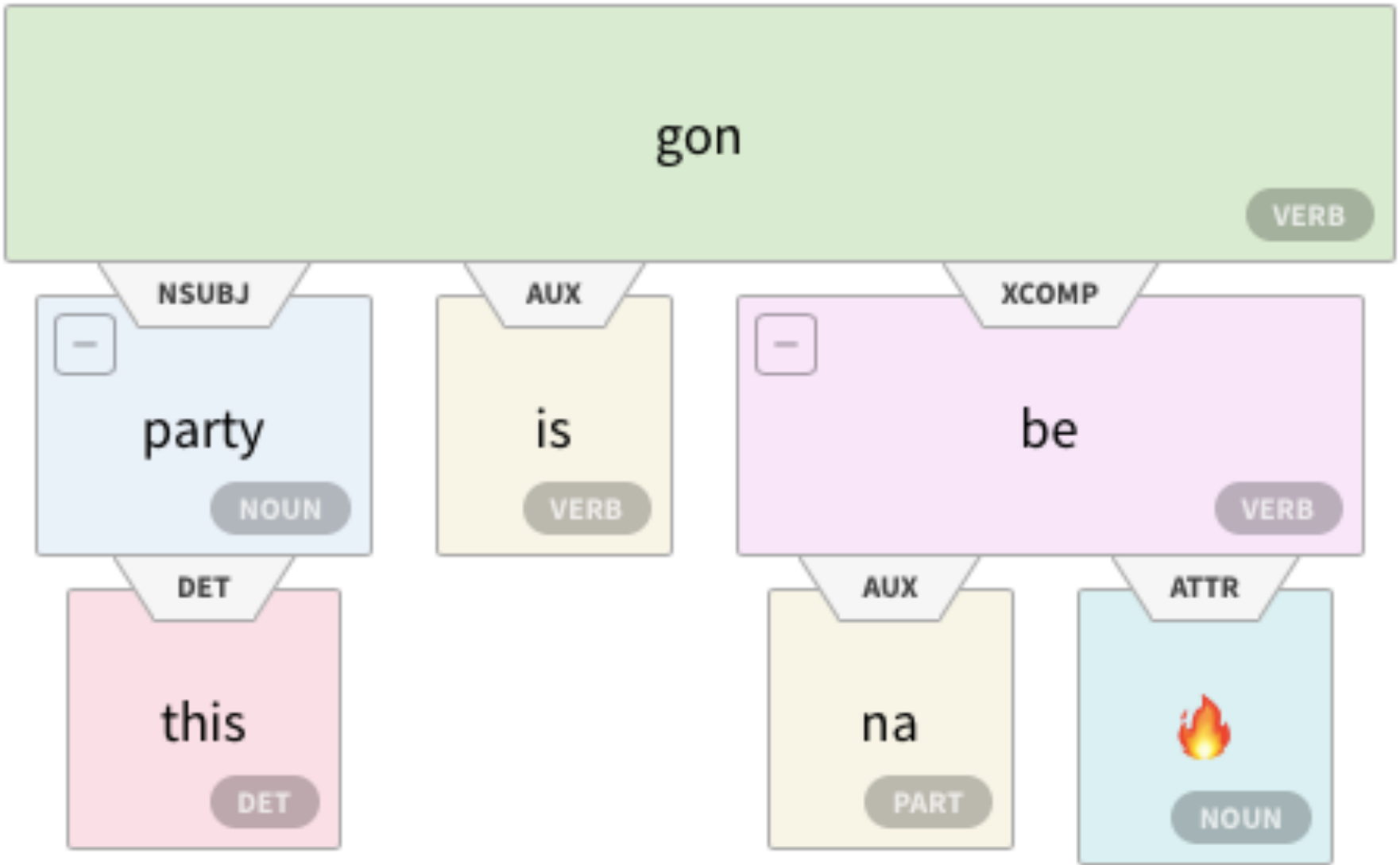
ML VIEW OF THE WORLD



WHERE IS THE MONEY & HOW HARD IS THE PROBLEM

SYNTACTIC PARSING

This party is gonna be 🔥

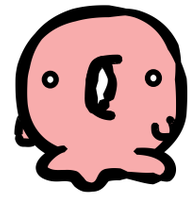


http://spacyvis.allennlp.org/spacy-parser

NLP PITFALLS

Garbage In, Garbage Out (special characters, encoding)

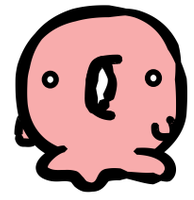
RT: Para pontuação também! #python @stackoverflow



NLP PITFALLS

Sounds and meaning (homonyms)

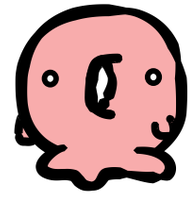
tear in eye, your dress will tear -Gerard Nolst Trenité



NLP PITFALLS

Word forms and noun phrases

This agreement is about a principled compromise, not a principle compromised. -John Hume

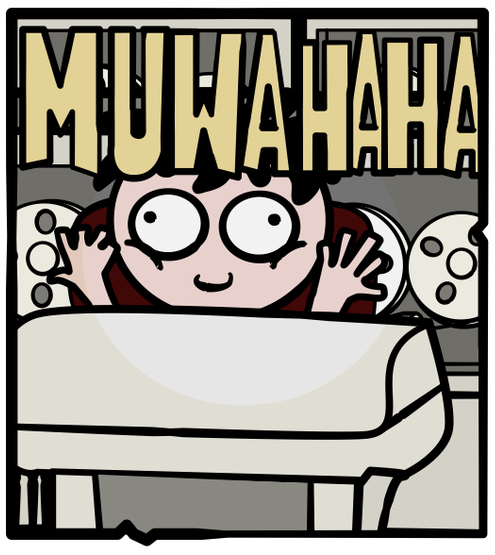


NLP: DATA DIVIDE

Big Data

Law of large numbers

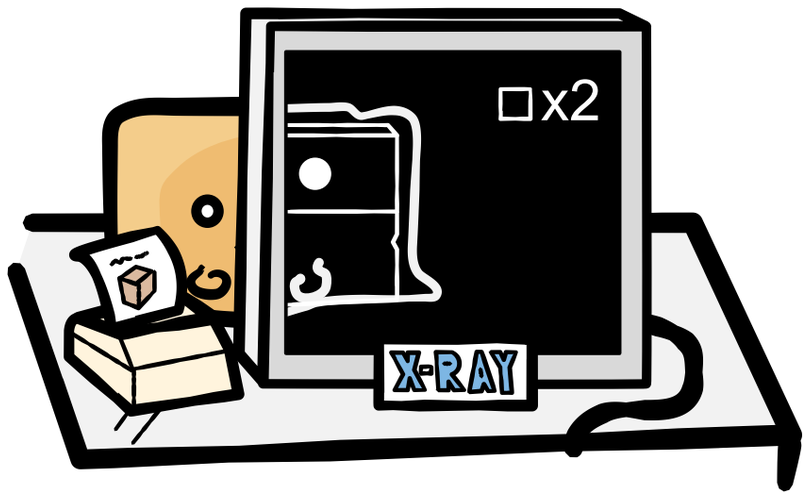
Annotation tools



Small Data

Every word counts

Pre-processing tools



NLP PIPELINES WITH SPACY

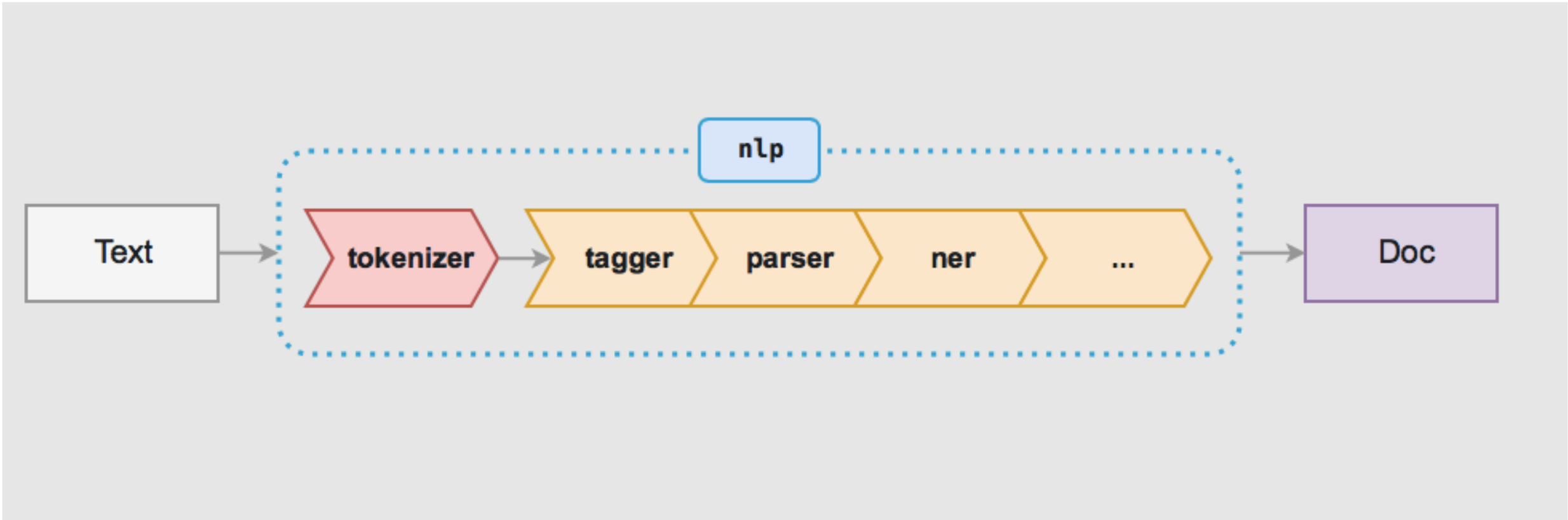
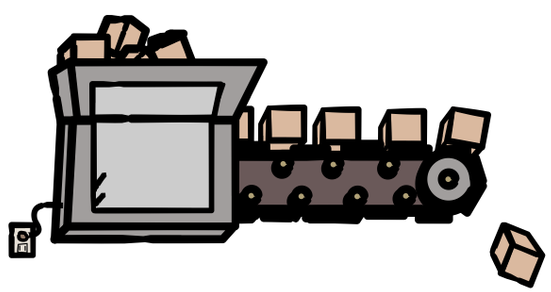


image source: spaCy pipeline documentation



RECIPE 1: ANATOMY OF A TWEET

https://twitter.com/dataandme/status/989938791744987137

RECIPE 1: PIPELINE

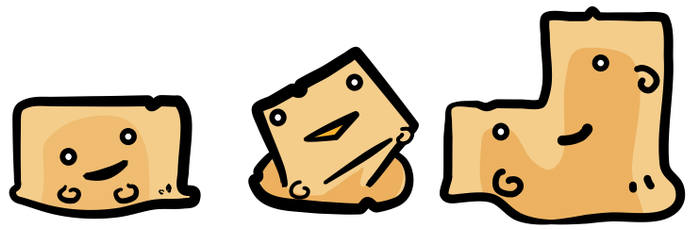
1. Tokenization: word boundary != whitespace

   income tax return 🇺🇸 = Einkommenssteuererklärung 🇩🇪

2. POS tagging, syntactic parsing, and emoji

3. Chunking (nice to have, improves accuracy)

   [chocolate bar], [bar exam], [wine bar], [space bar]



RECIPE 1: EMOJI DICTIONARY

Adding your own features/labels to tokens/chunks:

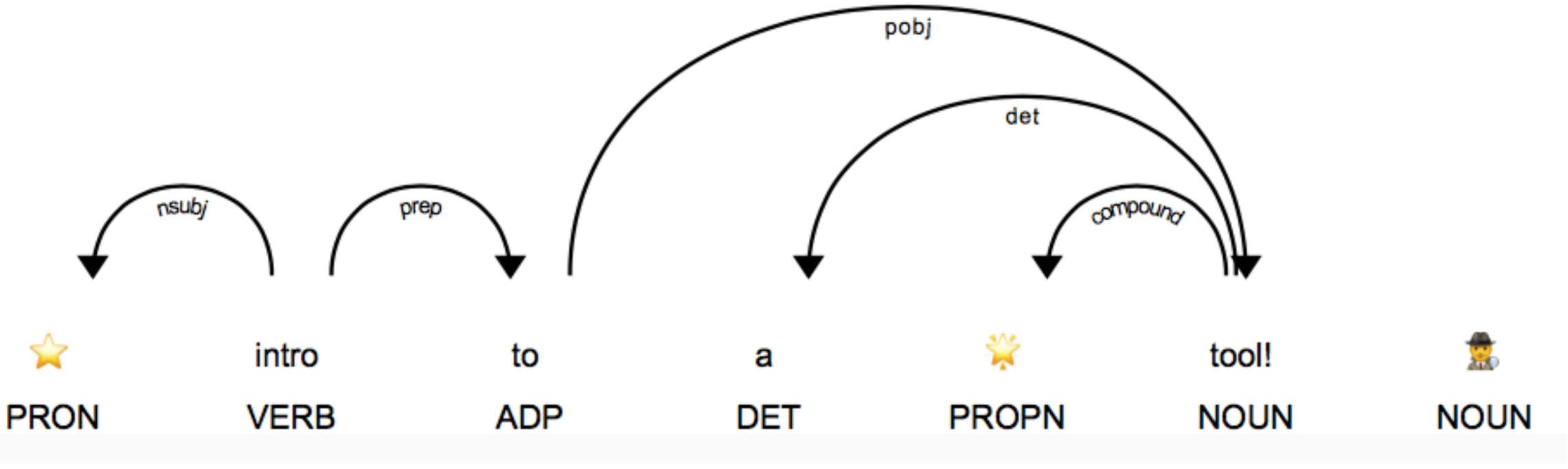
```
def is_positive(text):
    positive = [u'⭐', u'🌟', u'😀', u'🔥', u'✨', u'👍', u'👏']
    # want to use an existing sentiment system, for en? check out NLTK Vader
    # just note: it doesn't deal with unicode emoji only ascii emoji :) welcome to pitfalls
    return text in positive

SENTIMENT_POS = nlp.vocab.add_flag(is_positive)
```

Looking at your data may provide interesting insights:

```
alpha:  intro to a tool RegExplain by rstats RegEx
not_alpha:  ⭐ 🌟 ! 🕵    @grrrck https://buff.ly/2HXtBZ1    # #
sentiment score:  2
```

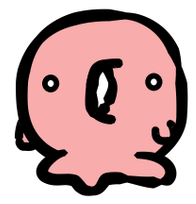
RECIPE 1: PARSE TREE

```
displacy.render(doc1, style='dep', jupyter=True, options= {'distance':120})
```



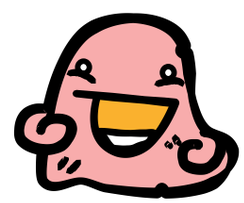
RECIPE 1:
NLP PITFALLS

- encoding (ascii, utf8 or unicode?)
- special characters and punctuation/emoji
- wrong POS tags



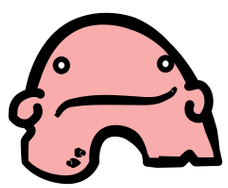
RECIPE 1: BEST PRACTICES

- do: identify language before any preprocessing
- do: filter or remove non-alphabetic characters
- nice to have: a custom dictionary (Vocab)
- nice to have: chunking/noun phrases



RECIPE 1: BEST PRACTICES

- do not: assume all text is in clean and simple English
- do not: rely on pre-trained models for POS tags



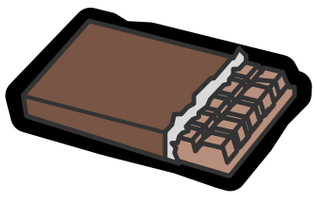
Recipe 2: Sentiment of a Yelp Review

5/7/2016

We drove 30min out of our way to get some dessert and ordered a ton to go... Only to find that they didn't give us what we ordered. What we did get was not good. We called and they refunded us $4.... Ridiculous. Don't waste your time or money here. Sixth Course right around the corner is WAY better.

1 person voted for this review

Useful | Funny 1 | Cool



RECIPE 2: PIPELINE

1. Normalization

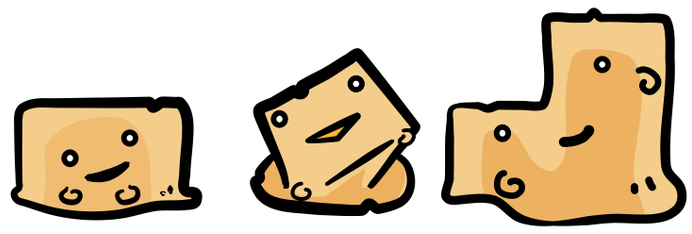
   it’s -> it is, drove -> drive, $5 -> five dollars

2. Stop word removal

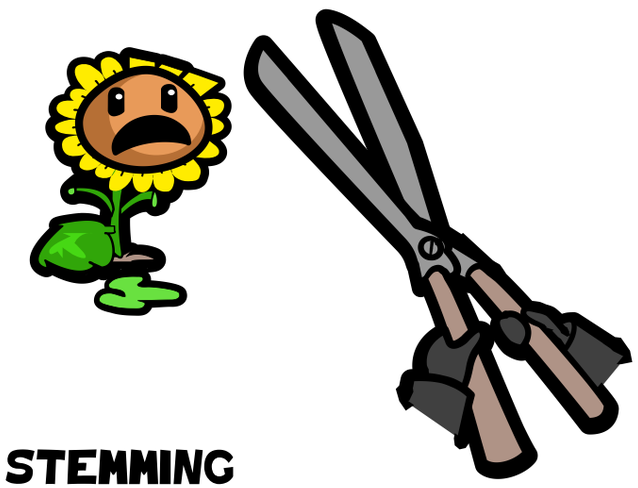
   the, and, of

3. Named Entity Recognition (NER)

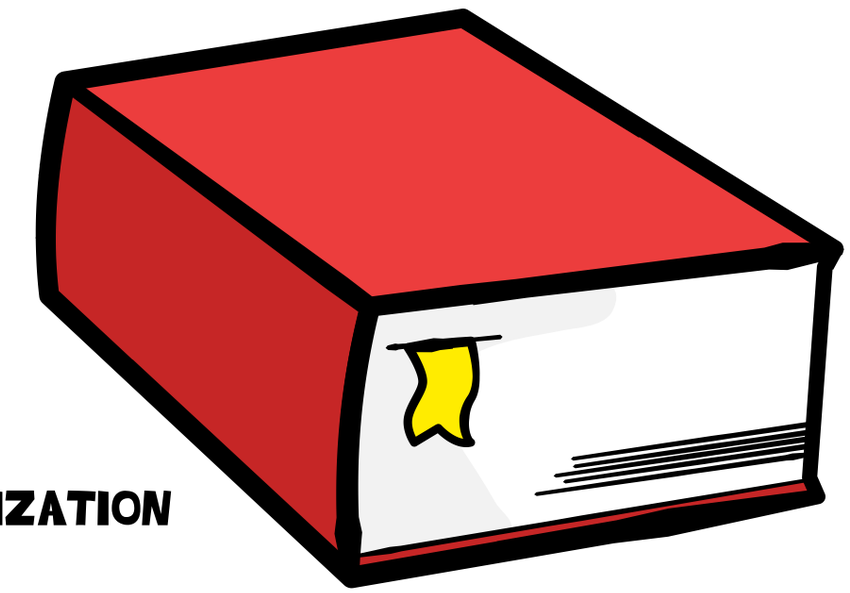
   Portland [GPE], May 1st [DATE]



RECIPE 3: STEMMING VS LEMMATIZATION



STEMMING



LEMMATIZATION

running : run

runs : run

employee : employe

wolves : wolve

running : run

runs : run

ran : run

employee : employee

employer : employer

wolves : wolf

RECIPE 2: NAMED ENTITIES IN A YELP REVIEW

```
#Filter stopwords and punctuation
for token in doc2:
    if not token.is_punct and not token.is_stop and token.pos_ !='PRON':
        filtered_tokens.append(token.lemma_)

#List of named entity labels to look for in the text
tracked_ne = ['PERSON', 'ORG', 'GPE', 'DATE', 'MONEY', 'CARDINAL', 'QUANTITY']
for named_entity in doc2.ents:
    if named_entity.label_ in tracked_ne:
        named_entities.append(named_entity.text)

print("words: ", " ".join(filtered_tokens))
print("named entities: ", ", ".join(named_entities))
```

RECIPE 2: NAMED ENTITIES IN A YELP REVIEW

```
displacy.render(doc2, style='ent', jupyter=True, options= {'distance':140}
)
```

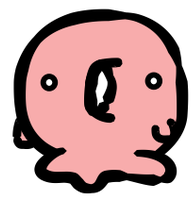
We drove 30min CARDINAL out of our way to get some dessert and ordered a ton QUANTITY to go... Only to find that they did not give us what we ordered. What we did get was not good. We called and they refunded us $ 4 MONEY .... Ridiculous GPE . Do not waste your time or money here. Sixth Course right around the corner is WAY ORG better.

RECIPE 2:
NLP PITFALLS

Expanding contractions

Anna’s : possessive or contraction?

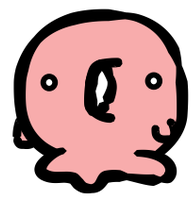
I’d -> : had or I would?



RECIPE 2:
NLP PITFALLS

Spell check suggestions

mango chili chocolate = mange child chocolate

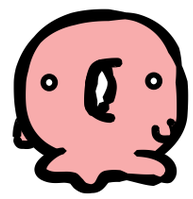


RECIPE 2:
NLP PITFALLS

Named Entity Recognition vs. lowercase

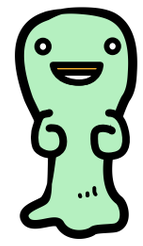
a Rose by any other name

US : United States, us : we



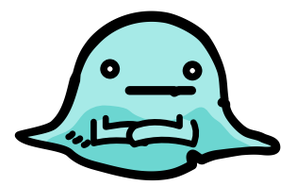
RECIPE 2: BEST PRACTICES

- do: Named Entity Recognition before lowercase
- optional: expand contractions before tokenization
- (optional) custom rules for noun phrases or negation



RECIPE 2: BEST PRACTICES

- do not: mess up the order of operations in the pipeline
- do not: rely on pre-trained models for NER
- do not: ignore domain knowledge (like local cafes)



RECIPE 3: TOPIC MODELING

Getting Started with spaCy for Natural Language Processing

◀ Previous post Next post ▶

Tags: Data Preparation, Data Preprocessing, NLP, Python, Text Analytics, Text Mining

spaCy is a Python natural language processing library specifically designed with the goal of being a useful library for implementing production-ready systems. It is particularly fast and intuitive, making it a top contender for NLP tasks.

By Matthew Mayo, KDnuggets.

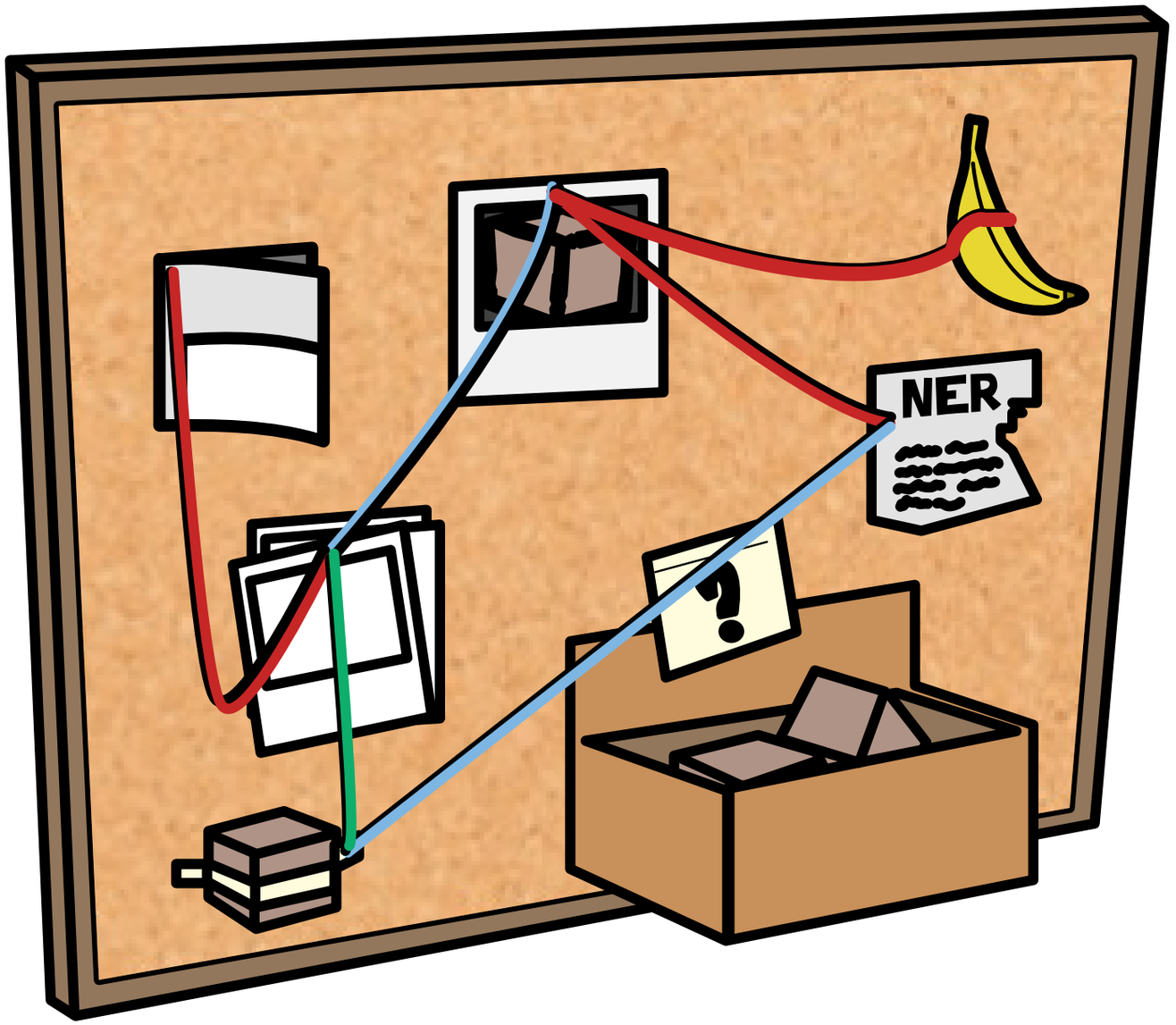
RECIPE 3: TOPIC MODELING

Matthew Mayo Follow
I turn coffee into code in desperate need of refactoring, which I then refactor. #MachineLearning Researcher & Editor @kdnuggets #NeuralNetworks
May 8 · 7 min read

Comenzando con spaCy para procesamiento de lenguaje natural

spaCy es una biblioteca de procesamiento de lenguaje natural Python diseñada específicamente con el objetivo de ser una biblioteca útil para implementar sistemas listos para producción. Es particularmente rápido e intuitivo, por lo que es un competidor superior para las tareas de procesamiento de lenguaje natural (PLN).

RECIPE 3: PIPELINE



RECIPE 3: TOPIC MODELING

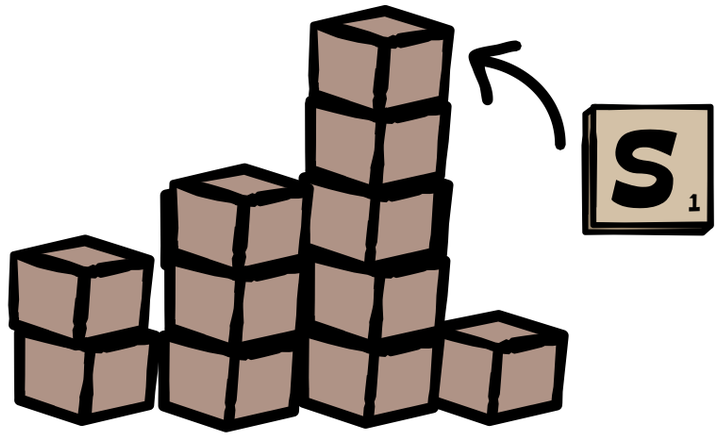
```
Topics Spanish:
          [(0, '0.018*"NLTK" + 0.016*"spaCy"'),
           (1, '0.019*"palabras" + 0.016*"StopWords"'),
           (2, '0.030*"spaCy" + 0.017*"obtener"'),
           (3, '0.023*"spaCy" + 0.018*"NLTK"'),
           (4, '0.018*"PLN" + 0.017*"datos"')]
Topics English:
          [(0, '0.022*"object" + 0.021*"Doc"'),
           (1, '0.024*"NLP" + 0.014*"particular"'),
           (2, '0.018*"preprocessing" + 0.018*"NLTK"'),
           (3, '0.024*"text" + 0.018*"tasks"'),
           (4, '0.020*"NLTK" + 0.017*"object"')]
```

RECIPE 3: BAG OF WORDS

Bag of Words

Order is not important

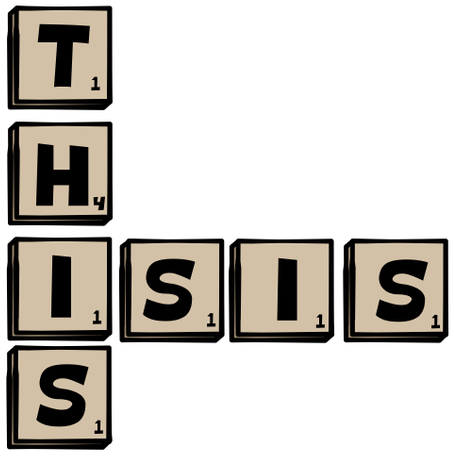
Example: doc2bow



Word Embedding

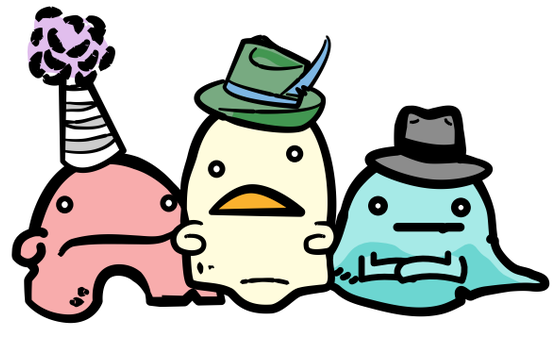
Context is important

Example: doc2vec



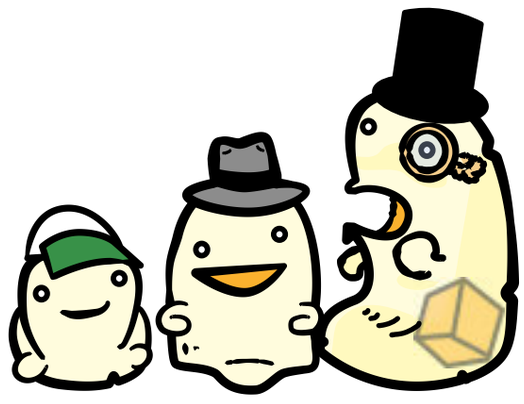
RECIPE 3: BEST PRACTICES

- do: use ngrams
- do: use TF-IDF (Term Frequency - Inverse Document Frequency) instead of chi-squared test or log likelihood ratio tests

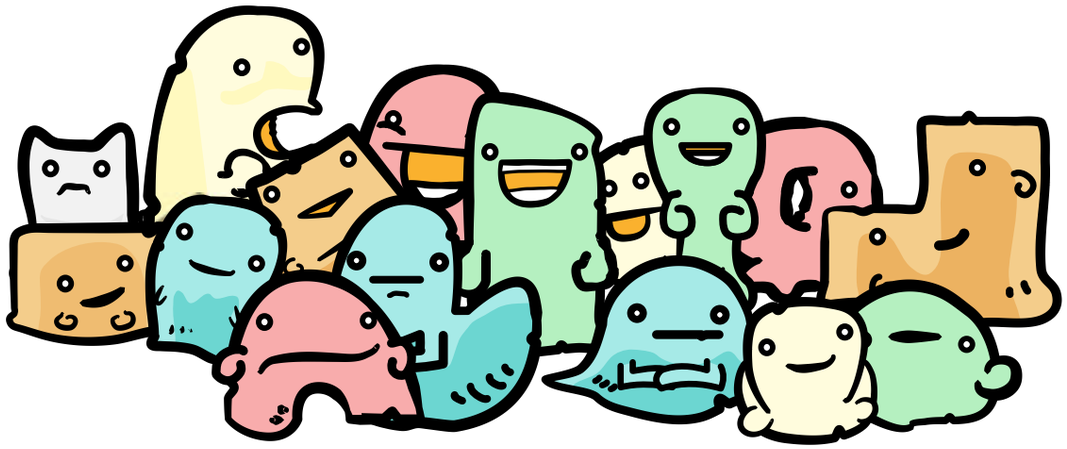


NLP PROJECTS TAKEAWAYS

- 👎 Text blobs are messy and words != numbers
- 👍 Looking at your input and output is helpful
- 👍 Customize NLP pipelines if needed
- 👍 Domain knowledge is important!
- 👍 Lots of NLP tools out there, pick the right one!



THANK YOU!



widiger-anna

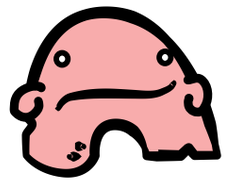
artwork : jay@theotherplayer

APPENDIX

1. Rule-Based NLP
2. Language Engineering
3. NLTK vs. spaCy
4. Word count and vectorization
5. TF-IDF example

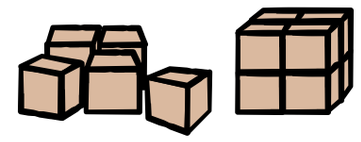
RULE-BASED NLP

Fields	Units	Features
Phonetics	sounds/phonemes	phonetic symbols IPA: /fəˈnɛtɪks/
Morphology	words/morphemes	word forms stems, affixes
Syntax	phrases, sentences	part-of-speech tags syntactic labels
Semantics	meaning/words	ontologies (WordNet) word vectors
Pragmatics	meaning/context	here there be dragons



LANGUAGE ENGINEERING

NLP tasks	Algorithms	Purpose
Pre-Processing	Clustering	Topic Modeling
Annotation	Classification	Sentiment Analysis
Annotation	Deep Learning	Machine Translation
Rules / Grammars	Deep Learning	Question Answering
none	Deep Learning	???



NLTK VS SPACY

	Hi, I'm spaCy	And I'm nltk
Audience	industry	academia
Syntactic parsing	fast, easy to use	verbose, slow
Semantics	word vectors	WordNet
Visualization	displaCy	none
Beyond the basics	API and universe	third party modules
Challenges	agile	stable
Resources	spaCy docs	books and tutorials

NLTK VS SPACY

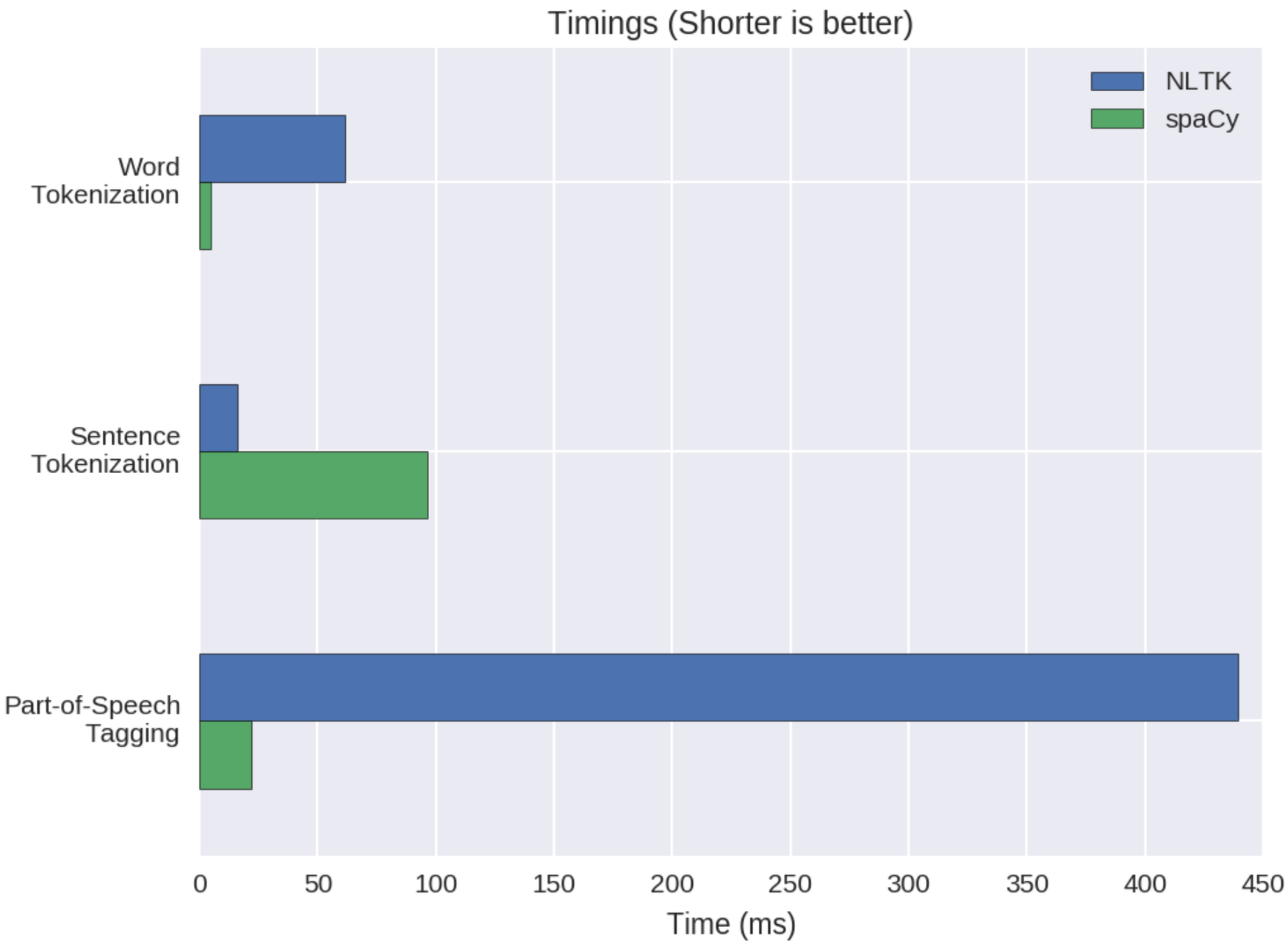
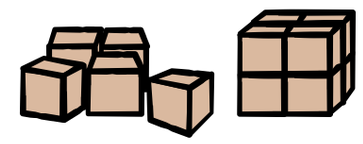


Image source: thedataincubator.com

WORD COUNT AND VECTORS

- Goal: matrix of words (rows) and their relative count per document (columns)
- Binary: one hot encoding
- Relative frequency per document
- TF-IDF



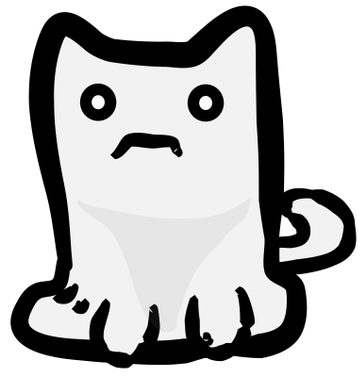
BINARY: ONE HOT ENCODING

ONE-HOT ENCODING

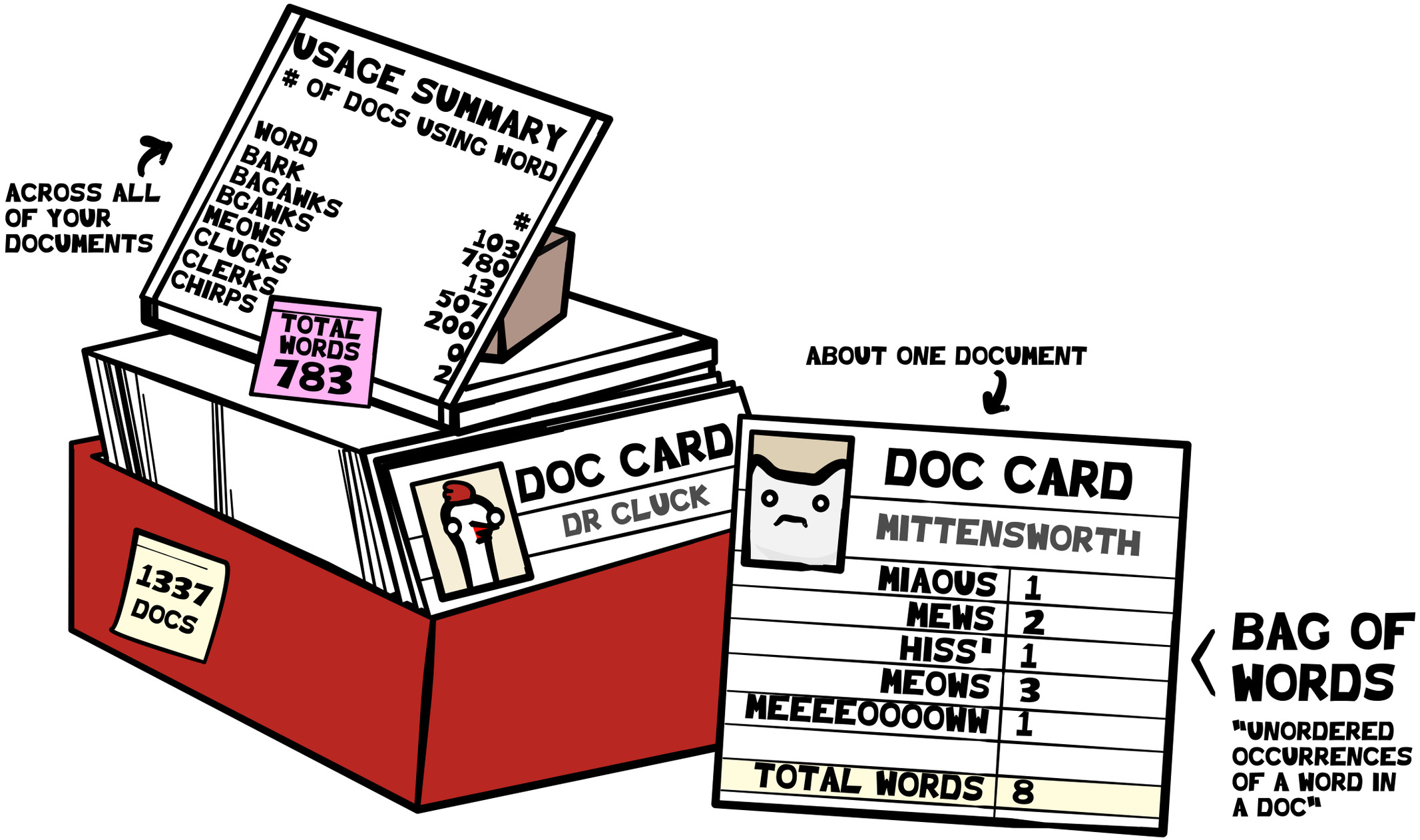
12

0100 0000 0000
0100 0000 0000
0100 0000 0000
0001 0000 0000
1000 0000 0000
0000 0000 0001
0000 1000 0000
0001 0000 0000
? WAITING ON CAT

..... MEOW
..... MEOW
..... MEOW
..... MEW
..... HISS
..... MIAOU
..... MEEEEOOOOWW
..... MEW



RELATIVE FREQUENCY PER DOCUMENT



TF-IDF

TF & IDF

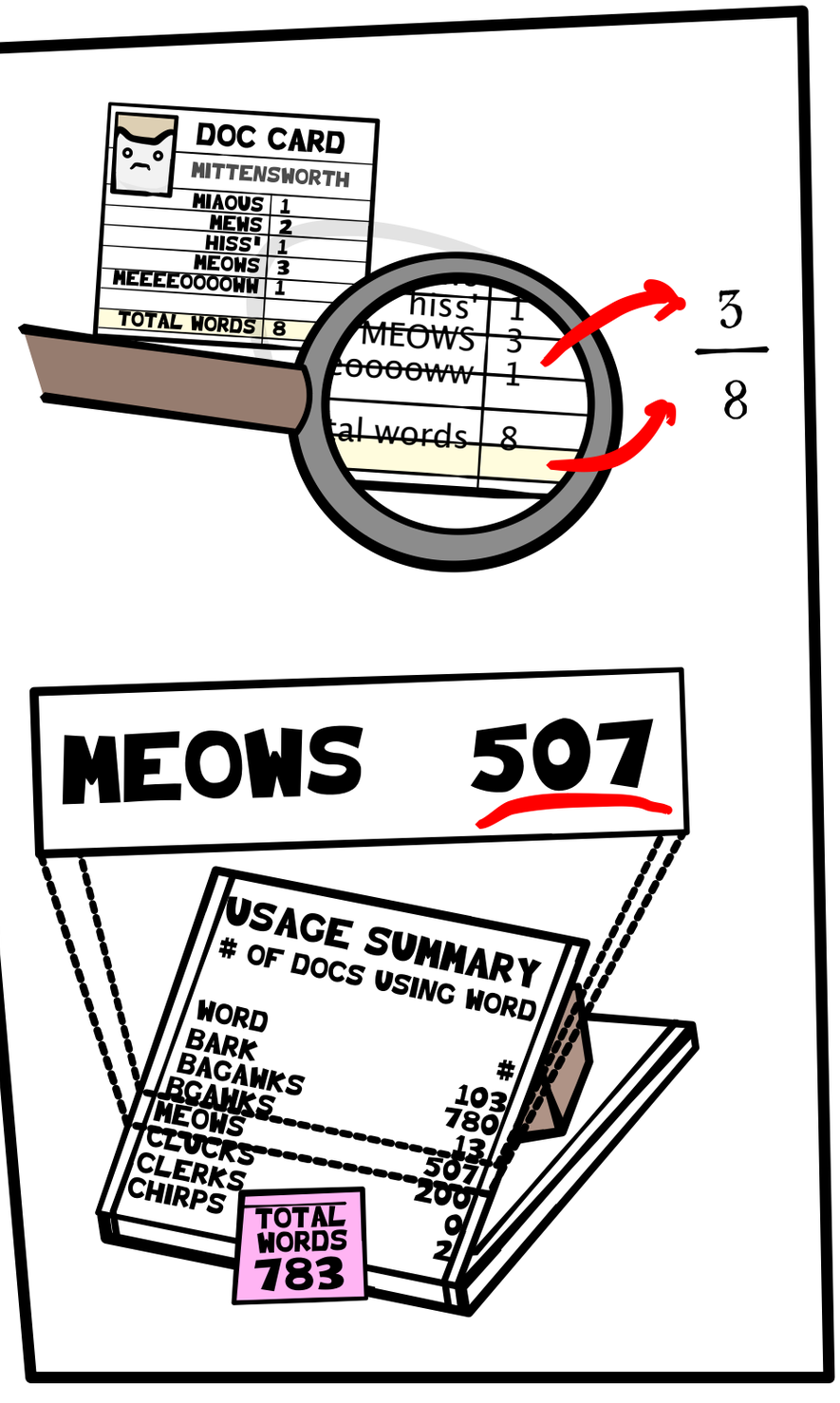
$$tf(t,d) = \frac{\text{occurrences of term}}{\text{terms in document}}$$

$$idf(t,D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|}$$

$$tf\text{-}idf(t,d,D) = tf(t,d) * idf(t,D)$$

$|\{d \in D : t \in d\}|$
LENGTH OF SET OF DOCUMENTS THAT CONTAINS TERM T

SET
$|\{\}|$
CARDINALITY (LENGTH OF)



TF-IDF

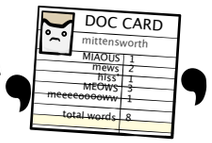
use what you know

= D (documents)

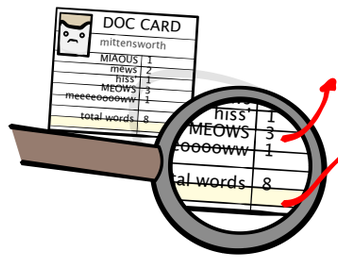
= N = |D|

= d (document)

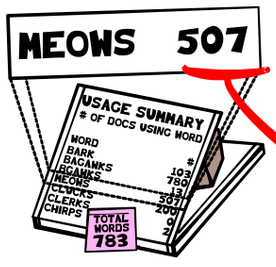
= set of (term, # of documents using term)

TF-IDF("MEOW", 

,D)



$$= \frac{3}{8} = 0.375$$



$$= \log \frac{1337}{1+507} = 0.42$$

$$= 0.375 * 0.42 = 0.1575$$