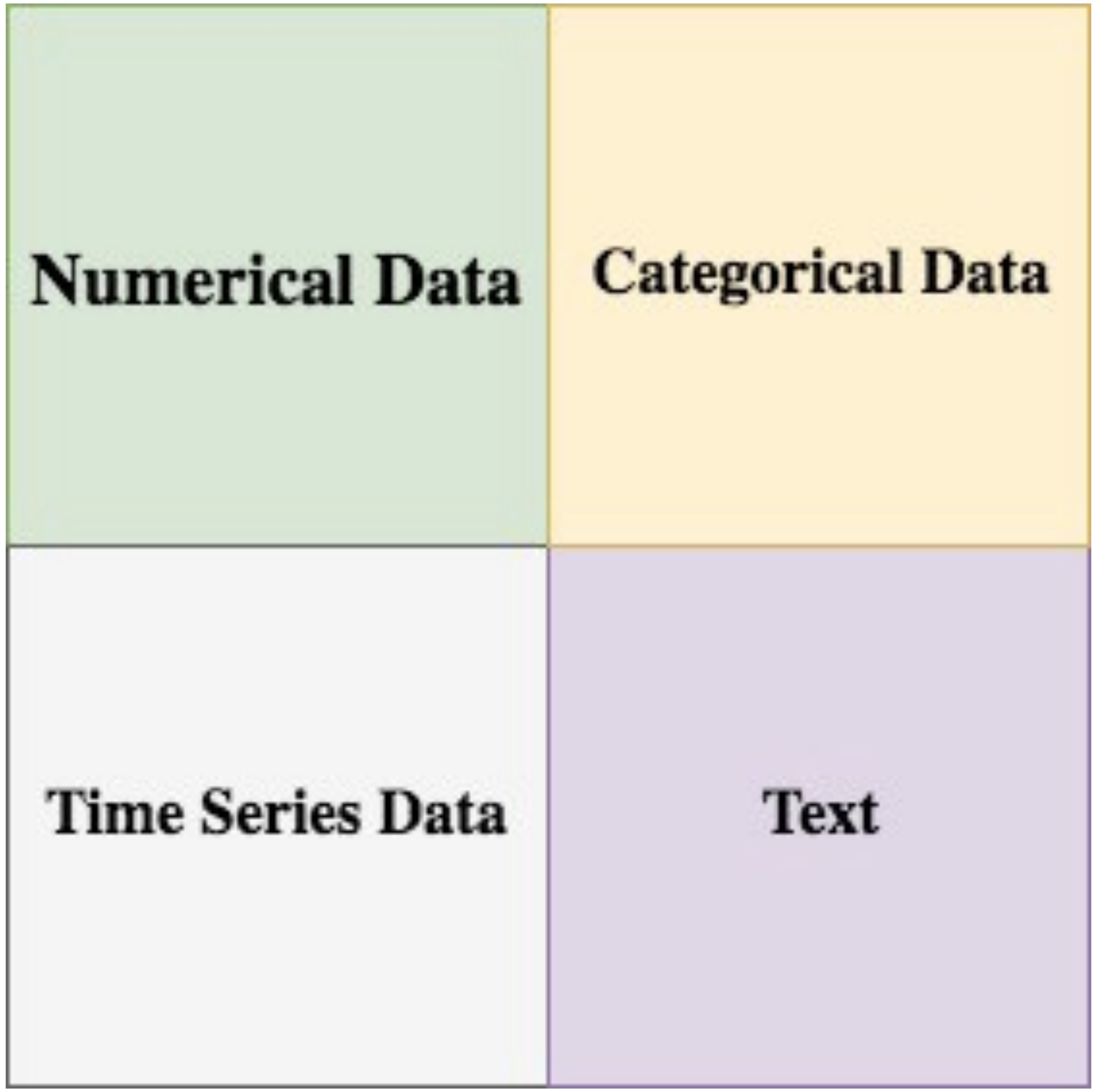
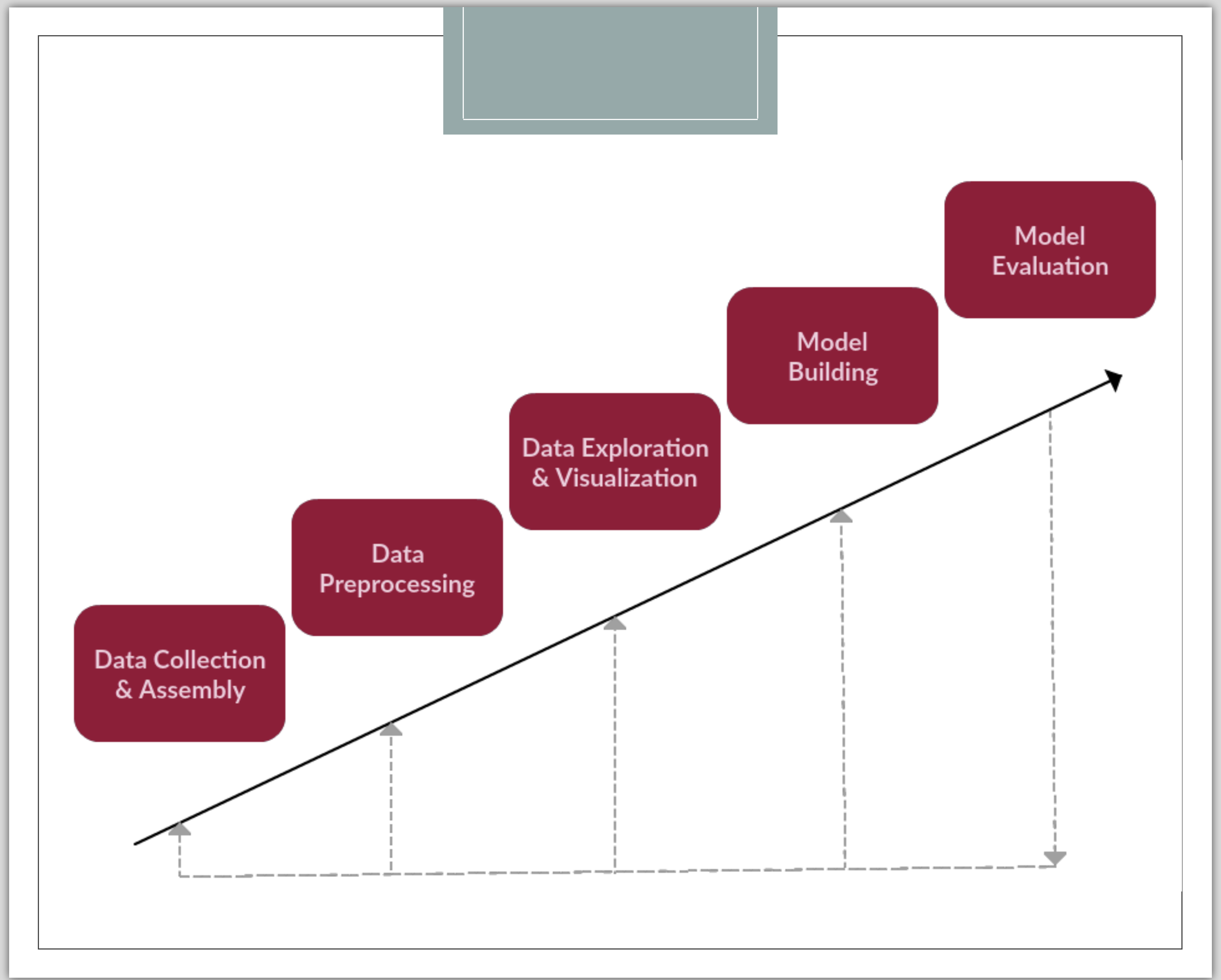
magnimind

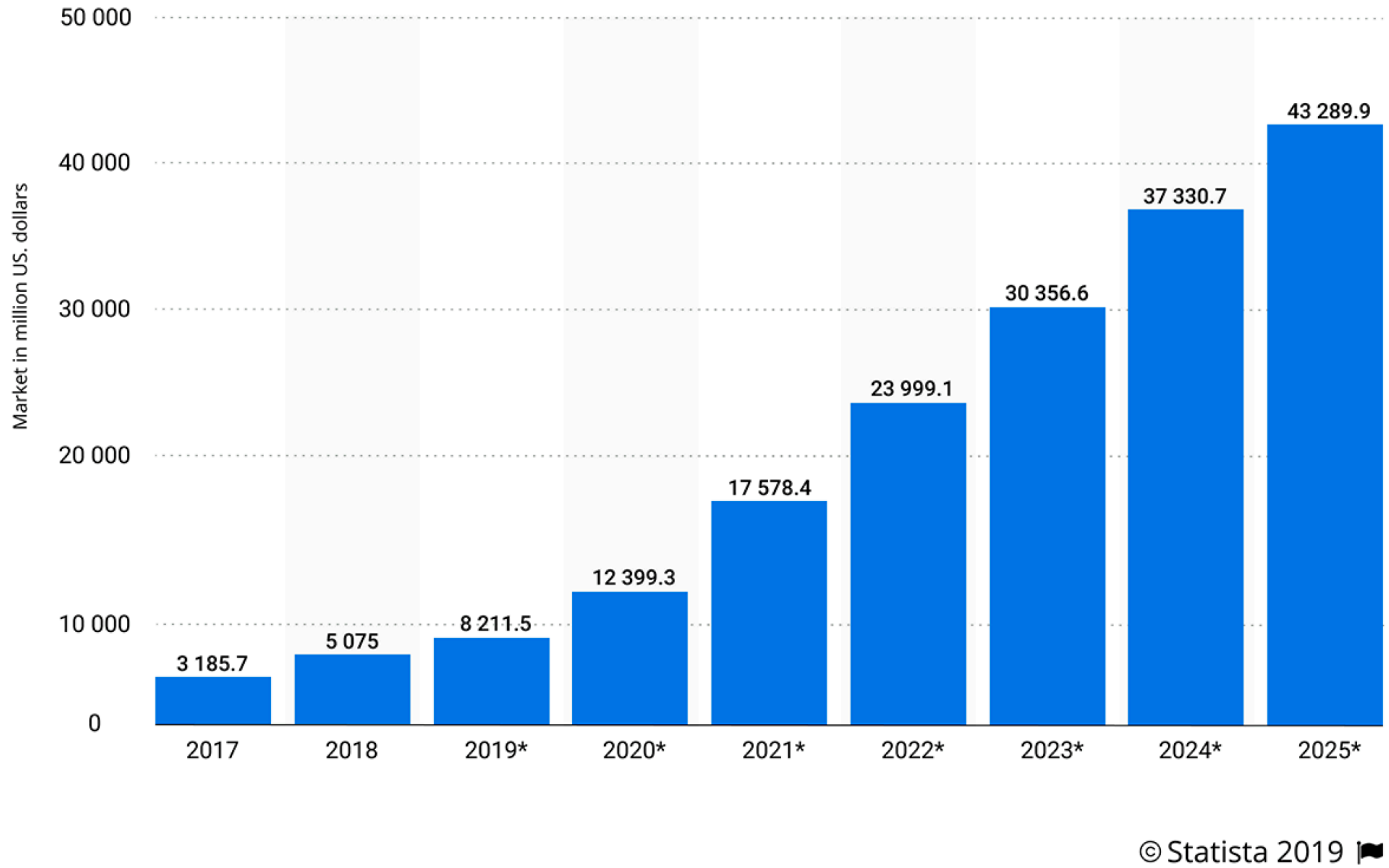
NLP Review

DATA TYPES





TEXT DATA SCIENCE FRAMEWORK



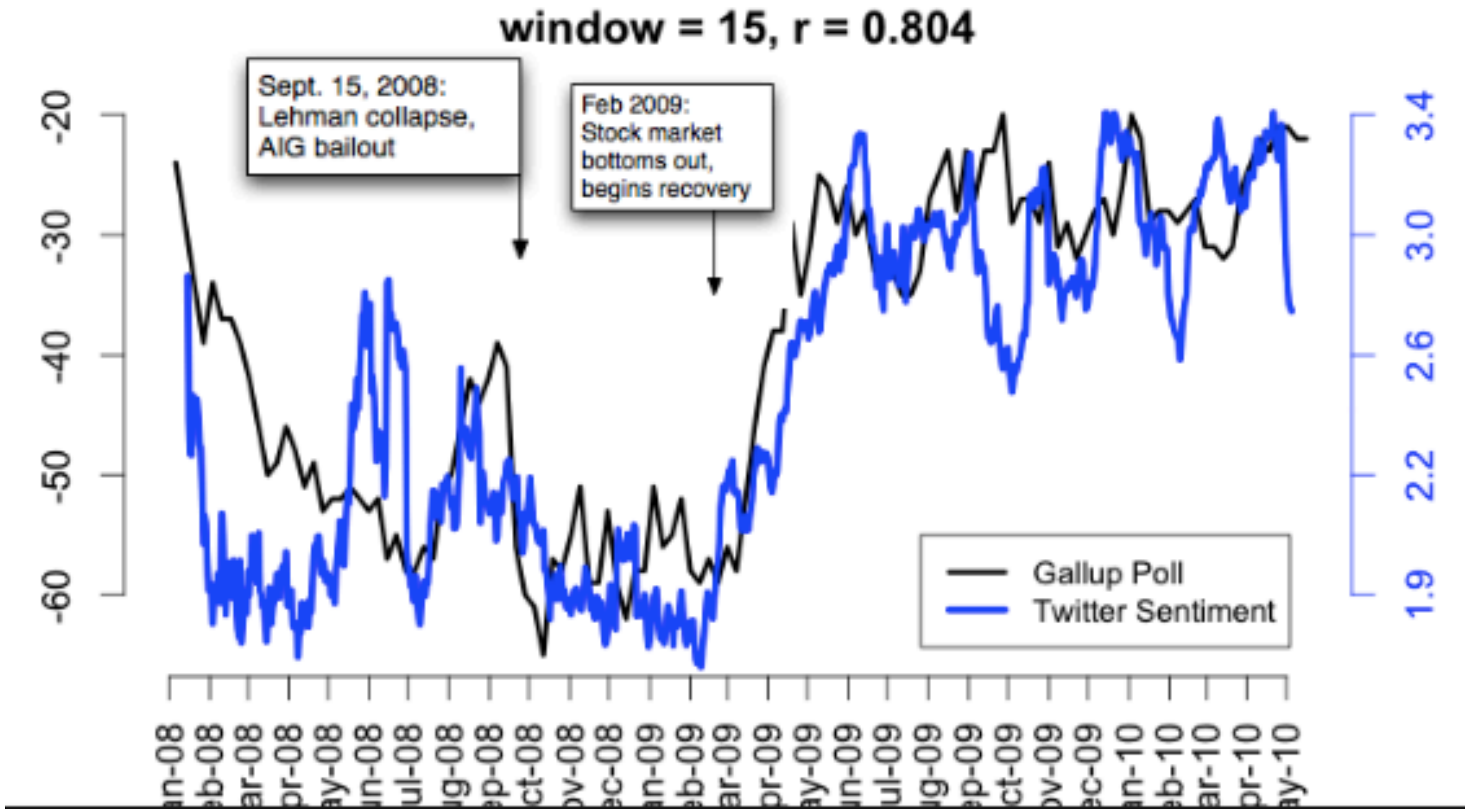
Revenues from the natural language processing (NLP) market worldwide from 2017 to 2025 (in million U.S. dollars)

NLP Use Cases

1. Alexa, Siri (Intelligent Voice Driven Interface)
2. Identifying fake news
3. Spam Filters
4. Sentiment Analysis from Social Media (Opinion mining)
5. Chatbots (Stanford's Woebot – therapist)
6. Creditworthiness assessment
7. Advertising and Marketing
8. Hiring and recruitment

Twitter sentiment versus Gallup Poll of Consumer Confidence

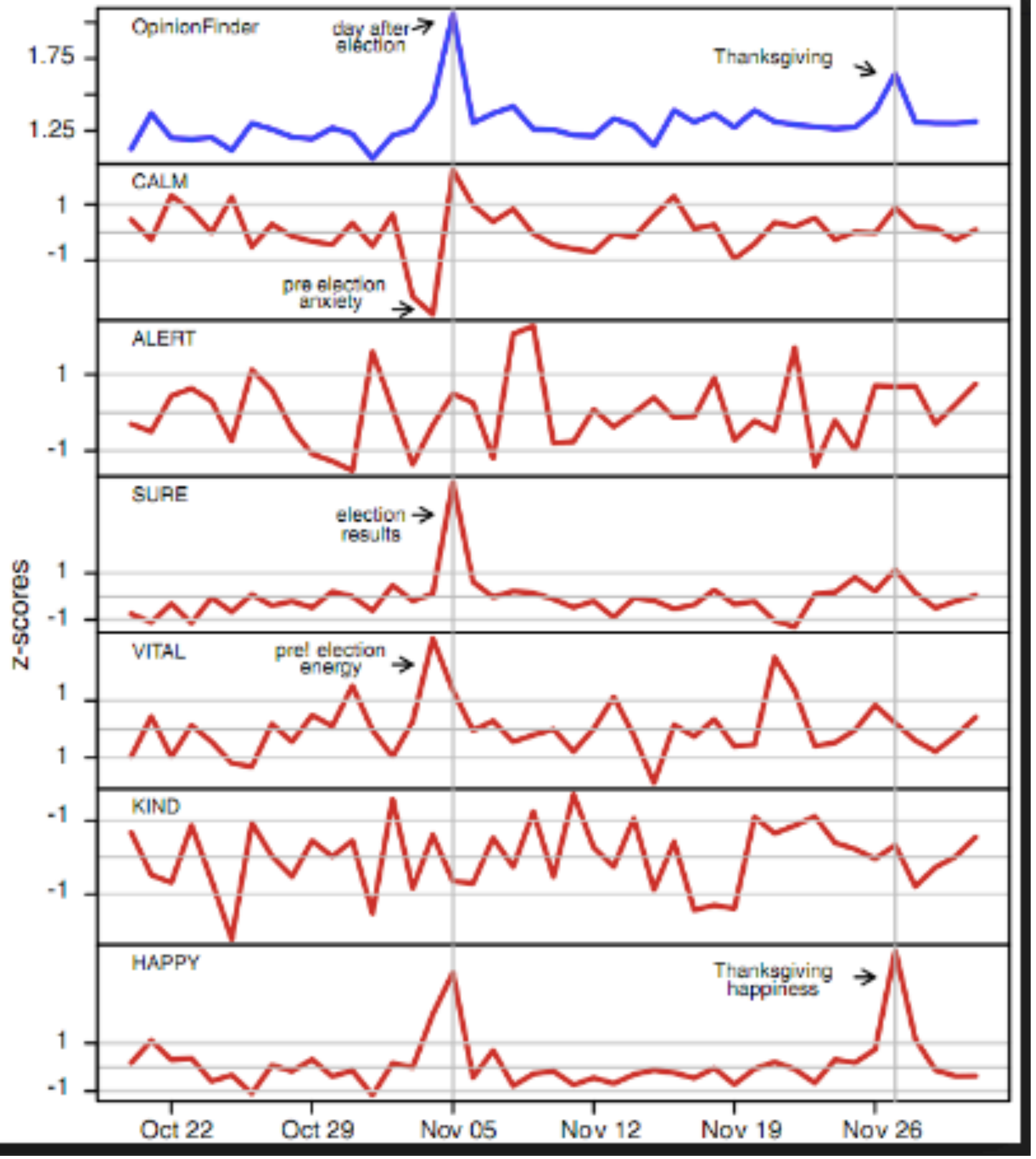
Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In ICWSM-2010



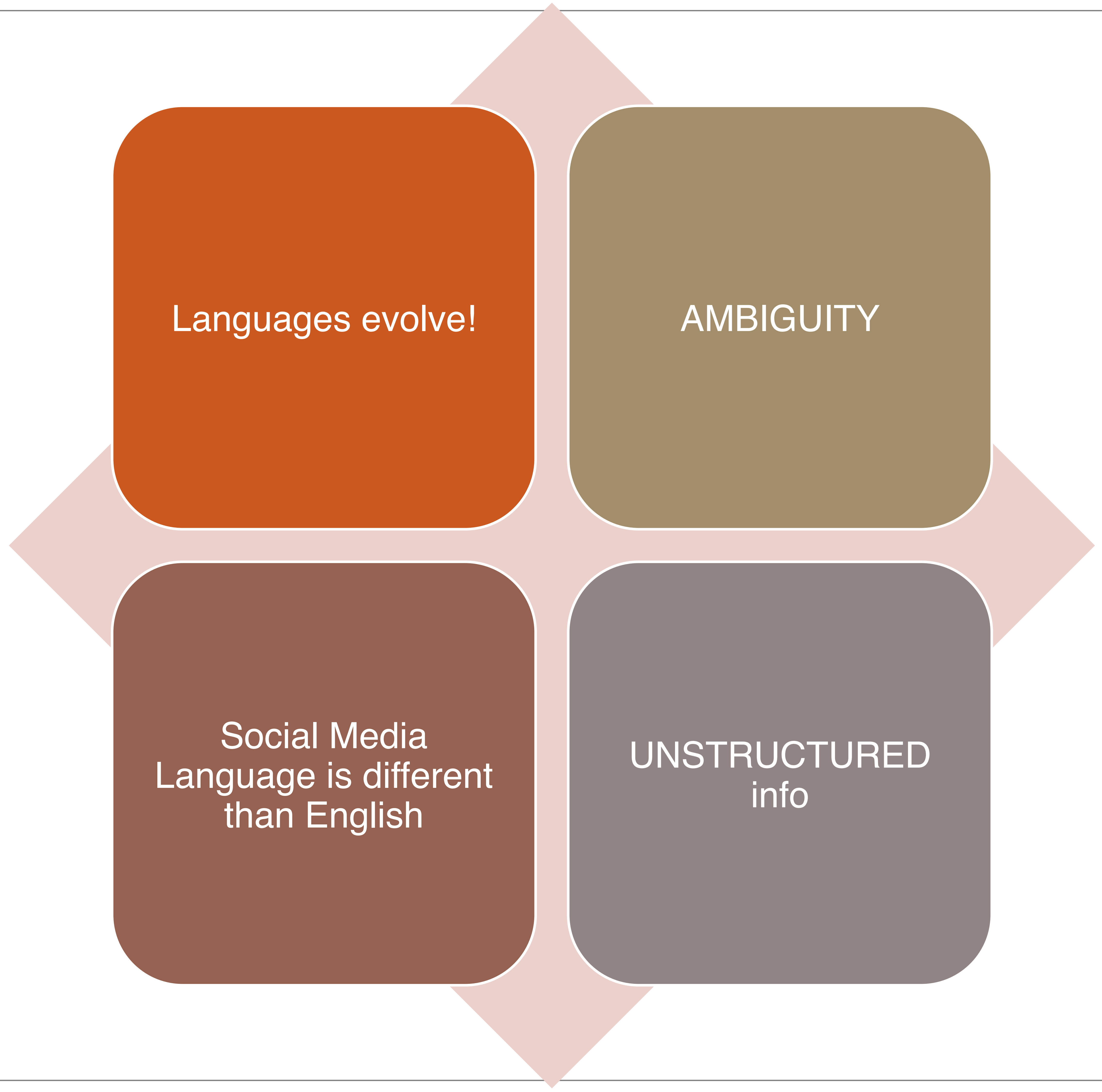
Twitter sentiment:

Johan Bollen, Huina Mao, Xiaojun Zeng. 2011. [Twitter mood predicts the stock market,](#)

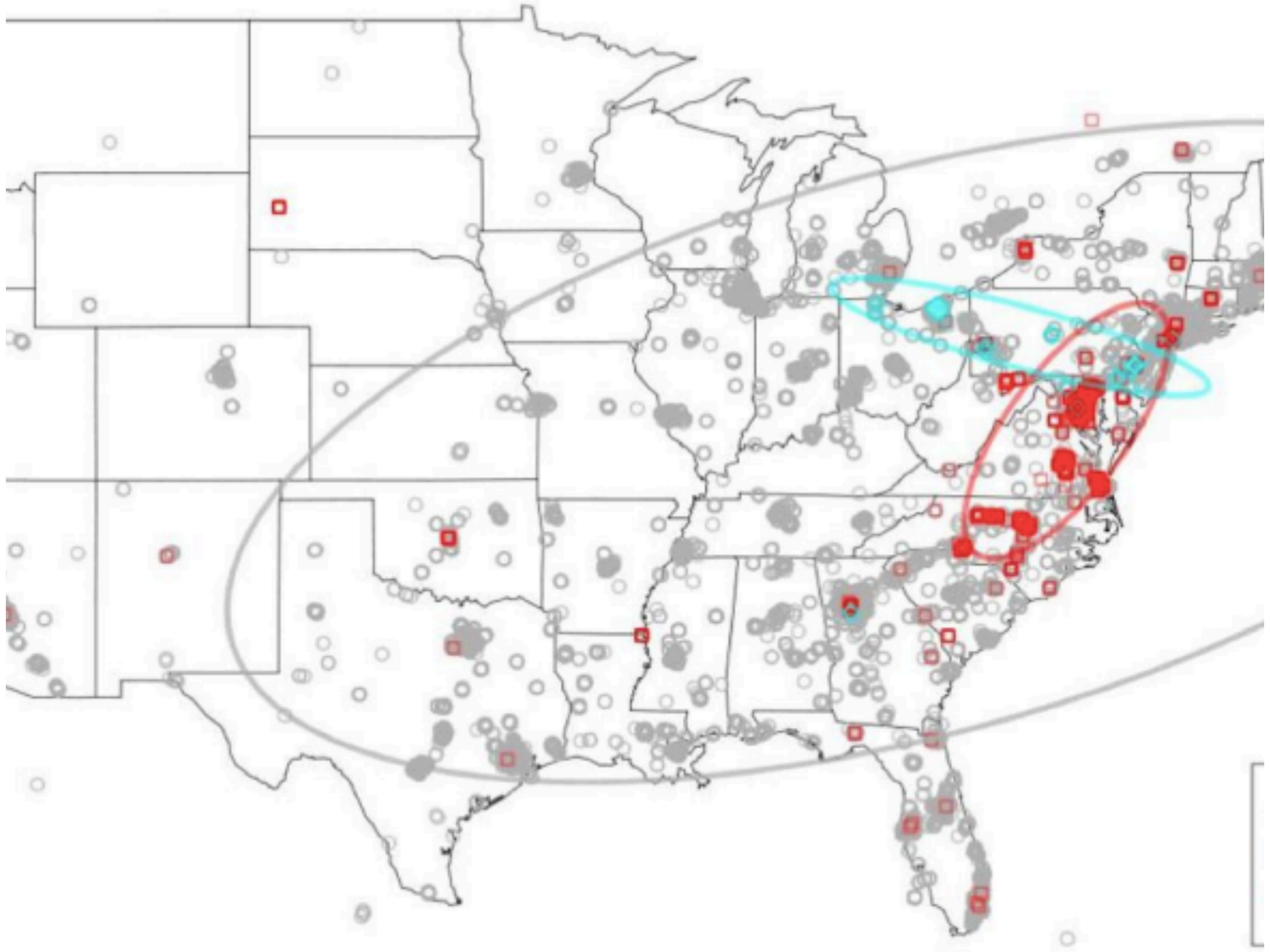
Journal of Computational Science 2:1, 1-8. 10.1016/j.jocs.2010.12.007.



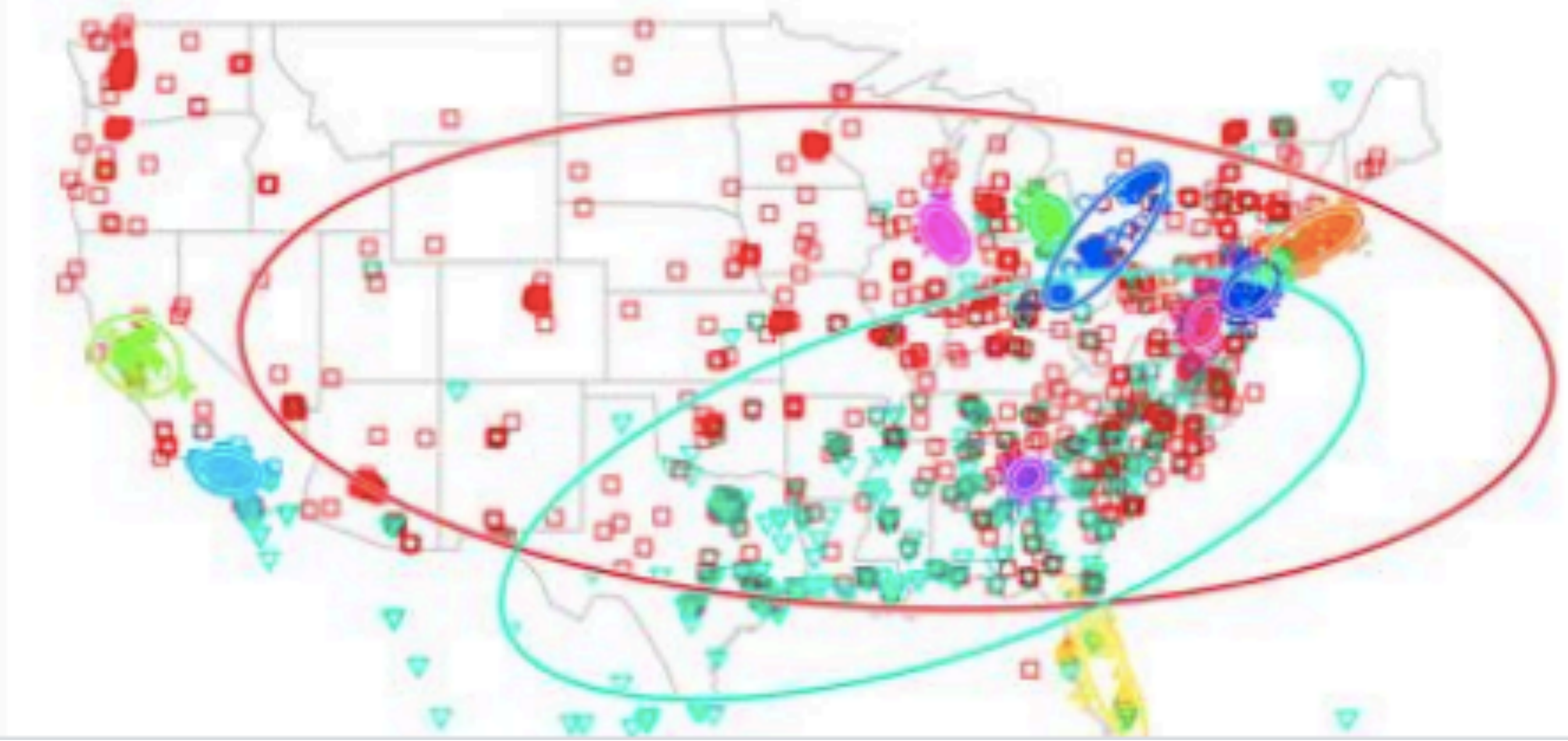
What Makes NLP Hard?



graphic Variation, Slang



	"basketball"	"chit chat"
	PISTONS KOBE LAKERS game DUKE NBA CAVS STUCKEY JETS KNICKS	lol smh jk yea wyd coo ima wassup somethin jp
Boston	CELTICS victory BOSTON CHARLOTTE	*ese* exam suttin sippin
N. California	THUNDER KINGS GIANTS pimp trees clap	hella flirt hut iono OAKLAND
New York	NETS KNICKS	wasssup nm
Los Angeles	#KOBE #LAKERS AUSTIN	wyd coo af *nada* tacos messin fasho bomb

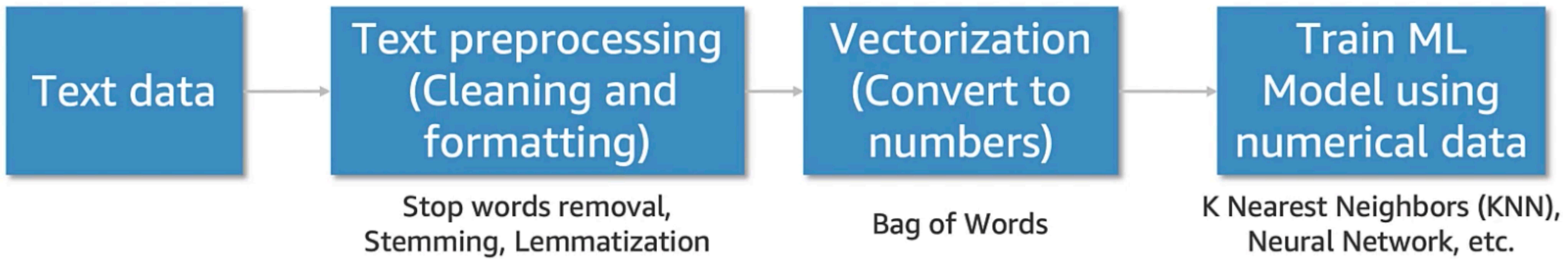


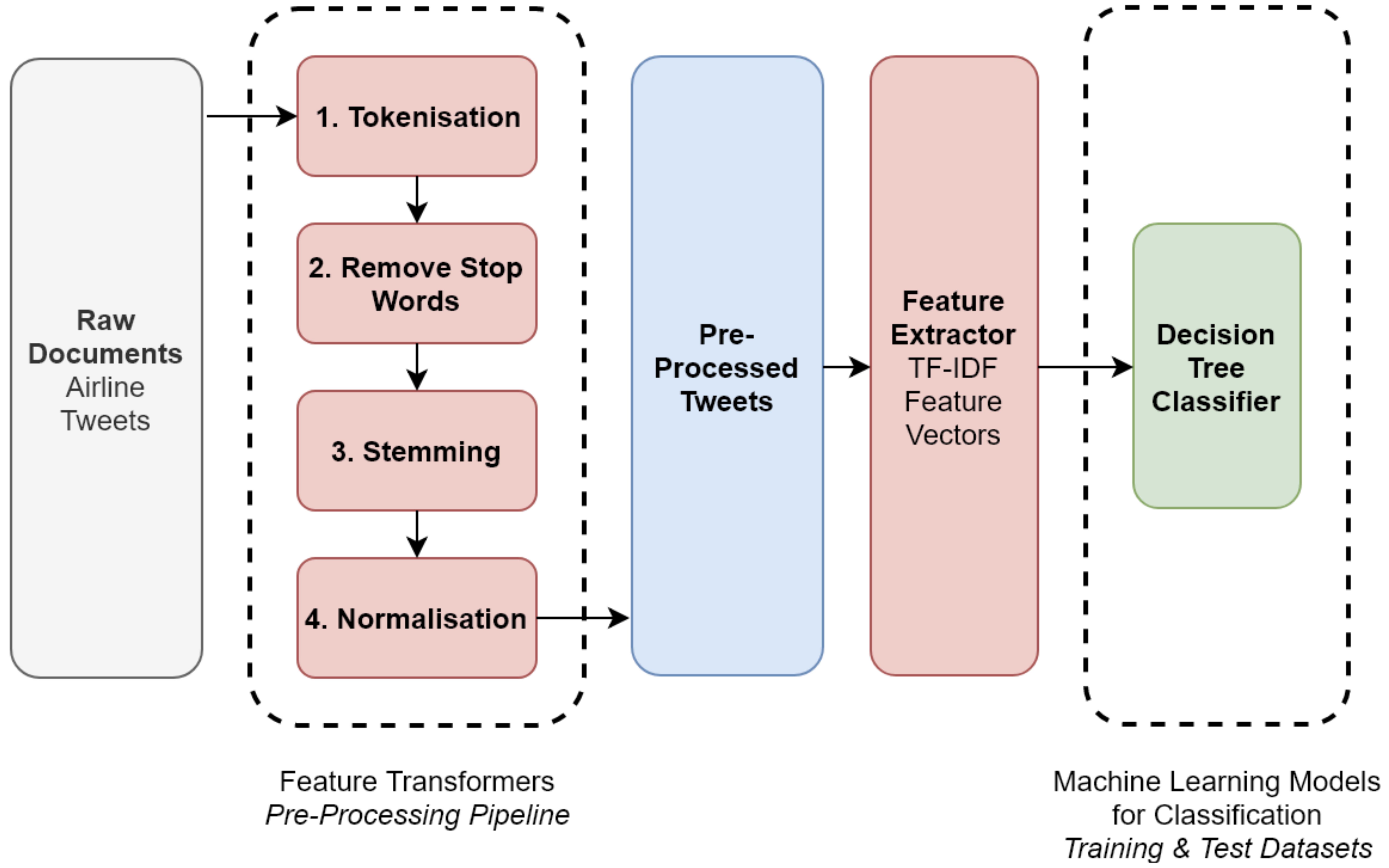
Microtext Features

- Highly relaxed spelling
- Reliance on emoticons
- Out-of-vocabulary (OOV) words
- Phonetic spellings (b4 for before)
- Emotional emphasis (Cooooooool)
- Popular acronyms (OTW – On the way)

Machine Learning with Text Data

ML models need **well-defined numerical data.**

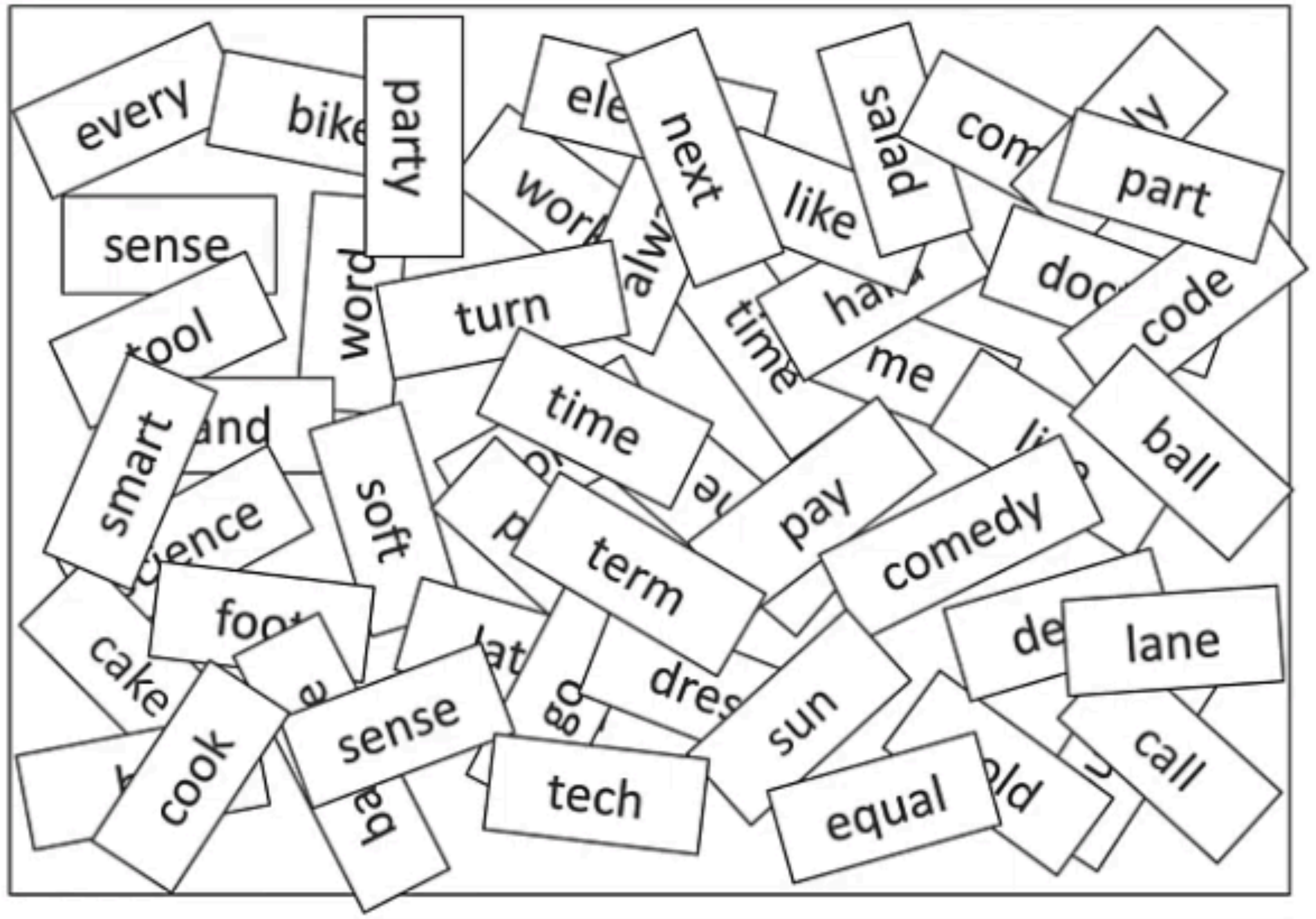



Raw Documents
Airline Tweets
1. Tokenisation
2. Remove Stop Words
3. Stemming
4. Normalisation
Pre-Processed Tweets
Feature Extractor
TF-IDF Feature Vectors
Decision Tree Classifier
Feature Transformers
Pre-Processing Pipeline
Machine Learning Models for Classification
Training & Test Datasets

Some NLP related words

- **Corpus**: Large collection of words or phrases - can come from different sources: documents, web sources, database
  - Common Crawl Corpus: web crawl data composed of over 5 billion web pages (541 TB)
  - Reddit Submission Corpus: publicly available Reddit submissions (42 GB)
  - Wikipedia XML Data: complete copy of all Wikimedia wikis, in the form of wikitext source and metadata embedded in XML. (500 GB)
  - Etc.

Some NLP Terms



Token: Words or phrases extracted from documents

Tokenization

Splits text/document into small parts by white space and punctuation.

Example:

Sentence	Tokens
“I don’t like eggs.”	“I”, “do”, “n’t”, “like”, “eggs”, “.”

These tokens will be used in the next steps in the pipeline.

Stop Word Removal

Stop words: Some words that **frequently appear** in texts, but they **don't contribute too much to the overall meaning**.

- Common stop words: "a", "the", "so", "is", "it", "at", "in", "this", "there", "that", "my"
- **Example**:

Original sentence	Without stop words
"There is a tree near the house"	"tree near house"

- Stemming : chopping the affixes

  compressed -> compress

  compression -> compress

    Porter's Stemmer

- Lemmatization : reducing the words to their base forms

  am, is, are -> be

  car, cars, car's, cars' -> car

Theory of NLP

Corpus

Document – entity/unit//object

Text segmentation

Tokenization

Word - Term

Terms are features of the doc

Each term has properties

- normalized form of the term -> term.baseform
- position(s) in the doc -> term.position(s)
- frequency of the term -> term.frequency.

Converting Words to Terms

- Preprocess and normalize the words

.tolower(), stemming, lemmatization

Feature Representation: Bag of Words

A single word is a one-hot encoding vector with the size of the dictionary :(

Bag of words

- simplified and effective way to process documents by:
  - disregarding grammar (term.baseform?)
  - disregarding word order (term.position)
  - keeping only multiplicity (term.frequency)

Bag of words

- sparse matrix
- numbers can be:
  - binary - 0/1
  - simple term frequency
  - weight - e.g. TF-IDF

Documents \ Terms	Investment Risk	Project Management	Software Engineering	Development	SAP	...
Document 1	1			1		
Document 2		1				
Document 3			3		1	
Document 4		1				
Document 5			2	1		
Document 6	1			1		
...						

Word Counts in the vector

Simple example using **word counts**:

	a	cat	dog	happy	is	it	my	not	old	wolf
"It is a dog."	1	0	1	0	1	1	0	0	0	0
"my cat is old"	0	1	0	0	1	0	1	0	1	0
"It is not a dog, it a is wolf."	2	0	1	0	2	2	0	1	0	1

TF in the vector

Term frequency (TF): Increases the weight for **common** words in a document.

$$tf(term, doc) = \frac{number\ of\ times\ the\ term\ occurs\ in\ the\ doc}{total\ number\ of\ terms\ in\ the\ doc}$$

	a	cat	dog	is	it	my	not	old	wolf
"It is a dog."	0.25	0	0.25	0.25	0.25	0	0	0	0
"my cat is old"	0	0.25	0	0.25	0	0.25	0	0.25	0
"It is not a dog, it a is wolf."	0.22	0	0.11	0.22	0.22	0	0.11	0	0.11

TF-IDF

term	idf
a	log(3/3)+1=**1**
cat	log(3/2)+1=**1.18**
dog	log(3/3)+1=**1**
is	log(3/4)+1=**0.87**
it	log(3/3)+1=**1**
my	log(3/2)+1=**1.18**
not	log(3/2)+1=**1.18**
old	log(3/2)+1=**1.18**
wolf	log(3/2)+1=**1.18**

Inverse document frequency (IDF): Decreases the weights for **commonly** used words and **increases** weights for **rare** words in the vocabulary.

$$idf(term) = log\left(\frac{n_{documents}}{n_{documents\ containing\ the\ term} + 1}\right) + 1$$

$$e.g.\ idf("cat") = 1.18$$

Term Freq. Inverse Doc. Freq (TF-IDF): Combines term frequency and inverse document frequency.

$$tf_{idf}(term, doc) = tf(term, doc) * idf(term)$$

	a	cat	dog	is	it	my	not	old	wolf
"It is a dog."	**0.25**	0	**0.25**	**0.22**	**0.25**	0	0	0	0
"my cat is old"	0	**0.3**	0	**0.22**	0	**0.3**	0	**0.3**	0
"It is not a dog, it a is wolf."	**0.22**	0	**0.11**	**0.19**	**0.22**	0	**0.13**	0	**0.13**

Tf-Idf

Term Frequency — Inverse Document Frequency

a technique to quantify a word in documents based on its relevancy

we generally compute a weight to each word which signifies the importance of the word in the document and corpus

This method is a widely used technique in Information Retrieval, keyword extraction, and Text Mining

◦ **TF-IDf** = Term Frequency (TF) * Inverse Document Frequency (IDF)
(0:no word in the doc, 1:the doc has only the word)

◦ **Tf** -> Term frequency : frequency of a word in a doc
Tf = count of t in doc/count of the words in the doc

◦ **Df** -> Document frequency: count of t in the document set N
(normalize df – divide with the number of documents)

Idf = N/df

Idf = log(N/(df+1))

Tf-Idf = tf / log(N/(df+1))

Example of Tf-IDf

- Tf- Consider a document containing 100 words wherein the word *cat* appears 3 times. The term frequency (i.e., tf) for *cat* is then (3 / 100) = 0.03.

- Idf- Now, assume we have 10 million documents and the word *cat* appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4.

- Tf-Idf: Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12.

Basic string manipulation

- keep it simple and stupid

```
.lower(), .strip(), .split(), .join(),
iterators, …
```

- regexp
  - not only match, but transformation, extraction (\1), backreferences etc.
  - re.options, re.multiline, repl can be function:

```
def repl(m): …

re.sub("pattern", repl, "string")
```

spaCy

- Industrial strength
- Faster than NLTK, CoreNLP, ZPar…
- Easy to install, simple
- Interoperates with Tensorflow, Keras, Scikit-Learn, Gensim

Stanford NLP

- http://nlp.stanford.edu/software/index.shtml
- statistical NLP, deep learning NLP, and rule-based NLP tools for major computational linguistics problems
- famous
- Java

Scikit-Learn

http://scikit-learn.org/stable/index.html

machine learning in python

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM*, *nearest neighbors*, *random forest*, ... — *Examples*

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: *SVR*, *ridge regression*, *Lasso*, ... — *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means*, *spectral clustering*, *mean-shift*, ... — *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA*, *feature selection*, *non-negative matrix factorization*. — *Examples*

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search*, *cross validation*, *metrics*. — *Examples*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing*, *feature extraction*. — *Examples*