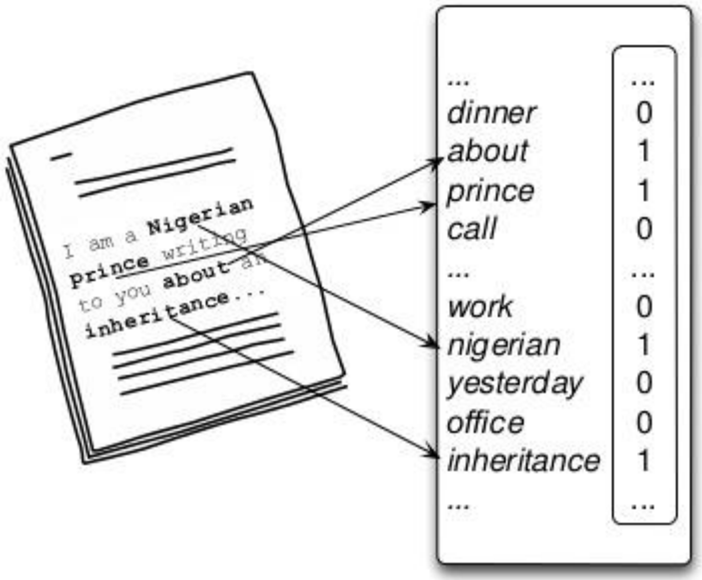
From Text to Vectors

Words and Representations

Snorre Ralund, Ph.D Fellow, SoDaS, UCPH

Text to Vector

bag of words



Text to Vector

Document Term Matrix – BoW (Bag of Words)

- Word counts as columns.

--> map word to an index in a matrix.

- Throw out word order.

Text to Vector

Document Term Matrix – BoW (Bag of Words)

- doc1: *"i really do like bacon"*
- doc2: *"i really do not like bacon"*

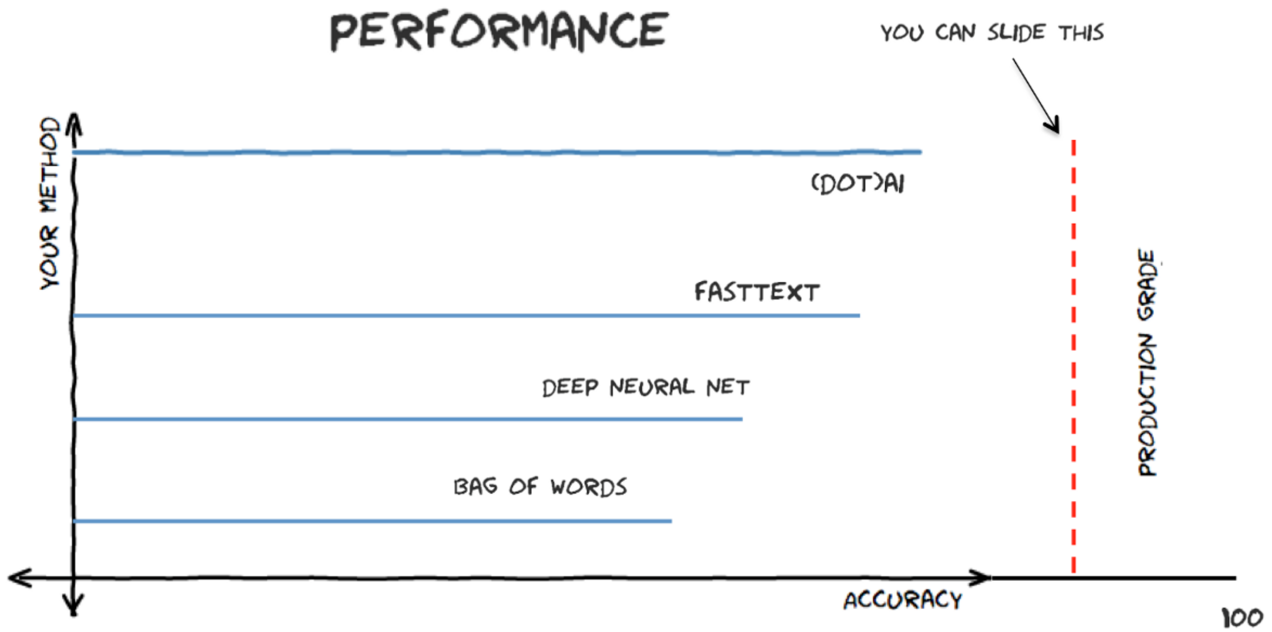
Text to Vector

Document Term Matrix – BoW (Bag of Words)

- doc1: *"i really do like bacon"*

- doc2: *"i really do not like bacon"*

	really	do	like	i	bacon	not
doc1	1	1	1	1	1	0
doc2	1	1	1	1	1	1



Baseline models: Naive Bayes, Logistic Regresion, K-nearest Neighbor and Support Vector Machines

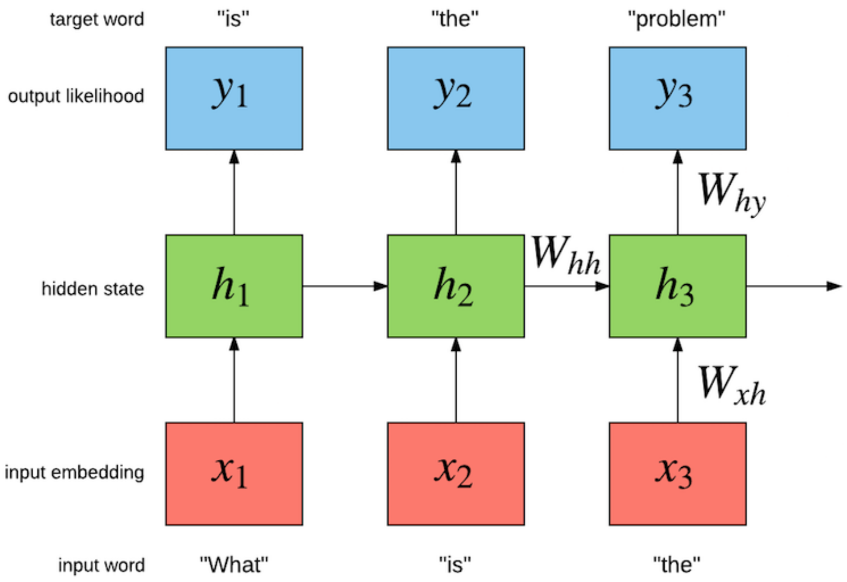
- **Wang and Manning 2012** *"Baselines and Bigrams: Simple, Good Sentiment and Topic Classification"*:
    - State-of-the-art (2012) Topic and Sentiment Classification using only atomized Words as input(BoW) to a linear model.
    - No grammar or reasoning.

Bag of Words(2) - Problem with polesemy

Consider the following to documents:
doc1: *"River A/S declared default by the bank."*
doc2: *"When camping my default is by the river bank."*

document	declared	by	default	bank	river	a/s	when	camping	my
doc1	1	1	1	1	1	1	0	0	0
doc2	0	1	1	1	1	0	1	1	1

Bag of Words(2) - Problem with polesemy



bank		*camping*		*Hidden state*
-1		**0**		**0**
-2		0		0
1	×	3	=	3
2		5		10
0		1		0

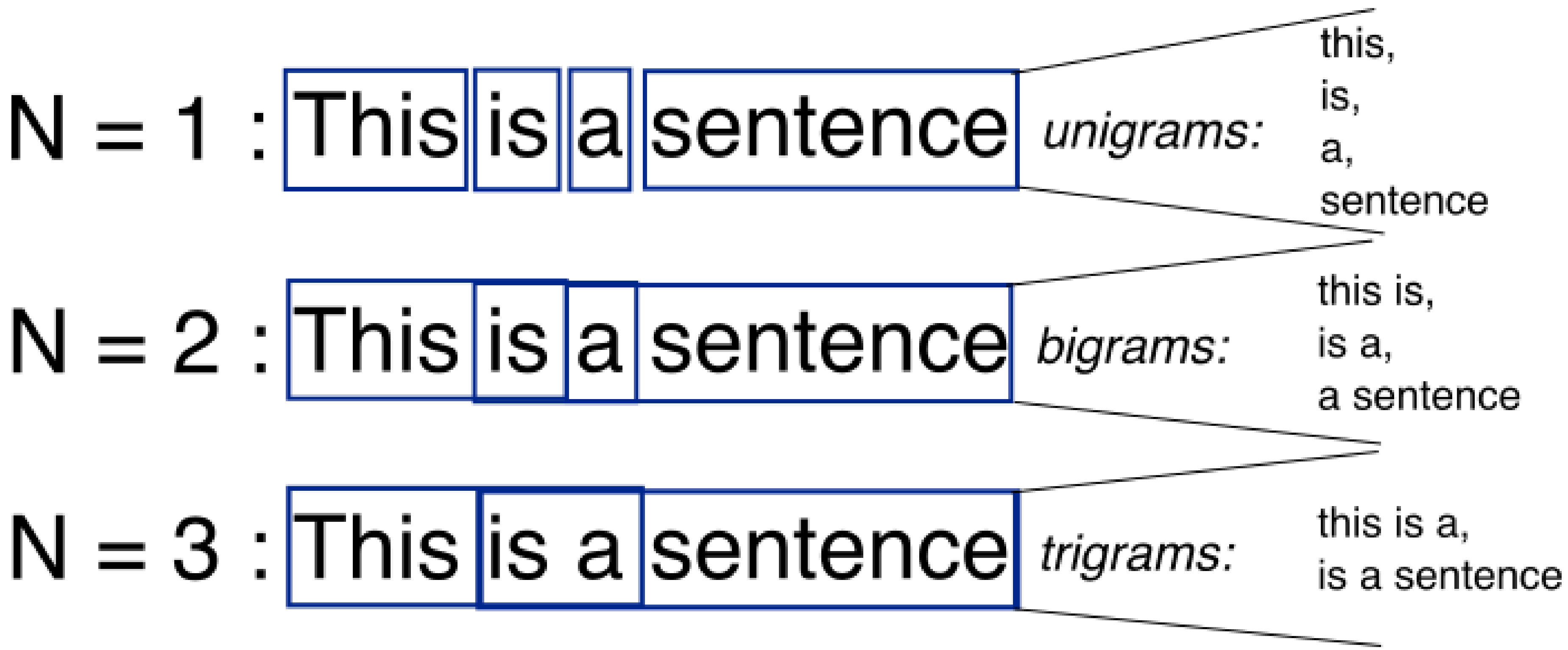
Bag of Words (3) - Lack of word order¶

doc1 = 'this was not the best movie'

doc2 = 'was this not the best movie ever?'

Will have very similar representations.

Ngrams to the Rescue



Ngrams to the Rescue(2): Problem with dimensionality

- Quad-grams Qvint Grams etc. Generates exponential number of features.

Solution

- Pick only the ngrams using statistical analysis of the word co-occurences.
- Check out methods for doing so in the Natural Language Processing Toolkit package nltk: `nltk.collocations` and the package `sumgram`


WAIT A MINUTE...
makeameme.org

```
"What are the boundaries between-words and meaning?".split('-')

['What are the boundaries between', 'words and meaning?']
```

Tokenization (1)¶

- How we "split" / locate words in text determines the number of dimensions (columns in the Document-Term Matrix)

Issues

- Computational inefficiency
- Parameters are not shared among equivalent words.
  - It makes a difference especially for low N tasks.
  - E.g. run, ran, runs will not share parameters.

Tokenization (2)¶

Issues

- Spelling mistakes, or weird uses of punctuation
  - fuzzy matching: ```fuzzymatching package```
- Emoticons: </3 , (:) , :-]
- Multiwords: #no-more-work, New York, Federal Bureau of Finance, word/concept
  → Collocations: ```sumgram package```

Representation (1)

How to encode all relevant information in our tokens?

- lower-casing: DO YOU REALLY WANT TO IGNORE MY ALLCAPS?!?!
  - Our featurespace can potentially double if we don't lowercase.
- Numbers: Infinite combinations of digits
- Filtering to reduce dimensions: Which words to lose?
  - Common or Rare?
  - TF-IDF

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Representation (2)

Grammatical Forms: Do we need grammar?

NO

- Stemming
  - Rulebased: Strips common endings: 'ing','ly','s'
- Lemmatization
  - Lookup in Lexical Ressources(e.g. WordNet): ran --> run, running --> run

Trade-off precision and coverage

Representation (3)

Grammatical Forms: Do we need grammar?

YES

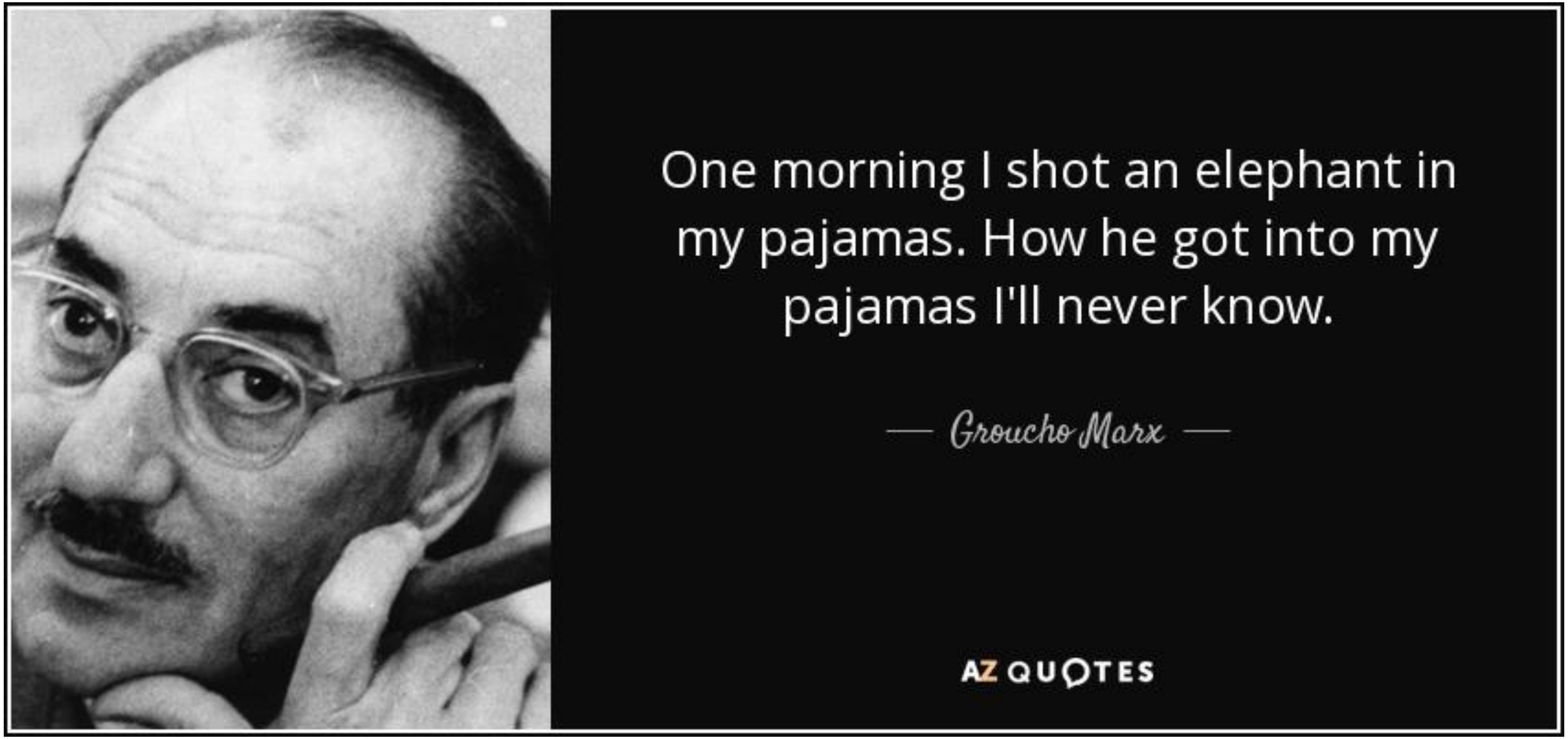
- Use NLP systems for parsing the text

Implementations

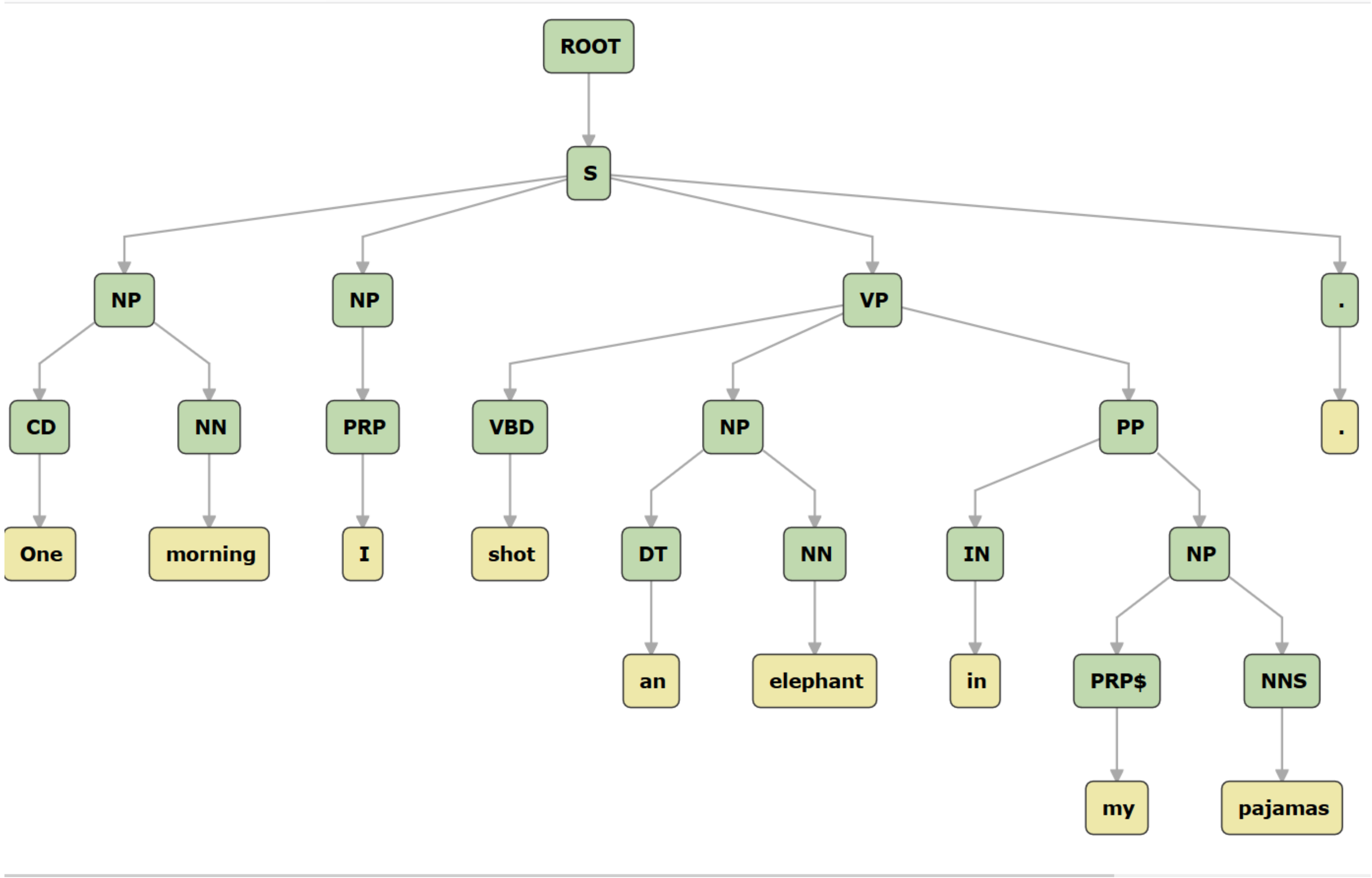
stanfordnlp package: https://stanfordnlp.github.io/stanfordnlp/

SpaCy package: https://spacy.io/

Representation: Wordsense and Relationship Parsing

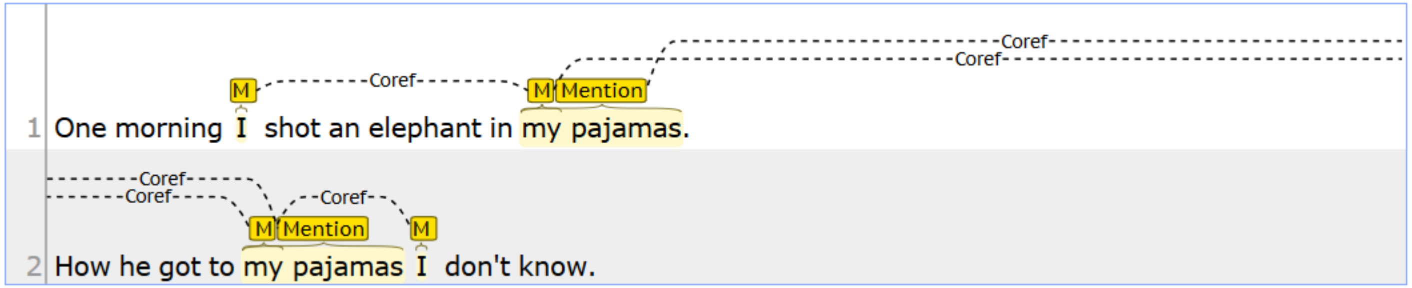


Wordsense and Relationship Parsing

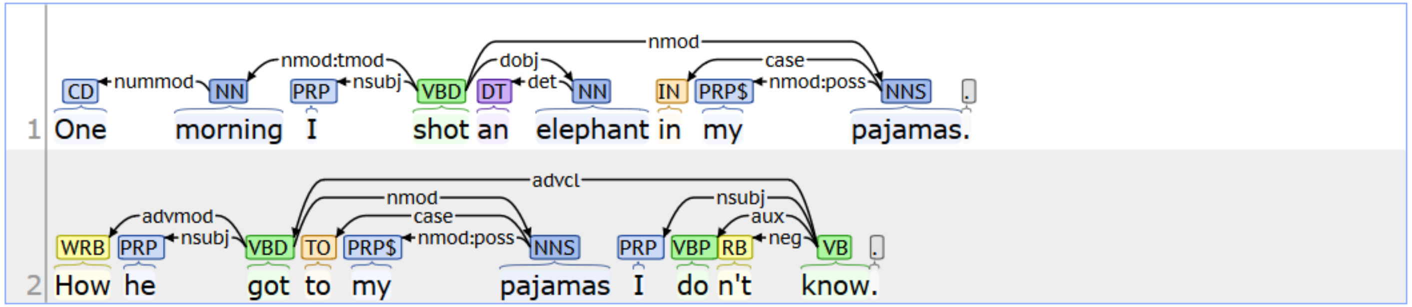


Representation: Wordsense and Relationship Parsing

Coreference:



Basic Dependencies:



Wordsense and Relationship Parsing

- Now we can distinquish between “to run” “a run”.
- We have now encoded that “Worst” and “Worse” has the same root lemma.

- Furthermore it can be used to create simple rules about **subject-verb-object** relations (more on this later).
  - Relationships: He (SUBJECT) loves (VERB) her (OBJECT).
  - Simple rule: Unrequitted Love: John (SUBJECT) loves (VERB) Jane (OBJECT) + Absence of reciprocity or Jane (SUBJECT) loves (VERB) Joe (OBJECT)

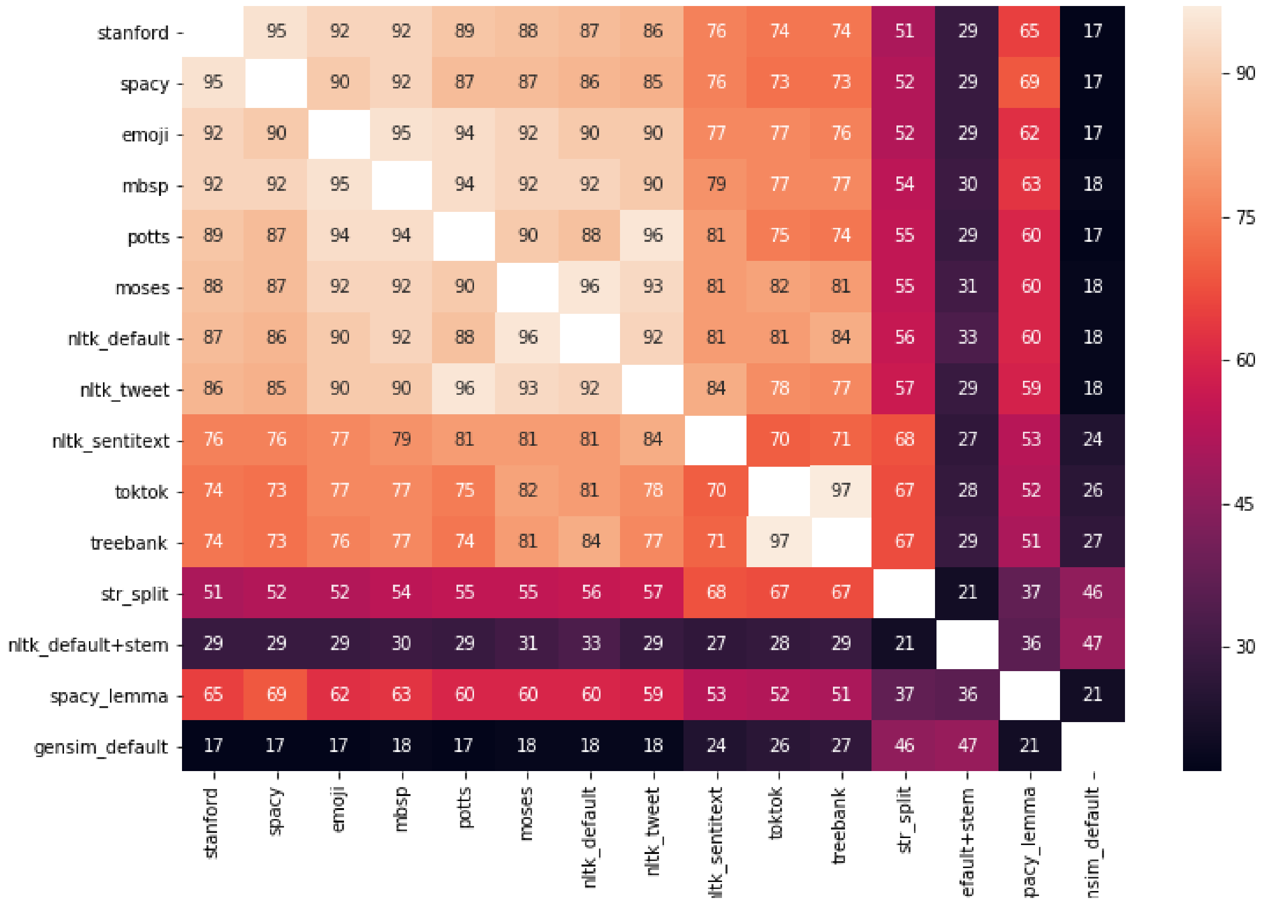

WAIT A MINUTE...
makeameme.org

Tokenization: Choosing a tokenizer

- Build around regular expressions assuming whitespace as separator.
  - e.g. ‘[a-z]+|[0-9]+|[^\sa-z0-9]’
- Hardcoding complex rules to capture:
  - Emojies. E.g. *=-) <3*
  - Abbreviations e.g. *Ph.D.*
  - Formulas*: e.g.* $C_{11}H_{12}N_2O_2$ $C_{13}H_{16}N_2O_2$ $C_8H_{11}NO_2$ $C_{28}H_{44}O$ – State-of-the-art

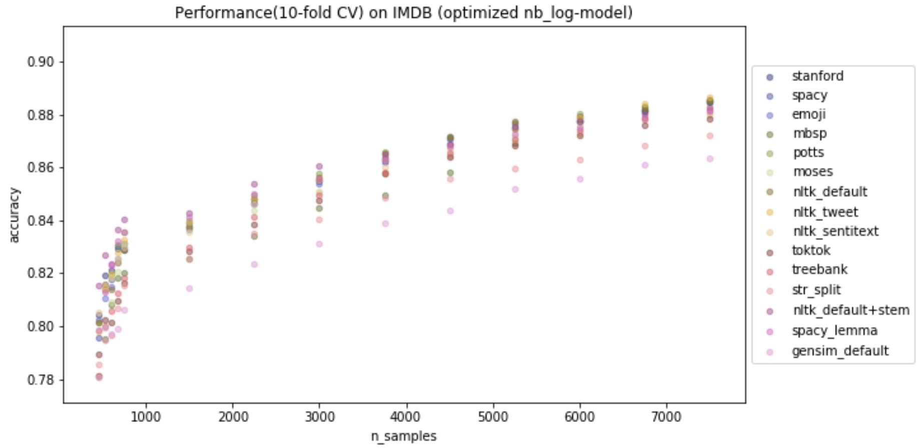
"Klatret©ªsen(Catch That Girl) is really great movie! It's a 'happy' movie. I watched this movie in 'Puchon International Fantastic Film Festival(PiFan)' on July 12nd, 2003. There is Action + Adventure + Comedy + Thrill + Happy + Romance(cute kids' love Triangle!). You must see this movie. :)"

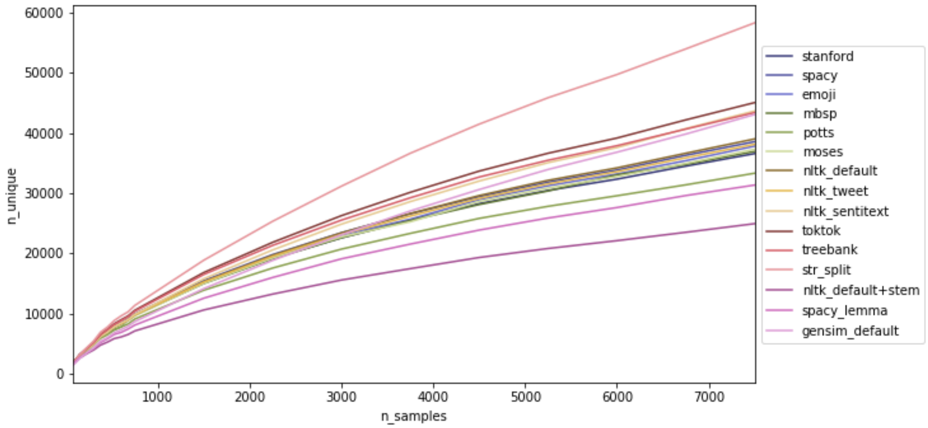
Tokenization: Choosing a tokenizer

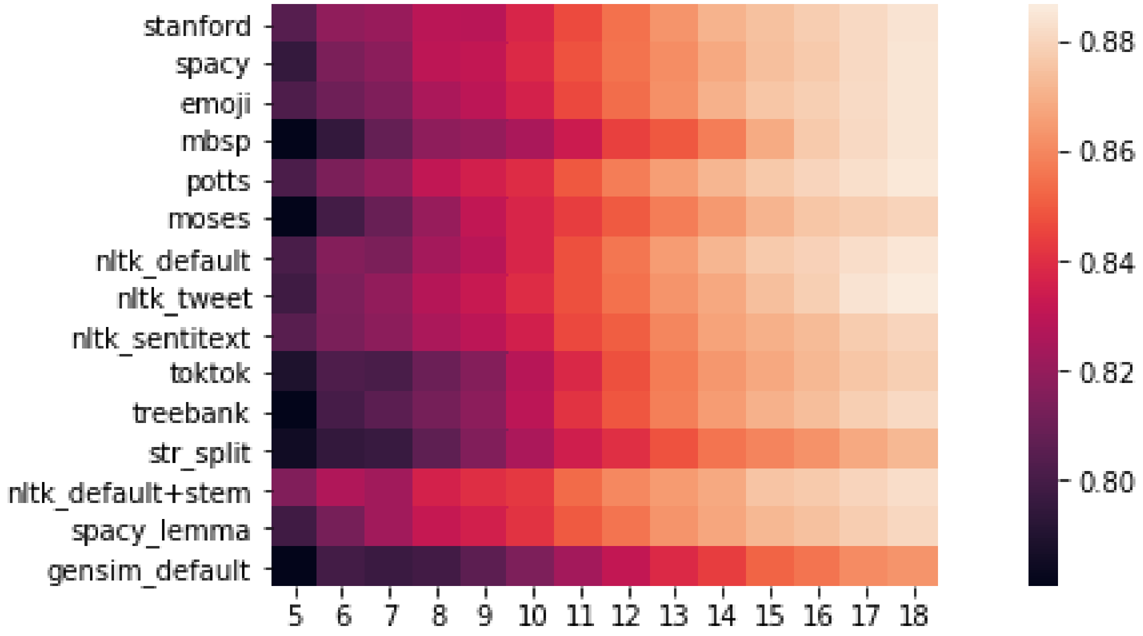


Klatret©ªsen(Catch That Girl) is really great movie! It's a 'happy' movie. I watched this movie in 'Puchon International Fantastic Film Festival(PiFan)' on July 12nd, 2003. There is Action + Adventure + Comedy + Thrill + Happy + Romance(cute kids' love Triangle!). You must see this movie. :)

Tokenization: Choosing a tokenizer







Tokenization: Choosing a tokenizer

For user generated content and social media data use:

- If enough data: nltk.tokenize.causal.tweet
- If smaller data: spacy or stanfordnlp lemmatizer.

For more formal text (e.g. scientific articles) test a few others.

```
import nltk
tweet_tokenizer = nltk.tokenize.casual.TweetTokenizer()
tweet_tokenizer.tokenize('hello I speak emoticon and #hashtag :)'
```

Subword tokenization

Problems

- Dimensionality constrains (many many words) and unseen words in relation a fixed Vocabulary
- Words can be compositional: e.g. “Speciallægepraksisplanlægningsstabiliseringsperiode”
- Grammer: You want to be able to learn similarities between ‘worse’ and ‘worst’.

Subword tokenization

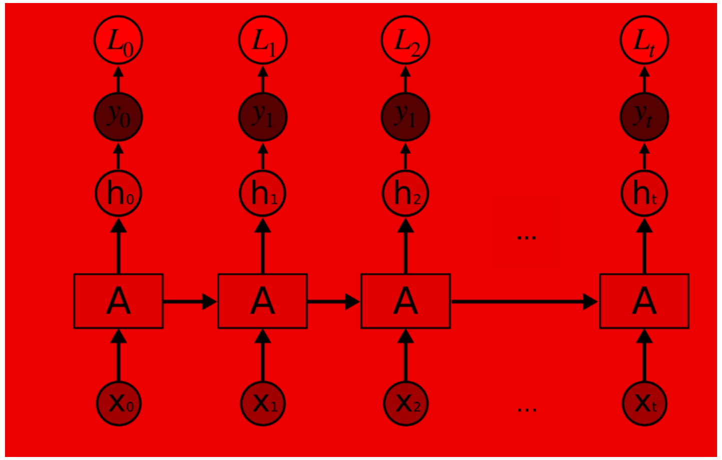
Problems

- Dimensionality constrains (many many words) and unseen words in relation a fixed Vocabulary
- Words can be compositional: e.g. “Speciallægepraksisplanlægningsstabiliseringsperiode”

Subword tokenization

Solutions

- Characterbased LSTM
  - Can work, but has long training times. E.g. ":https://openai.com/blog/unsupervised-sentiment-neuron/
  - Maybe the "embedding" of a character has to serve too "heterogenous" purposes / i.e. complex combinatorial transformations.
    - Remember the Idea of a RNN.



Subword tokenization

Solutions

- Subword tokenization
  - Preprocess by extracting subwords as highly co-occuring characters sequences.
- BytePairEncoding: Seinrich et. al 2016 “Neural Machine Translation of Rare Words with Subword Units”.
  - “is a simple data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with a single unused byte”.

Example: Subword tokenization of a RAP song

Example: Subword tokenization of a rap song

"I said a hip, hop, the hippie, the hippie, to the hip hip-hop, and you don't stop".

- "**op**","**ip**" and "**he**" most frequent.

→ substitute for byte 0 1 and 2

"I said a h0, h1, t2 h0pie, t2 h0pie, to t2 h0 h0-h1, and you don't st1"

- Now "**0p**" most frequent.
- "I said a h0, h1, t2 h3ie, t2 h3ie, to t2 h0 h0-h1, and you don't st1". "3i" most frequent.