# BoW Models

Vsevolod Dyomkin

prj-nlp

2020-04-02

# Contents

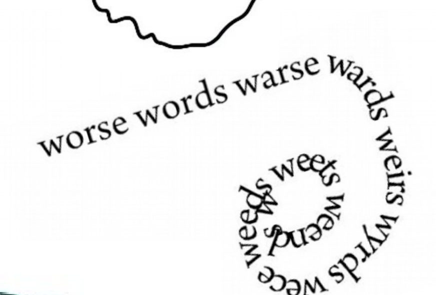
* BoW Concept
* Generative & Discriminative Models
* AP & NB Learning Algorithms
* Feature Collection, Stopwords
* A Number of Practical Examples
* Simple Text Similarity Measures
* TF-IDF

# NLP Viewpoints

bag-of-words

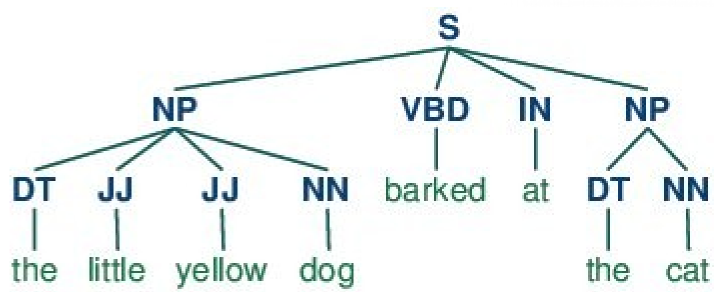

lexics

sequence



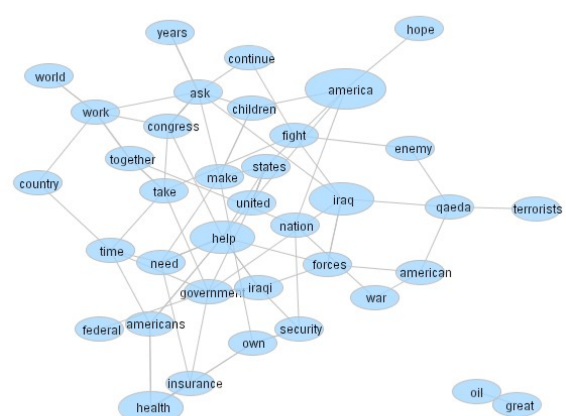

discourse

tree




syntax

graph



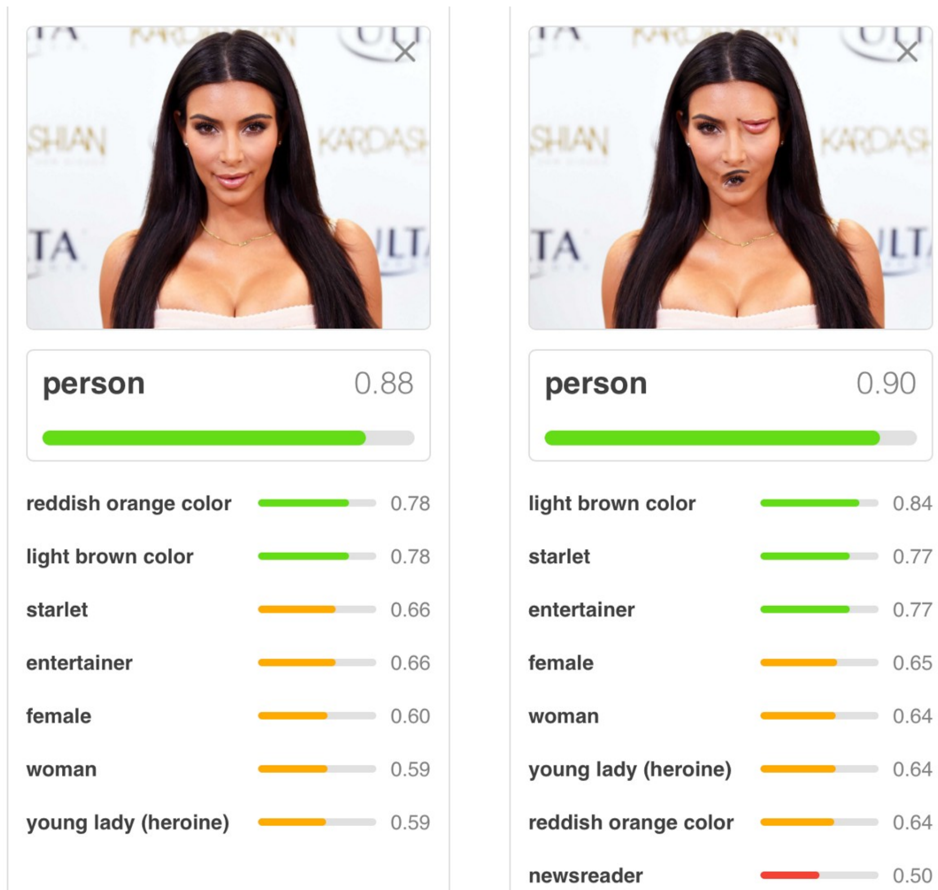

semantics

# The Glorious BoW

* Simplest model
* Feature vector in N-dim space - vector of words (with or w/o counts) (N = dictionary size) - a.k.a **1-hot representation**

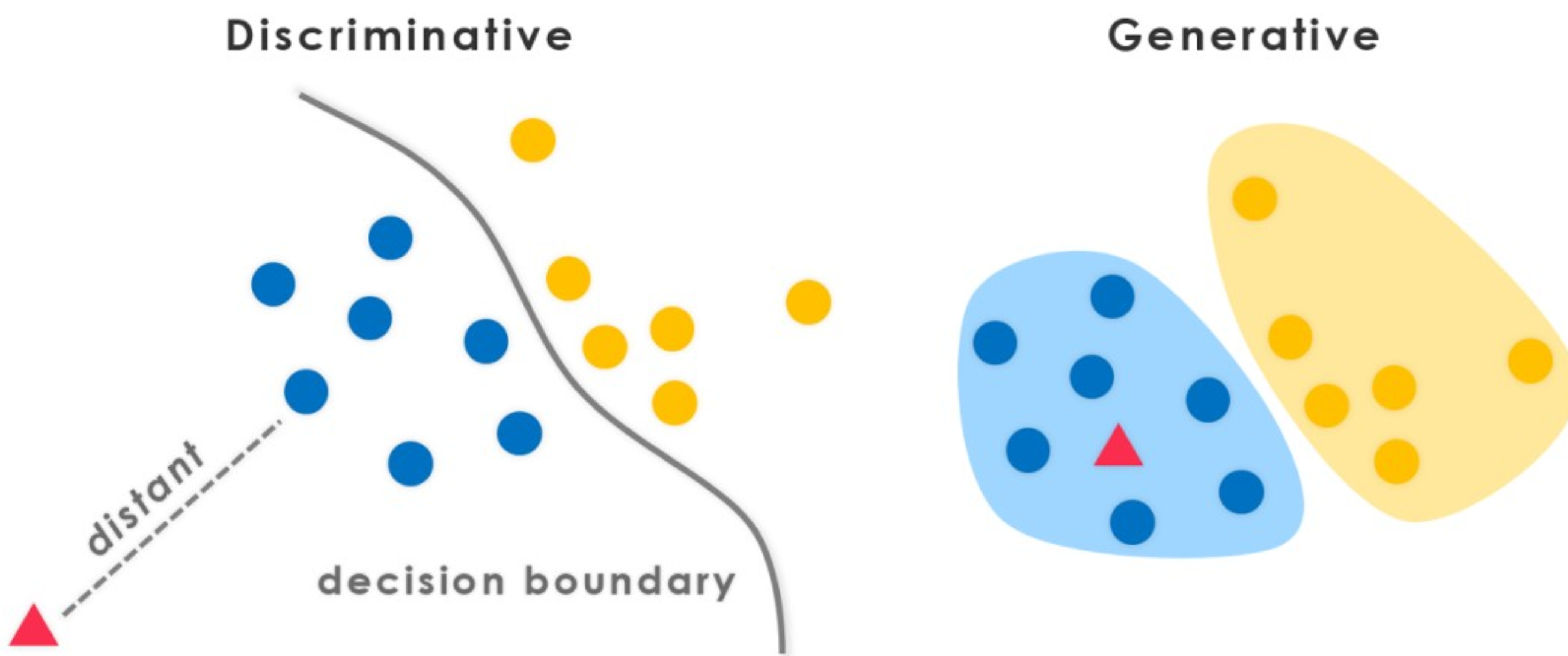

* Position information disregarded
* Works mostly for c12n

# … not only for text




https://hackernoon.com/capsule-networks-are-shaking-up-ai-heres-how-to-use-them-c233a0971952

# Generative vs Discriminative ML Models



# Generative Models

* Model joint probability of a sample and label:
  can be used both to classify
  and generate
* Introduce some structure (constraints)
* That’s why accuracy is usually asymptotically lower (but learn faster)
* Examples:
  - Naive Bayes
  - GMM
  - HMM
  - PCFG
  - GAN

# Generative vs Discriminative Models

https://stats.stackexchange.com/questions/12421/generative-vs-discriminative

a) The generative model does indeed have a higher asymptotic error (as the number of training examples become large) than the discriminative model, but
b) The generative model may also approach its asymptotic error much faster than the discriminative model — possibly with a number of training examples that is only logarithmic, rather than linear, in the number of parameters

http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf

# Spam Identification

A 2-class whole text classification problem with a bias towards minimizing FPs.

Default approach - Rule-based (SpamAssassin)

Apache SpamAssassin

Problems:
- scales poorly
- hard to reach arbitrary precision
- hard to rank the importance of complex features
- hard to interpret score and use it in upstream calculations

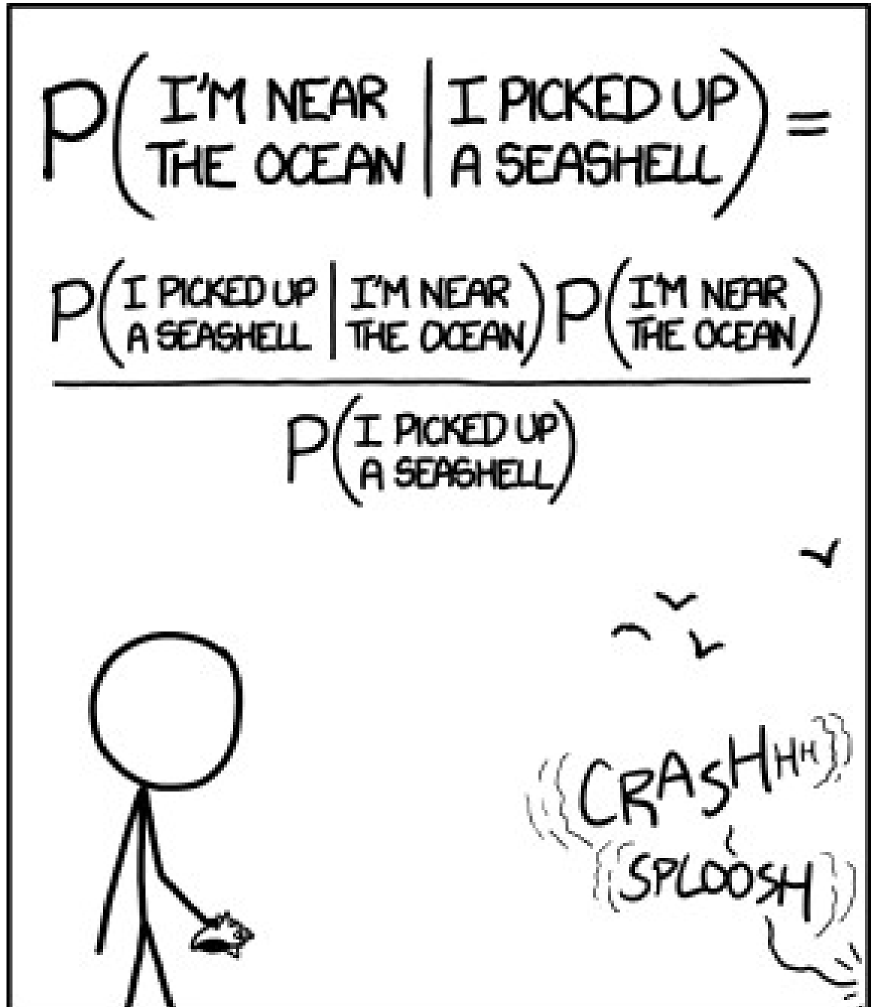
# “A Plan for Spam”

Proposed by Paul Graham
(http://www.paulgraham.com/spam.html)

1. Use the BoW approach
2. Use the Naive Bayes learning algorithm
3. Train on a balanced corpus

Initial results: Rec: 92%, Prec: 98.84%
Improved results: Rec: 99.5%, Prec: 99.97%

# Bayes Rule




STATISTICALLY SPEAKING, IF YOU PICK UP A SEASHELL AND *DON'T* HOLD IT TO YOUR EAR, YOU CAN PROBABLY HEAR THE OCEAN.

# Naive Bayes Classifier

```
P(Y|X) = P(Y) * P(X|Y) / P(X)
select Y = argmax P(Y|x)

Naive step:

    P(Y|x) = P(Y) * prod(P(x|Y))
                    for all x in X

(P(x) is marginalized out because it's the
same for all Y)
```

# NB Model for Spam

| | |
|---|---|
| madam | 0.99 |
| promotion | 0.99 |
| republic | 0.99 |
| shortest | 0.047225013 |
| mandatory | 0.047225013 |
| standardization | 0.07347802 |
| sorry | 0.08221981 |
| supported | 0.09019077 |
| people's | 0.09019077 |
| enter | 0.9075001 |
| quality | 0.8921298 |
| organization | 0.12454646 |
| investment | 0.8568143 |
| very | 0.14758544 |
| Valuable | 0.82347786 |

| | |
|---|---|
| perl | 0.01 |
| python | 0.01 |
| tcl | 0.01 |
| scripting | 0.01 |
| morris | 0.01 |
| graham | 0.01491078 |
| guarantee | 0.9762507 |
| cgi | 0.9734398 |
| paul | 0.027040077 |
| quite | 0.030676773 |
| pop3 | 0.042199217 |
| various | 0.06080265 |
| prices | 0.9359873 |
| managed | 0.06451222 |
| difficult | 0.071706355 |

https://alexn.org/blog/2012/02/09/howto-build-naive-bayes-classifier.html

# The Value of Pre/Post-Processing

“Clever tricks”:
- title is more important than text
- text in the beginning is more important than at the end
- UNKs handling (spammers are smart)

Pre-processing:
- numbers pre-processing
- take only 15 most “interesting” words

…also: non-NLP features

# NB Model for Lang ID

* The problem of using words
* Character ngrams to the rescue
* Combining them

# Sentiment Analysis

A 3-class whole-text[1] classification problem.

Default approach - Lexicon-based

Possible problems:
- ???

# BoW Models for Sentiment

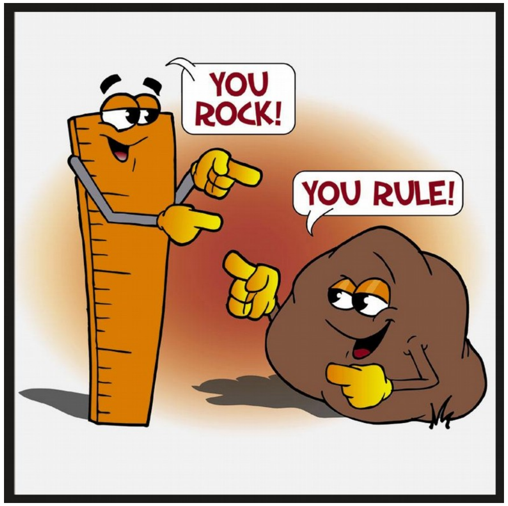
Features: words, bigrams

Models:
* Multinomial NB
* SVM with 2nd-order polynomial kernel
* NBSVM

https://www.aclweb.org/anthology/P12-2018

# BoW Fail Cases

* polysemy
* negation
* neutralization
* multiple sentiments
* multiple objects
* ambiguity
* noise (errors)




# Negation Examples

* Morphological:
  The food was no good.
  I did not like them.
  Their food was without any taste.
  They lack good manners.
* Syntactic:
  If only their prices weren't that high!
  I wish the food they served was more delicious.
  Unlike The X, The Y has great service.
  If they weren't rude, they wouldn't have lost their customers.
  They are unlikely to improve.

# False Negation

High prices were no surprise.

There is no reason to not like them.

It will bring us nowhere, but to success.

There’s no doubt they are going to win the market.

The restaurant was not only cozy, but also located in a wonderful place.

Not only were the waiters rude, but they also brought the wrong dishes.

# Neutralization

* Morphological:

  The X was once described as a leader in sales.

  Earlier, The X used to put off the customers a lot.

* Syntactic:

  If they engage more customers, they will earn more.

  All the hotels, excluding The X Hotel, were sued.

  The restaurant was neither good, nor bad.

# Multiple Sentiments

My sons loved The Playground. They are great, not like The Sandbox with their unsanitary kitchen. High prices were no surprise, though.

# Ambiguity

The company is worth the words that were said earlier.

It tastes like beer.

It’s in the same league with The Happiness Project, trust me.

Obama was right about it.

# More BoW "Tricks"

* normalization of special tokens
* lemmatization/stemming
* stopwords removal
* filtering by "relevance" (e.g. TF-IDF)
* filtering by LM, parse, SRL…
* combining words (negation, prepostions, NER…)

# Stopwords

**thelousylinguist**
@lousylinguist

NLPers, stop removing stop words "just cuz". I repeated a text classification tutorial (analyticsvidhya.com/blog/2018/11/t...) but skipped the 'remove stop words' section and got a 2.8% INCREASE in accuracy. Stop words can improve your outcomes in many cases.

```
[ ] #again try to fit our model to see a big increase in accuracy.
    learn.fit_one_cycle(1, 1e-2)
```

Total time: 00:36

| epoch | train_loss | valid_loss | accuracy |
|---|---|---|---|
| 1 | 0.547412 | 0.396379 | 0.896624 |

```
[37] #WITH STOP WORDS
     #again try to fit our model to see a big increase in accuracy.
     learn.fit_one_cycle(1, 1e-2)
```

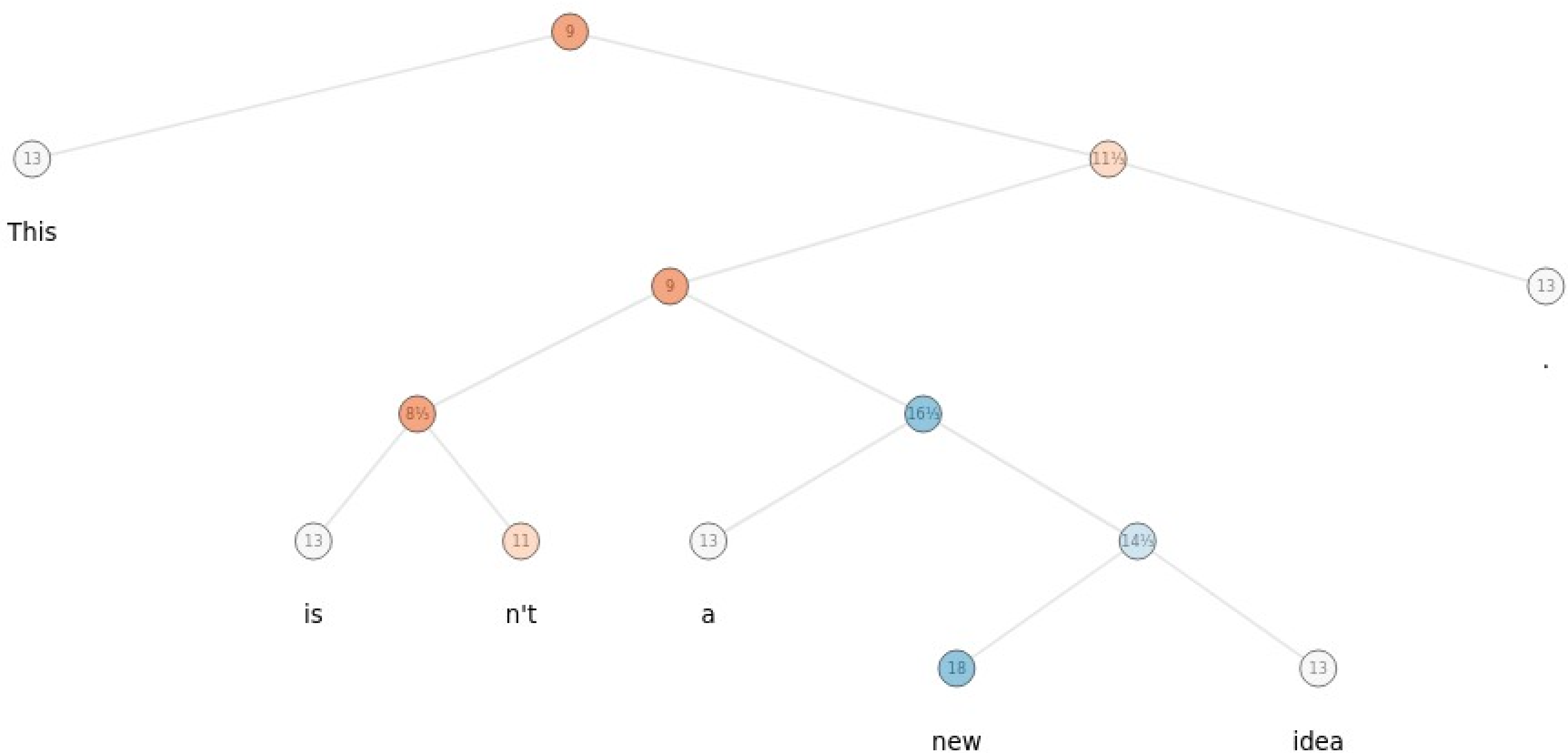
Total time: 00:36

| epoch | train_loss | valid_loss | accuracy |
|---|---|---|---|
| 1 | 0.493484 | 0.285463 | 0.924051 |

1:30 AM · Nov 30, 2018 from Fairfield, CA · Twitter for Android

**112** Retweets **445** Likes

# Sentiment Treebank




https://nlp.stanford.edu/sentiment/treebank.html

# Massive Multi-Class Problems

Problem: classification of user-generated product post according to a 1k labels catalog

Issues:
- class imbalance
- low representation of long-tail classes

Solutions:
- multi-level classification
- 1 model per level with the output classes limited to a particular subset
- AP learning algorithm

# Discriminative Models

* Model conditional probability of label, given a sample:
  can be used only to classify
* Training is direct
* Examples:
  - kNN
  - Perceptron & Averaged Perceptron
  - Logistic Regression (aka MaxEnt)
  - AROW
  - SVM
  - CRF
  - Feed-forward Neural Nets (FNN)

# (Averaged) Perceptron

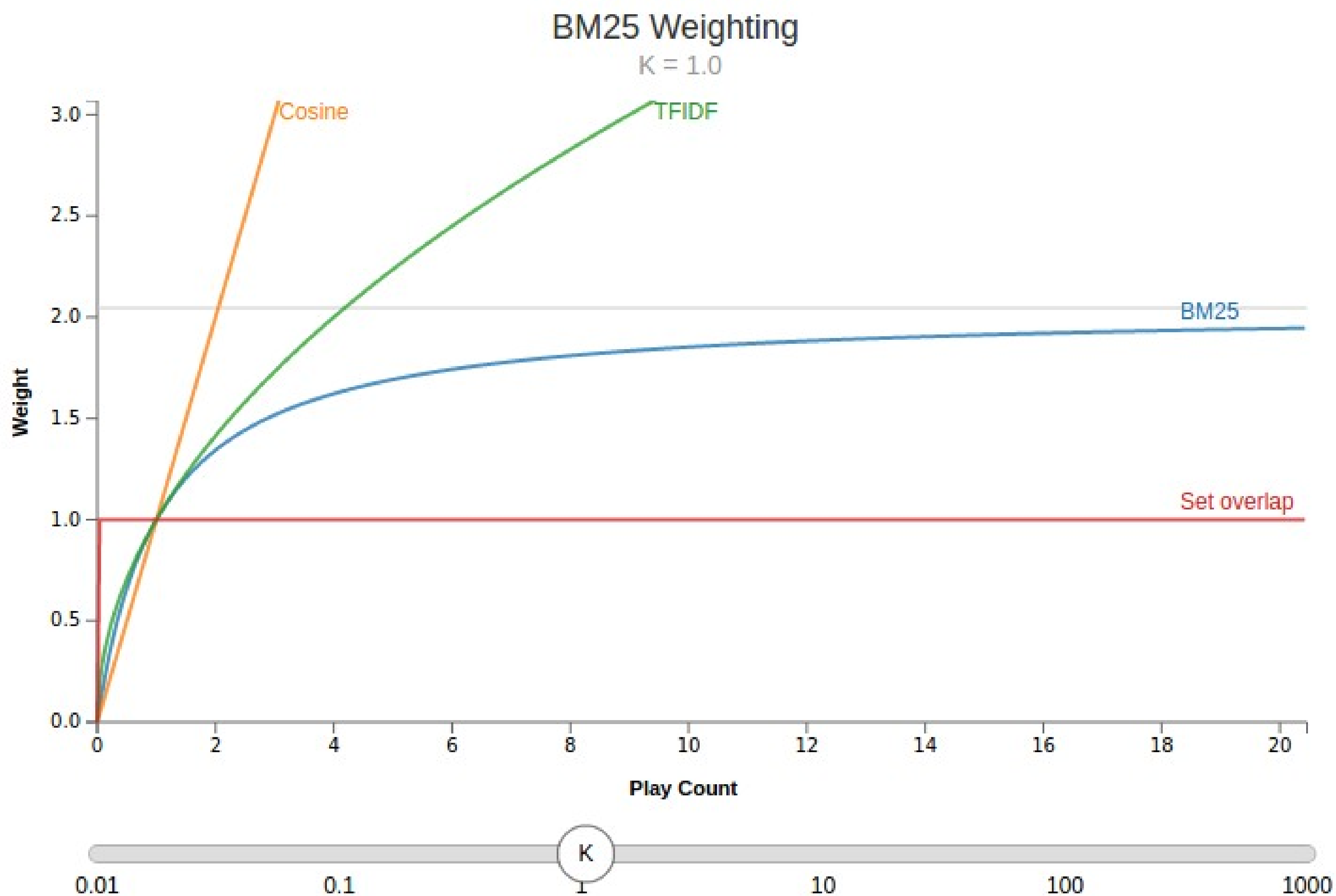
* Simplest linear discriminative model
* On-line learning
* When averaged — ensemble, assimptotic optimality

Perceptron learning rule:

```
def train(self, nr_iter, examples):
    for i in range(nr_iter):
        for features, true_tag in examples:
            guess = self.predict(features)
            if guess != true_tag:
                for f in features:
                    self.weights[f][true_tag] += 1
                    self.weights[f][guess] -= 1
        random.shuffle(examples)
```

https://explosion.ai/blog/part-of-speech-pos-tagger-in-python

# Similarity Metrics




http://www.benfrederickson.com/distance-metrics/

# Jaccard Similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

# Cosine Similarity

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

# BM25 Similarity

```
def bm25_tf_weight(item):
    return item * (K1 + 1.0) / (K1 + item)
```

By changing the value of K1 in this function, we can change the shape to go between the step function used in the Jaccard distance (K1 = 0) and the linear weighting used in the Cosine distance (K1 = +infinity).

The other major change with BM25 is how length normalization is handled. Term counts are scaled by the ratio of the size of the document to the average size. But since sometimes it makes sense to prefer longer documents, BM25 also introduces a parameter B which controls how much influence the length normalization has on the results.

# TF-IDF

A classic IR technic for ranking relevancy

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

Term $x$ within document $y$

$tf_{x,y}$ = frequency of $x$ in $y$

$df_x$ = number of documents containing $x$

$N$ = total number of documents

# TF-IDF

**Variants of term frequency (TF) weight**

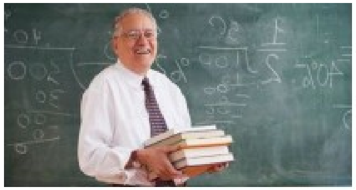
| weighting scheme | TF weight |
|---|---|
| binary | $0, 1$ |
| raw count | $f_{t,d}$ |
| term frequency | $f_{t,d} \Bigg/ \sum_{t' \in d} f_{t',d}$ |
| log normalization | $1 + \log(f_{t,d})$ |
| double normalization 0.5 | $0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$ |
| double normalization K | $K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$ |

# TF-IDF

**Variants of inverse document frequency (IDF) weight**

| weighting scheme | IDF weight ($n_t = \|\{d \in D : t \in d\}\|$) |
|---|---|
| unary | 1 |
| inverse document frequency | $\log \frac{N}{n_t} = -\log \frac{n_t}{N}$ |
| inverse document frequency smooth | $\log \left(1 + \frac{N}{n_t}\right)$ |
| inverse document frequency max | $\log \left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$ |
| probabilistic inverse document frequency | $\log \frac{N - n_t}{n_t}$ |

# Science Pulse



| Keyphrase | Weight | Keyphrase | Weight |
|---|---|---|---|
| massive multiple input output | 7,25 | все буде добре | 1 |
| long short  term memory architecture | 5,12 | коли тебе нема | 1 |
| Live action virtual reality games | 3,15 | небо над дніпром | 1 |
| low rank hankel matrix completion | 3,04 | хочу напитись тобою | 0,78 |
| multi point wireless energy transmission | 3,01 | жити без мети | 0,78 |
| tree  augmented naive bayes classifier | 2,89 | мила моя сьюзі | 0,78 |
| long short term memorized fusion | 2,15 | тінь твого тіла | 0,75 |
| fine grained entity type classification | 1,51 | коли настане день | 0,75 |
| high speed railway communication systems | 1,27 | кожну хвилину життя | 0,75 |
| partially observable markov decision process | 1,13 | коли тобі важко | 0,75 |

https://aiukraine.com/wp-content/uploads/2016/09/Tetiana-Kodliuk.pdf

# RAKE

RAKE short for Rapid Automatic Keyword Extraction algorithm, is a domain independent keyword extraction algorithm which tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurance with other words in the text.

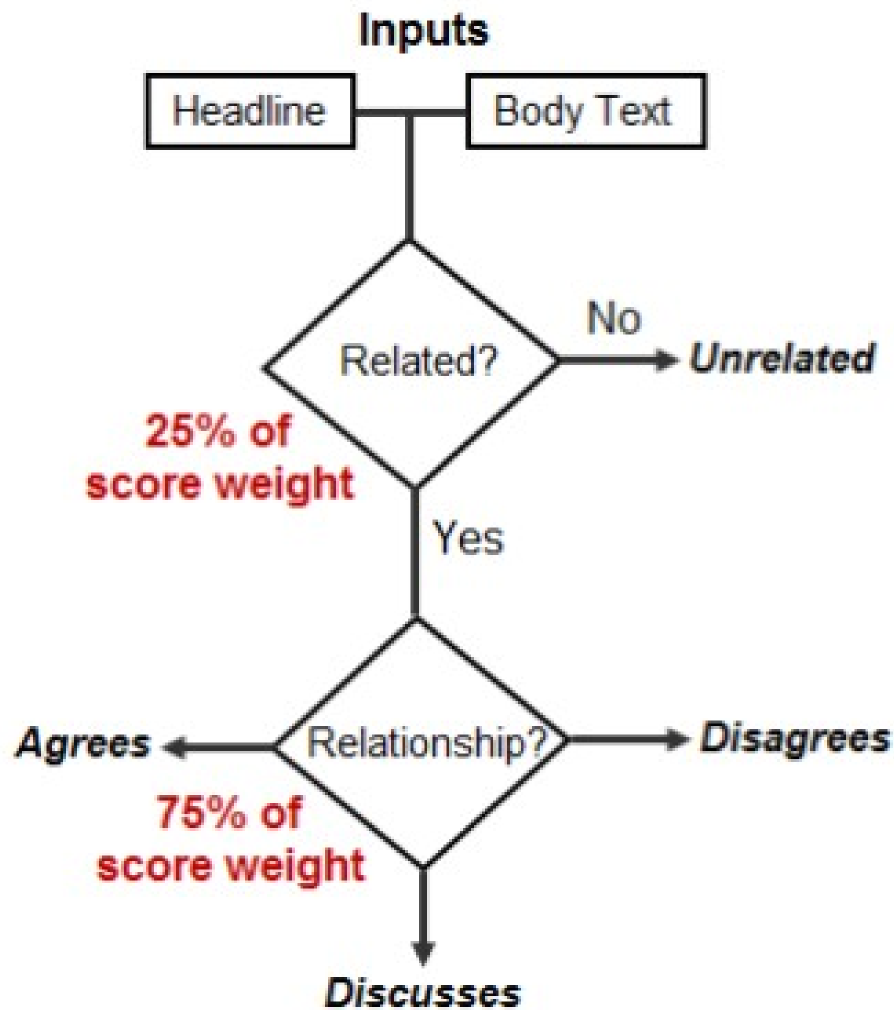
RAKE Pipeline:

* Partition the text by stop words and punctuation
* Creates a co-occurrence matrix of terms.
* For each content word, count deg(word)/freq(word).
* For each key phrase, sum scores of words.
* Return the top 1/3 of key phrases.

https://pypi.org/project/rake-nltk/

# Stance Detection

A 4-class whole text
Hierarchical
classification problem:
* unrelated,
* related:
  - discuss
  - agree
  - disagree



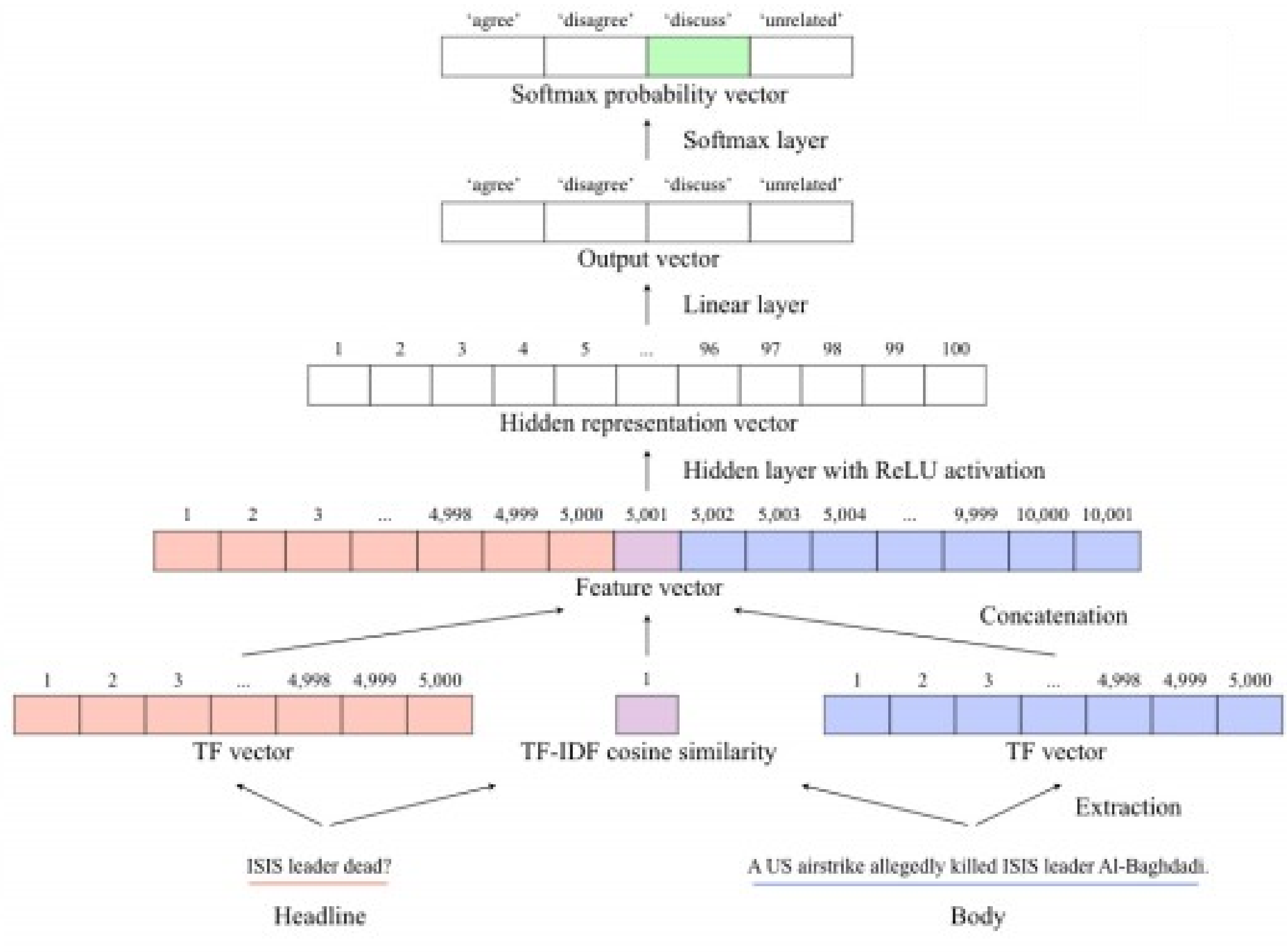

https://github.com/FakeNewsChallenge/fnc-1

# TF-IDF Cosine Similarity Baseline

| | agree | disagree | discuss | unrelated |
|---|---|---|---|---|
| agree | 94 | 14 | 111 | 543 |
| disagree | 13 | 27 | 9 | 113 |
| discuss | 11 | 31 | 607 | 1151 |
| unrelated | 379 | 91 | 650 | 5778 |

Score: 2219.75 out of 4448.5 (49.89884230639541 6%)

# BoW-MLP Model




https://128.84.21.199/pdf/1707.03264.pdf

# BoW Recap

Pros:
+ simple
+ easy to compute
+ flexible
+ doesn’t require lots of data
+ with lots of data works very well

Cons:
- doesn’t capture order
- hard to capture inter-word relations
- hard to scale to real-world vocabularies
- poor generalization

# Read More

Sentiment analysis:

- http://www.datasciencecentral.com/profiles/blogs/test?xg_source=activity
- http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/
- http://ataspinar.com/2016/02/15/sentiment-analysis-with-the-naive-bayes-classifier/
- https://www.cs.uic.edu/~liub/FBS/Sentiment-Analysis-tutorial-AAAI-2011.pdf

BoW in CV:

- http://cs.nyu.edu/~fergus/teaching/vision_2012/9_BoW.pdf

BoW on Extremely Small Datasets:

- https://towardsdatascience.com/text-classification-with-extremely-small-datasets-333d322caee2

Beyond Cosine Similarity

- https://stefansavev.com/blog/beyond-cosine-similarity/