Text preprocessing

Victor Kitov

v.v.kitov@yandex.ru

Text mining

- In text mining feature space is usually high dimensional and sparse.
- To handle sparsity design matrix X may be stored in *sparse matrix format*.
- Linear models work well in high dimensional spaces
  - models are already complex due to many features
  - non-linear models have much more parameters and overfit
- Examples of linear models:
  - regression: linear regression with different regularization
  - classification: logistic regression, SVM
- We may also use arbitrary models in diminished feature space with
  - feature extraction (using, for example, PCA)
  - feature selection (using correlation, mutual information, etc.)

Token set

1. Split documents into individual tokens.
    - tokens may be words or symbol sequences
    - may or may not include punctuation

Token set

1. Split documents into individual tokens.
    - tokens may be words or symbol sequences
    - may or may not include punctuation
2. Form the set of all distinct tokens $\{t_1, t_2, ...\}$.
    - ignore *stop-words* (exact list depends on the application)
    - ignore tokens which are too rare and too frequent
    - account only for particular parts of speech (nouns, adjectives? verbs? ...)

Token set

1. Split documents into individual tokens.
   - tokens may be words or symbol sequences
   - may or may not include punctuation
2. Form the set of all distinct tokens $\{t_1, t_2, ...\}$.
   - ignore *stop-words* (exact list depends on the application)
   - ignore tokens which are too rare and too frequent
   - account only for particular parts of speech (nouns, adjectives? verbs? ...)
3. May add *bigram/trigram collocations*

Token set

1. Split documents into individual tokens.
    - tokens may be words or symbol sequences
    - may or may not include punctuation
2. Form the set of all distinct tokens $\{t_1, t_2, ...\}$.
    - ignore *stop-words* (exact list depends on the application)
    - ignore tokens which are too rare and too frequent
    - account only for particular parts of speech (nouns, adjectives? verbs? ...)
3. May add *bigram/trigram collocations*
4. May normalize words:
    - *stemming*
        - faster
    - *lemmatization*
        - more accurate

Table of contents

Collocations

- Collocations are words that too frequently co-appear in text.
- Examples: New York, fast food, vice president, stock exchange, real estate, deja vu...

Collocations extraction: t-test

- t-test for checking co-occurence of $w_i w_j$:
    - define $x = \mathbb{I}[w_i w_j]$
    - $\overline{x} = \frac{\#[w_i w_j]}{N}$, where N is text length
    - test statistic:

$$\frac{\overline{x} - \mu}{\sqrt{s^2/N}} \to Student(N-1) \to Normal(0,1) \text{ for } N \to \infty$$

    - where $\mu = p(w_i)p(w_j) = \frac{\#[w_i]}{N}\frac{\#[w_j]}{N}$ - expected co-occurence, given independence assumption.
    - $s^2 = \overline{x}(1-\overline{x})$ - sample variance.
    - to be a collocation test statistic should be large.

Collocations extraction: PMI

- Pointwise mutual information:

$$PMI(w_i w_j) = \frac{p(w_i w_j)}{p(w_i)p(w_j)}$$

Collocations extraction: χ^2 Person test

χ^2 Pearson test for independence:

$$
\begin{aligned}
TS &= N\frac{\left[p(w_i w_j)-p(w_i)p(w_j)\right]^2}{p(w_i)p(w_j)}+N\frac{\left[p(w_i\overline{w}_j)-p(w_i)p(\overline{w}_j)\right]^2}{p(w_i)p(\overline{w}_j)}\\
&+ N\frac{\left[p(\overline{w}_i w_j)-p(\overline{w}_i)p(w_j)\right]^2}{p(\overline{w}_i)p(w_j)}+N\frac{\left[p(\overline{w}_i\overline{w}_j)-p(\overline{w}_i)p(\overline{w}_j)\right]^2}{p(\overline{w}_i)p(\overline{w}_j)}
\end{aligned}
$$

$$TS \approx N\frac{\left[p(w_i w_j)-p(w_i)p(w_j)\right]^2}{p(w_i)p(w_j)}$$

$$TS \sim \chi^2(1)$$

Table of contents

Term frequency

- Term-frequency model: $TF(i) = \frac{n_i}{n}$
    - n_i is the number of times t_i appeared in d
    - n total number of tokens in d.
- TF(i) measures how common is token t_i in the document.
- To make $TF(i)$ less skewed it is usually calculated as

$$TF(i) = ln\left(1 + \frac{n_i}{n}\right)$$

Inverted document frequency

- Inverted document frequency: $IDF(i) = \frac{N}{N_i}$
  - N - total number of documents in the collection
  - N_i - number of documents, containing token t_i.
- $IDF(i)$ measures how specific is token i.
- To avoid skewness IDF is more frequently used as

$$IDF(i) = \ln\left(1 + \frac{N}{N_i}\right)$$

Vector representation of documents

- Consider document d and its feature representation x.

- Indicator model: $x^i = \mathbb{I}[t_i \in d]$.
- TF model: $x^i = TF(i)$
- TF-IDF model: $x^i = TF(i) * IDF(i)$

- Several representations, indexed by $I_1, I_2, ... I_K$ can be united into single feature representation.

Different account for different features

- Optimization task with regularization:

$$\sum_{n=1}^{N} \mathcal{L}(\widehat{y}_n, y_n | w) + \lambda R(w) \to \min_{w}$$

- Here λ controls complexity of the model:

Different account for different features

- Optimization task with regularization:

$$\sum_{n=1}^{N} \mathcal{L}(\widehat{y}_n, y_n|w) + \lambda R(w) \to \min_w$$

- Here λ controls complexity of the model:$\uparrow \lambda \Leftrightarrow$ complexity$\downarrow$.

Different account for different features

- Optimization task with regularization:

$$\sum_{n=1}^{N}\mathcal{L}(\widehat{y}_n, y_n|w)+\lambda R(w)\to\min_w$$

- Here λ controls complexity of the model:$\uparrow\lambda\Leftrightarrow$ complexity$\downarrow$.
- Suppose we have K groups of features with indices:

$$I_1, I_2, ...I_K$$

- We may control the impact of each group on the model:

$$\sum_{n=1}^{N}\mathcal{L}(\widehat{y}_n, y_n|w)+\lambda_1 R(\{w_i|i\in I_1\})+...+\lambda_K R(\{w_i|i\in I_K\})\to\min_w$$

- $\lambda_1, \lambda_2, ...\lambda_K$ can be set using cross-validation.
- Scikit-learn allows to set only single λ. But we can control impact of each feature group by different feature scaling.

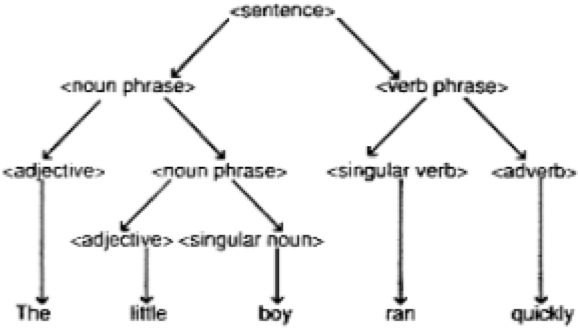
Table of contents

Common problems in NLP

Syntax problems:

- POS-tagging
- Sentence syntax parsing
- Coreference resolution
- Named entity resolution

Sentence syntax parsing



Common problems in NLP

Semantic problems:

- Question answering
- Paraphrase detection
- Chatbots & dialog systems
- Named entity resolution
- Text summarization
- Machine translation
- Topic modeling

Table of contents

Word embeddings

- Distributional hypothesis: similar context leads to similar meaning.
- Form co-occurence matrix $M = \{m_{wc}\}_{w \in W, c \in 2W}$
  - rows: words w
  - columns: counts of words co-occuring with w in the context
    - left context
    - right context
- We can get reduced word representation with reduced SVD:

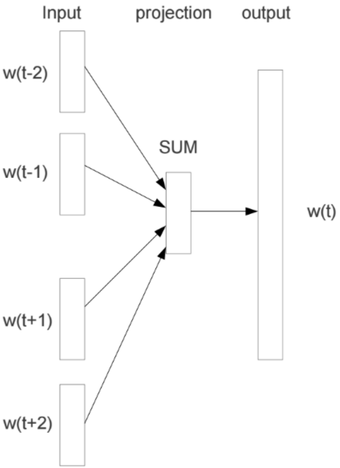
$$M = U_K \Sigma_K V_K^T$$

- Rows of U_K give compact word representations.

Software tools

- gensim.models.word2vec
- Word2vec tool & precomputed representations

Continious bag of words (CBOW)



Continious bag of words (CBOW)

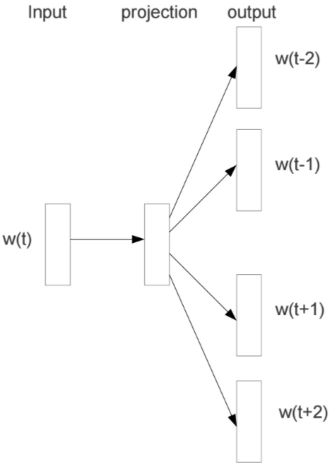
Task: predict current word given context.

$$\frac{1}{T}\sum_{t=1}^{T}\ln p(w_t|w_{t-c},..w_{t-1},w_{t+1},...w_{t+c},\theta)\to\max_{\theta}$$

where $\tilde{v}_{context}=\sum_{t-c\le\tau\le t+c,\,\tau\ne t}\tilde{v}_{w_\tau}$ and

$$p(w_O|w_{t-c},..w_{t-1},w_{t+1},...w_{t+c},\theta)=\frac{\exp(v_{w_0}^T\tilde{v}_{context})}{\sum_{w=1}^{W}\exp(v_{w_O}^T\tilde{v}_{context})}$$

Skip-gram model[1]



[1]Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS, 1–9.

Skip-gram model

- Task: predict context, given current word:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\ln p(w_{t+j}|w_t,\theta)\to\max_{\theta}$$

where W - number of words in the vocabulary and

$$p(w_O|w_I)=\frac{\exp(v_{w_0}^T v_{w_I})}{\sum_{w=1}^{W}\exp(v_{w_O}^T v_I)}$$

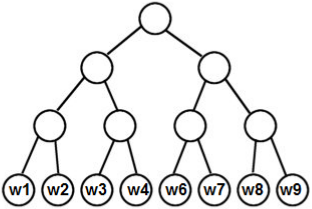
Optimizations

$$p(w_O|w_I) = \frac{\exp(v_{w_0}^T v_{w_I})}{\sum_{w=1}^{W} \exp(v_{w_O}^T v_I)}$$

- Summation over all words in vocabulary is impractical.
- Two optimization approaches:
  - hierarchical soft-max
    - calculates probabilities in $O(\log_2 W)$
  - negative sampling
    - uses different optimization criteria

Hierarchical softmax

- Form a binary tree with words as leaves.
- For each node associate probabilities to follow each child node.
- Probability of a word = probability of path to the word.



Hierarchical softmax

- Consider word w:
  - let $n(w,j)$ be the j-th node on the path
  - let $L(w)$ the length of the path to leaf w:

$$n(w, L(w)) = w$$

  - define $[x] = \begin{cases} +1, & \text{if } x \text{ is satisfied} \\ -1 & \text{if } x \text{ is not satisfied} \end{cases}$

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w, j+1) = \text{left child } n(w,j)]\tilde{v}_{n(w,j)}^T v_{w_I})$$

Hierarchical softmax

- Comments:
  - for balanced tree height is $\log_2 W$
  - each node n is associated internal parameter $\tilde{v}_n$
  - there are $W-1$ internal parameters in total.
  - Huffman tree assigns short codes for frequent words
    - faster

Negative sampling

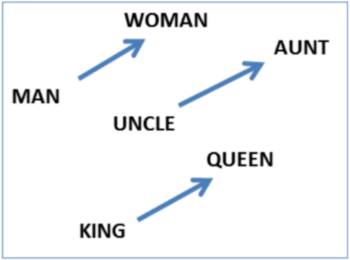
- $P_n(w)$ - noise distribution
- Task: differentiate true (word/context) pairs from noisy ones.
- Formalization:

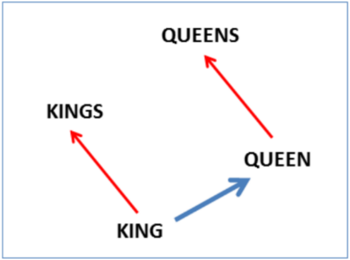
$$\sum_{w_I}\left\{\ln\sigma(\tilde{v}_{W_O}v_{w_I})+\sum_{k=1}^{K}\mathbb{E}_{w_k\sim P_n(w)}\left[\ln\sigma(-\tilde{v}_{w_k}^{T}w_{w_I})\right]\right\}\to\max_{v,\tilde{v}}$$

- $P_n(w)$ - random unigram word sampling

Table of contents

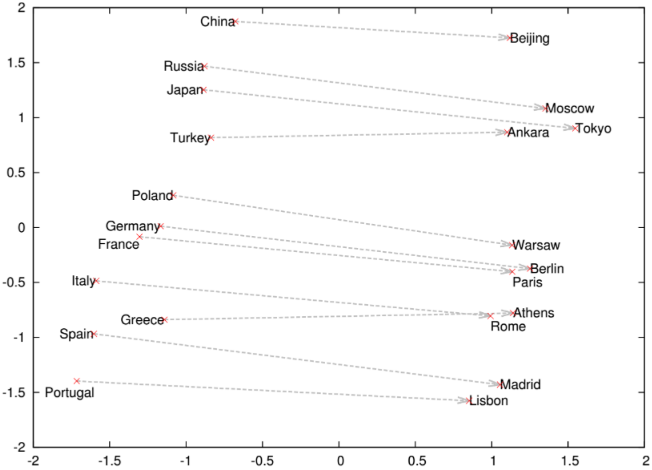
Regularities in vector space[2]



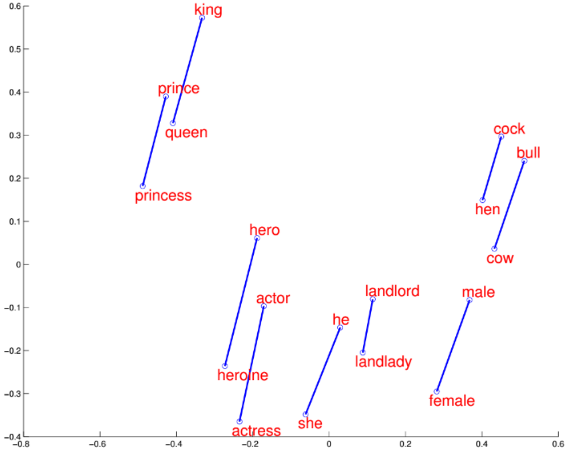


[2]From NIPS presentation of Tomas Mikolov

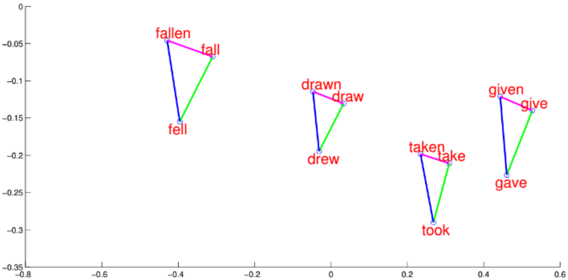
Regularities in vector space



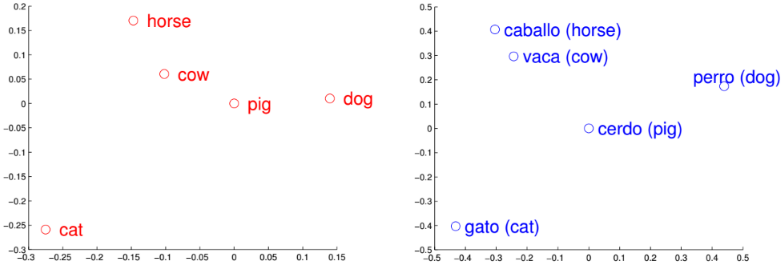
Regularities in vector space



Regularities in vector space



Regularities in vector space[3]



[3] Images were manually rotated and scaled.