Working with text

Victor Kitov

v.v.kitov@yandex.ru

Pipeline

Working with text pipeline:

1. tokenization
2. document encoding by tokens statistics
3. possibly:
    - feature selection
    - feature extraction
4. modelling

Recommended tools

- re - python package for regular expressions
- scipy.sparse - sparse matrices
- scikit-learn - models, document preprocessing, dimensionality reduction
- Vowpal Wabbit - fast linear models
- NLTK-python package for text mining
- lda-topic modeling

Tokenization

1. Split documents into individual tokens.
    - usually tokens are words
        - may be sequences of symbols
    - include punctuation?
        - may lose emotion (e.g. when removing !!!)
    - lowercase all words?
        - decreases dictionary of tokens
        - may lose meaning (US->us) or emotion (GREAT->great)

Tokenization

1. Split documents into individual tokens.
    - usually tokens are words
        - may be sequences of symbols
    - include punctuation?
        - may lose emotion (e.g. when removing !!!)
    - lowercase all words?
        - decreases dictionary of tokens
        - may lose meaning (US->us) or emotion (GREAT->great)
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? ...)

Tokenization

1. Split documents into individual tokens.
   - usually tokens are words
     - may be sequences of symbols
   - include punctuation?
     - may lose emotion (e.g. when removing !!!)
   - lowercase all words?
     - decreases dictionary of tokens
     - may lose meaning (US->us) or emotion (GREAT->great)
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/trigrams (*frequently appearing, *collocations)*

Tokenization

1. Split documents into individual tokens.
    - usually tokens are words
        - may be sequences of symbols
    - include punctuation?
        - may lose emotion (e.g. when removing !!!)
    - lowercase all words?
        - decreases dictionary of tokens
        - may lose meaning (US->us) or emotion (GREAT->great)
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/trigrams (*frequently appearing, *collocations)*
4. May normalize words (stemming, lemmatization)

Word normalization

Normalization words or not?

- decreases dictionary of tokens
- may lose some meaning in word endings

Metohds:

- **stemming** - remove variable endings with list of fixed rules, such as:
  1. ATIONAL->ATE (e.g. relational->relate)
  2. ING->- (e.g. motoring->motor)
  3. SSES->SS (e.g. grasses->grass)
  4. ...
- **lemmatization** - replace wordform with lemma using dictionary
  - is more accurate, needs dictionary
  - e.g.: went->go, fought->fight.
  - even if dictionary is not available may guess lemma by similar words
    - кроказябры->кроказябра because зебры->зебра.

Table of Contents

Standard document representations

- Denote:
  - w_i: i-th token in vocabulary , $i = 1, 2, ...D$.
  - D: total number of unique tokens
- Text has arbitrary length and is not numeric
- Text may be represented by D-dimensional floating vector:
  - indicator model: $x^i = \mathbb{I}[w_i \in document]$
  - TF model: $x^i = TF(i)$
    - TF(i) measures frequency of w_i in the document
  - TF-IDF model: $x^i = TF(i) * IDF(i)$
    - IDF(i) measures specificity of w_i in documents collection
- Several representations, indexed by $I_1, I_2, ...I_K$ can be united into single feature representation.

Term frequency (TF)

- Term-frequency model: $TF(i) = n_i$ or $TF(i) = \frac{n_i}{n}$
  - n_i is the number of times t_i appeared in document
  - n total number of tokens in document
  - second definition gives invariance to document length
- TF(i) measures how common is token t_i in the document.
- To make distribution of $TF(i) = n_i$ less skewed it is usually calculated as $TF(i) = ln(1 + n_i)$

Inverted document frequency (IDF)

- 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

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

Standard representations

- When account for all tokens:
  - number of features is large (D is large)
  - many features are zero (X is sparse)
- To handle sparsity design matrix X may be stored in *sparse matrix format*[1].
- 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 regularizations
  - classification: logistic regression, SVM

[1] In python use scipy.sparse

Dense document representations

- Standard representation with dimensionality reduction with
  - SVD (latent semantic indexing)
  - non-negative matrix factorization
- Topic distribution inside document after topic modelling
  - pLSA, LDA, etc.
- Word2vec - semantically meaningful representation of words
  - using neural network, predicting words by words close by
  - using linearity of word2vec, we can get doc2vec as average over word2vec document word representations

Different account for different features

- In text mining we work with different kinds of features:
  - Text of title, subtitle, body
  - individual words, bigrams, trigrams
  - indicator, TF, TF-IDF representations
- May want to account different groups of features differently.

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

Feature selection for text classification

- Feature selection - select tokens with most discriminative information about document classes.
- We estimate criterion $I(w)$, order words by decreasing $I(w)$ and select features to top K values of $I(w)$.
- Define $p(c|w) = p(y = c|\text{word } w \text{ is present})$ - conditional probability of c-th class of document, given it contains word w.

- When classes are unbalances may replace $p(c|w)$ with $p'(c|w)$:

$$p'(c|w) = \frac{p(y = c|w)/p(y = c)}{\sum_i p(y = i|w)/p(y = i)}$$

All classes informativeness criteria

- Natural measures of discrimination by w:

$$I(w) = std.dev\left(\{p(c|w)\}_{c=1}^{C}\right)$$
$$I(w) = \max\left(\{p(c|w)\}_{c=1}^{C}\right) - \min\left(\{p(c|w)\}_{c=1}^{C}\right)$$

- Gini index for word w:

$$G(w) = \sum_{c=1}^{C} p(c|w)^2$$

- Information gain:

$$\begin{aligned} I(w) &= Entropy(c) - Entropy(c|w) \\ &= -\sum_{c} p(c)\ln p(c) + p(w)\sum_{c} p(c|w)\ln p(c|w) \\ &\quad + (1-p(w))\sum_{c}(1-p(c|w))\ln(1-p(c|w)) \end{aligned}$$

Fixed class informativeness criteria

- Mutual information

$$I_c(w) = \ln\left(\frac{p(w,c)}{p(w)p(c)}\right) = \ln\left(\frac{p(w)p(c|w)}{p(w)p(c)}\right) = \ln\left(\frac{p(c|w)}{p(c)}\right)$$

- χ^2-statistic (test H_0: occurrence of w and occurrence of class c are independent)

$$I_c(w) = \frac{Np(w)^2\left(p(c|w) - p(w)\right)^2}{p(w)\left(1 - p(w)\right)p(c)\left(1 - p(c)\right)}$$

Fixed class informativeness criteria

- Mutual information

$$I_c(w) = \ln\left(\frac{p(w,c)}{p(w)p(c)}\right) = \ln\left(\frac{p(w)p(c|w)}{p(w)p(c)}\right) = \ln\left(\frac{p(c|w)}{p(c)}\right)$$

- χ^2-statistic (test H_0: occurrence of w and occurrence of class c are independent)

$$I_c(w) = \frac{Np(w)^2\left(p(c|w) - p(w)\right)^2}{p(w)\left(1 - p(w)\right)p(c)\left(1 - p(c)\right)}$$

- 2 previous measures estimate word informativeness with respect to fixed class.
- Informativeness of w for all classes can be generated by:

$$I(w) = \sum_c p(c) I_c(w)$$

$$I(w) = \max_c I_c(w)$$

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...
- Algorithm:
    - for each encountered pair of words $w_i w_j$:
        - evaluate collocation score (equal to some test statistic)
        - order word pairs by decreasing score
        - take top ranking pairs as collocations

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: 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: χ^2 Person test

χ^2 Pearson test for independence:

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

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

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