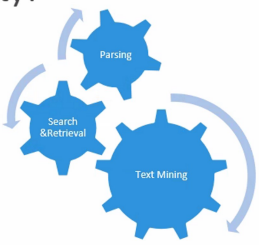
Lec 14

12 декабря 2024 г. 19:21

1. Count encoding - encoding category value with its frequency across training dataset
2. Target encoding - encoding category value with average target variable value given that this category value equal to current. Average(target | x)
3. Now nlp:
4.

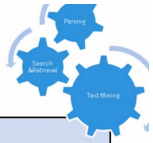
Text Analysis – Problem-solving Tasks

- Parsing
  - Impose a structure on the unstructured/semi-structured text for downstream analysis
- Search/Retrieval
  - Which documents have this word or phrase?
  - Which documents are about this topic or this entity?
- Text-mining
  - "Understand" the content
  - Clustering, classification
- Tasks are not an ordered list
  - Does not represent process
  - Set of tasks used appropriately depending on the problem addressed



1. Here sorting reviews is low level classification and number 4 is more higher level classification.
2.

Buzz Tracking: The Process



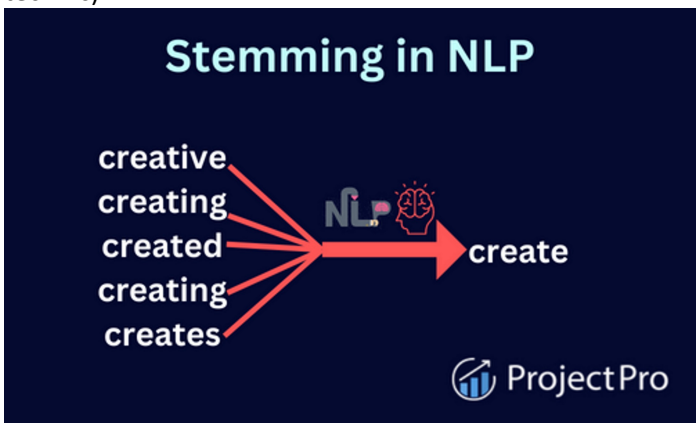
1. Monitor social networks, review sites for mentions of our products.	**Parse** the data feeds to get actual content. Find and filter the raw text for product names (Use **Regular Expression**).
2. Collect the reviews.	Extract the relevant raw text. Convert the raw text into a suitable **document representation**. **Index** into our review **corpus**.
3. Sort the reviews by product.	**Classification** (or **"Topic Tagging"**)
4. Are they good reviews or bad reviews? We can keep a simple count here, for trend analysis.	**Classification** (sentiment analysis)
5. Marketing calls up and reads selected reviews in full, for greater insight.	**Search/Information Retrieval**.

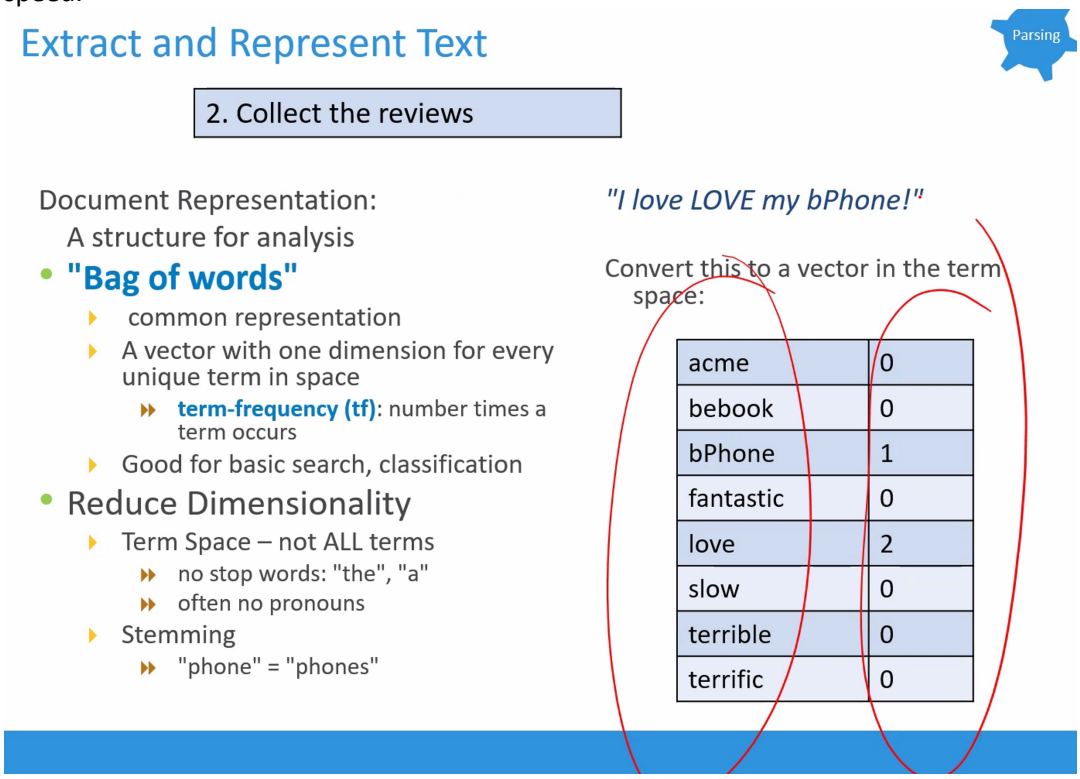
Parsing:

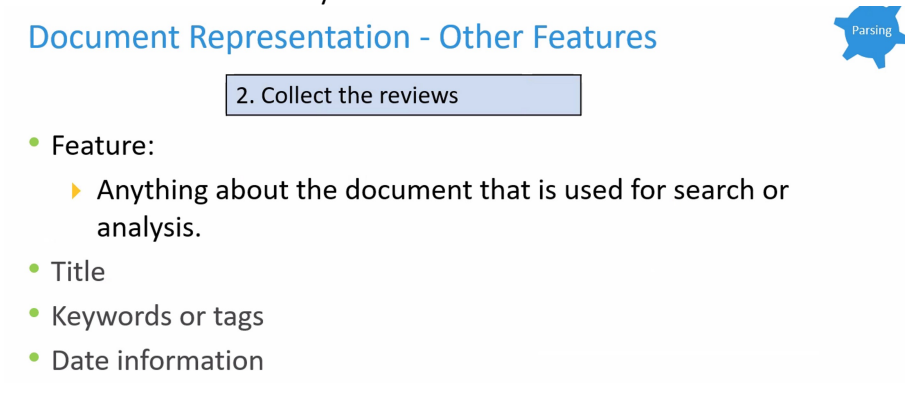
1. Consider structure and semi-structure and just REGEXP accuratly

Search:

1. Forming space of features - bag of words

2. Bag of words - document representation in word space of bag of words. Each document is basically a vector in this Bag of words space. The vector value is usually described by term frequency, but can be binary as well.
3. Natural languages are quite complex, so usually only roots of words are used(stemming technic).
4. 

5. Moreover stop-words such as articles, pronouns usually skipped, cause they don't make much sense.
6. This is made to reduce number of dimensions in bag of words and hence increase search speed.
7. 

1. Extra features such as location of publication/review or device where it was publisjed from or date of review or number of stars and many more features can help us with search and text mining further
2. Stars in reviews are literally tons of labeled data to learn on
7. 

Document Representation - Other Features

2. Collect the reviews

- Feature:
  - Anything about the document that is used for search or analysis.
- Title
- Keywords or tags
- Date information
- Source information
- Named entities

7.

8. Reverse index: which documents contain this feature(tag, number of stars, words and etc.)?
9. Ofcourse we can go through vectors but… too slow
10. Reverse index is basically a bag of documents. A vector for every feature with information of where can we meet this term.
11. The big problem is to update data so it is actual

Classification:

1. One of the problem is more complex formulations or not standard or slangy form of names of products

Text Classification (I) - "Topic Tagging"

3. Sort the Reviews by Product

Not as straightforward as it seems

2.

"The bPhone-5X has coverage everywhere. It's much less flaky than my old bPhone-4G."

"While I love Acme's bPhone series, I've been quite disappointed by the bEbook. The text is illegible, and it makes even my old Newton look blazingly fast."

3. Big problem is labeling and tagging reviews.
4. Naïve bayes can be first attempt

Search and information retrieval:

1. TF - term frequency - basically value of frequency in vector of document. But the problem is that terms can have different sense value. For example, in phone reviews the term "phone" will be almost in every document, but it's not very useful since we already know that our set of documents is about phones.
2. DF - document frequency - number of documents where this term occurs.
3. IDF - inverse document frequency. The more df of term the less its idf value the less important it is. The measure of importance of certain term. Across corpus metric. Not connected to particular document.

4.

Inverse Document Frequency (idf)

Search &Retrieval

5. Marketing calls up and reads selected reviews in full, for greater insight.

$$idf(t) = log\ [N/df(t)]$$

- *N*: Number of documents in the corpus
- *df*(t): Number of documents in the corpus that contain a term *t*

- Measures term uniqueness in corpus
  - "phone" vs. "brick"
- Indicates the importance of the term
  - Search (relevance)
  - Classification (discriminatory power)

5. TF-IDF - measure of relevance. Relevance consists of tf and idf. So it accounts term frequency in document as well as importance of this term. And to calculate relevance of some document for some request we have to sum up all tf-idfs for every term in request for this document. And then pick the document with maximum relevance.

6.

TF-IDF and Modified Retrieval Algorithm

Search &Retrieval

5. Marketing calls up and reads selected reviews in full, for greater insight.

- Term frequency – inverse document frequency (tf-idf or tfidf) of term t in document d:

$$tfidf(t, d) = tf\ (t, d) * idf(t)$$

query: *brick, phone*

- Document with "brick" a few times more relevant than document with "phone" many times
- Measure of Relevance with tf-idf
- Call up all the documents that have any of the terms from the query, and sum up the tf-idf of each term:

$$\text{Relevance(d)} = \sum_{i \in [1,n]} tfidf\,(t_i, d)$$

7. How to calc relevance another way? Quite interesting metric PageRank - calculating links on this document. The more links on it the more power it has.
PageRank:

In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links **L( )**.

$$PR(A) = \frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}.$$

In the general case, the PageRank value for any page **u** can be expressed as:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)},$$

Initially all N pages have rank 1/N. Then for each page rank is recalculated using formula above, where PR(v) is current rank of page v and L(v) is number of outgoing links from page v.

This recalculation repeats until convergence - when ranks don't change anymore or change very little(if there was a loop somewhere).

8.