ITCS 4111/5111: Introduction to NLP

Text Classification using Naïve Bayes

Razvan C. Bunescu

Department of Computer Science @ CCI

rbunescu@uncc.edu

Text Classification: Sentiment Analysis

Movie reviews:

Positive: This was a great movie, which I thoroughly enjoyed.

Negative: I was very disappointed in this movie, it was a waste of time.

- Lexical features, e.g. presence of words such as *great* or *disappointed,* can be used to determine the sentiment orientation.
  - Can you think of examples where the same word may be used for both types of sentiment? How would you fix that?
- Represent each review as a *bag-of-words* feature vector:
  - High dimensional, sparse feature vector => use sparse representations that map features to indeces.
  - Feature value is 1 if word is present, 0 otherwise:
    - Can use more sophisticated word weighting schemes from IR, such as tf, tf.idf.
    - Can use stems instead of tokens.

Why sentiment analysis?

- **Movies**:
  - Is this review positive or negative?
    - Predict box office performance from sentiment in initial reviews, …
- **Products**:
  - What do people think about the new iPhone?
    - Predict market share, value of manufacturer company stock, …
- **Public sentiment**:
  - How is consumer confidence?
    - Predict debt, mortgage lending, credit card use, …
- **Politics**:
  - What do people think about this candidate or issue?
    - Predict election outcomes, ballot vote outcomes, …

Scherer Typology of Affective States

- **Emotion**: brief organically synchronized … evaluation of a major event
  - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood**: diffuse non-caused low-intensity long-duration change in subjective feeling
  - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances**: affective stance toward another person in a specific interaction
  - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes**: enduring, affectively colored beliefs, dispositions towards objects or persons
  - *liking, loving, hating, valuing, desiring*
- **Personality traits**: stable personality dispositions and typical behavior tendencies
  - *nervous, anxious, reckless, morose, hostile, jealous*

Sentiment Analysis: Is the **attitude** of the text *positive* or *negative*?

- **Emotion**: brief organically synchronized … evaluation of a major event
  - *angry, sad, joyful, fearful, ashamed, proud, elated*
- **Mood**: diffuse non-caused low-intensity long-duration change in subjective feeling
  - *cheerful, gloomy, irritable, listless, depressed, buoyant*
- **Interpersonal stances**: affective stance toward another person in a specific interaction
  - *friendly, flirtatious, distant, cold, warm, supportive, contemptuous*
- **Attitudes**: enduring, affectively colored beliefs, dispositions towards objects or persons
  - *liking, loving, hating, valuing, desiring*
- **Personality traits**: stable personality dispositions and typical behavior tendencies
  - *nervous, anxious, reckless, morose, hostile, jealous*

Text Classification

- Sentiment analysis.
- Spam detection.
- Authorship identification.
- Language identification.
- Assigning subject categories, topics, or genres.

Text classification: Spam detection

From: Tammy Jordan
jordant@oak.cats.ohiou.edu
Subject: Spring 2015 Course
--

CS690: Machine Learning

Instructor: Razvan Bunescu
Email: bunescu@ohio.edu
Time and Location: Tue, Thu 9:00 AM , ARC 101
Website: http://ace.cs.ohio.edu/~razvan/courses/ml6830

Course description:
Machine Learning is concerned with the design and analysis of algorithms that enable computers to automatically find patterns in the data. This introductory course will give an overview …

From: UK National Lottery
edreyes@uknational.co.uk
Subject: Award Winning Notice
--

UK NATIONAL LOTTERY. GOVERNMENT ACCREDITED LICENSED LOTTERY. REGISTERED UNDER THE UNITED KINGDOM DATA PROTECTION ACT;

We happily announce to you the draws of (UK NATIONAL LOTTERY PROMOTION) International programs held in London , England Your email address attached to ticket number :3456 with serial number :7576/06 drew the lucky number 4-2-274, which subsequently won you the lottery in the first category …

- Email filtering:
  - Provide emails labeled as *{Spam, Ham}*.
  - Train *Naïve Bayes* model to discriminate between the two.
    - [Sahami, Dumais & Heckerman, AAAI'98]

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

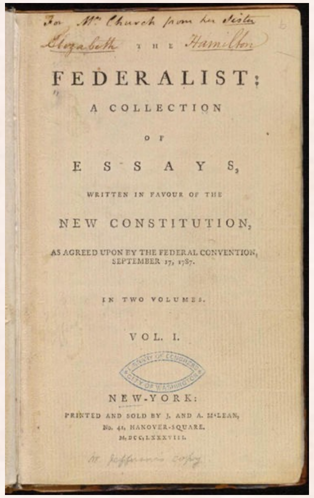
http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

- Adversarial setting:
  - Text encapsulated in images.
  - Misspelled words, …

Text classification: Authorship identification

- Who wrote which Federalist papers?
  - 1787-8: anonymous essays try to convince New York to ratify U.S Constitution: Jay, Madison, Hamilton.
    - Authors tried to conceal their identities. => authorship of 12 of the letters in dispute.
    - 1963: solved by Mosteller and Wallace using Bayesian methods.



- Who is the author of Shakespeare’s plays?
  - *Disclaimer*: It is widely accepted that Shakespeare is the author of Shakespeare.
  - *Theory*: Francis Bacon wrote the plays.
    - Bacon’s official rise might have been impacted if he were known as the authors of plays for the public stage.
      - The plays were credited to Shakespeare, a front for shielding Bacon.

Text classification: Mapping Medline article to MeSH categories


Available online at www.sciencedirect.com

Brain and Cognition

Syntactic frame and verb bias in aphasia: Plausibility judgments of undergoer-subject sentences

Susanne Gahl, Lise Menn, Gail Ramsberger, Daniel S. Jurafsky, Elizabeth Elder, Molly Rewega, and L. Holland Audrey

- **MeSH Subject Category Hierarchy:**
  - Antagonists and Inhibitors
  - Blood Supply
  - Chemistry
  - Drug Therapy
  - Embryology
  - Epidemiology
  - …

Text Classification: Definition

- *Input*:
  - a document $d \in \mathrm{D}$
  - a fixed set of classes $\mathrm{C} = \{C_1, C_2, \ldots, C_K\}$

- *Output*:
  - a predicted class $C_k \in \mathrm{C}$.

Rule-based vs. Machine learning

- Hand-coded Rules based on combinations of words or other features:
  - **Spam filtering**: black-list-address OR ("dollars" AND "you have been selected").
  - **Sentiment analysis**: use *affective lexicons*.
    - Accuracy can be high:
      - If rules carefully refined by expert.
      - But building and maintaining these rules is expensive.

- Supervised learning:
  - **Input**:
    - A fixed set of classes C = $\{C_1, C_2, \ldots, C_K\}$
    - A training set of N hand-labeled documents $(d_1, t_1), (d_2, t_2), \ldots, (d_N, t_N)$, where $t_n \in$ C
  - **Output**:
    - A learned classifier h: D → C

Classification Algorithms

- Train a classification algorithm on the labeled feature vectors, i.e. training examples.
  - Use trained model to determine the sentiment orientation of new, unseen reviews.

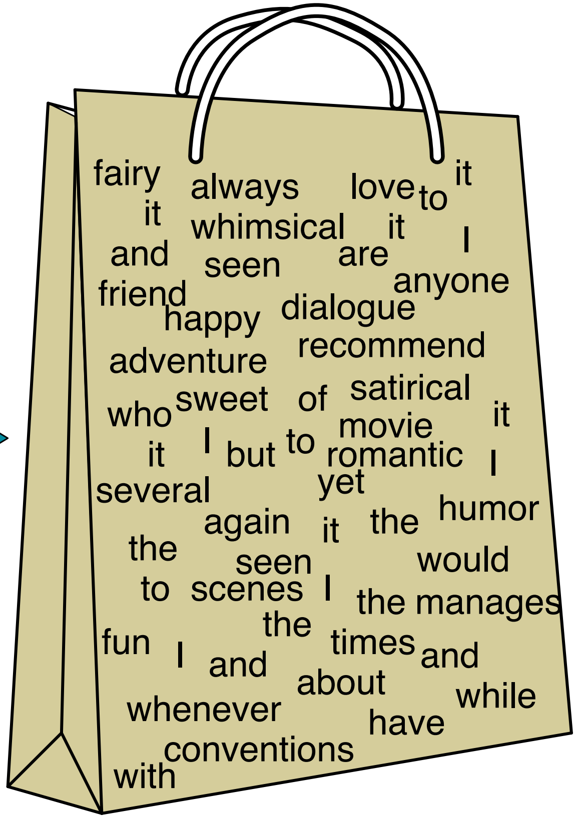
- Machine learning models:
  - **Naïve Bayes**
  - **Logistic Regression**
  - Perceptron
  - Support Vector Machines
  - **Neural networks**
  - k-Nearest Neighbors
  - …

Naïve Bayes model

- Simple ("naive") classification method based on:
  - Bayes rule.
  - Simple class-conditional independence between words.
- Relies on very simple representation of document:
  - **Bag of Words (BoW).**

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

The bag of words representation

$h($

seen	2
sweet	1
whimsical	1
recommend	1
happy	1
...	...

$) = C_k$

Sentiment Analysis

Movie reviews:

Positive: This was a great movie, which I thoroughly enjoyed.

Positive: Despite the bad reviews I read online, I liked this move.

Negative: The movie was not as good as I expected.

- It appears that the bag-of-words approach is not sufficient.
- Can try to address negation:
  - Use bigram NOT_X for all words X following the negation [Pang et al. EMNLP'02].
- Model sentiment compositionality:
  - Train recursive deep models over sentiment treebanks [Socher et al., EMNLP'13]
- Apply more sophisticated classifiers:
  - Convolutional Neural Networks (CNNs) [Kim, 2014]

Sentiment Analysis

More examples showing the limitations of *bag-of-words* models [Eisenstein, 2019]:

a. That's not bad for the first day.

b. This is not the worst thing that can happen.

c. It would be nice if you acted like you understood.

d. There is no reason at all to believe that the polluters are suddenly going to become reasonable. (Wilson et al., 2005)

e. This film should be brilliant. The actors are first grade. Stallone plays a happy, wonderful man. His sweet wife is beautiful and adores him. He has a fascinating gift for living life fully. It sounds like a great plot, **however**, the film is a failure. (Pang et al., 2002)

Bayes' Rule Applied to Documents and Classes

- For a document d and a class c:

Likelihood

Posterior

Prior

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

$$P(d) = P(d, c_1) + P(d, c_2)$$

$$P(d) = P(d|c_1)P(c_1) + P(d|c_2)P(c_2)$$

$$P(d) = \sum_{c \in C} P(d|c)P(c)$$

- Inference $\equiv$ find c_{MAP} to minimize misclassification rate:

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} = \underset{c \in C}{\operatorname{argmax}}\, P(d|c)P(c)$$

Maximum a Posteriori

What if we want to compute this too?

Naive Bayes Classifier

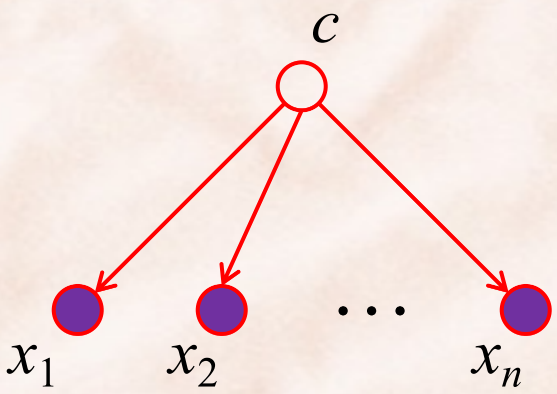
- Inference (at test time): find *maximum a posteriori* (MAP) class:

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, \overset{\textit{Likelihood}}{\overbrace{P(d \mid c)}}\, \overset{\textit{Prior}}{\overbrace{P(c)}} = \underset{c \in C}{\operatorname{argmax}}\, P(\underbrace{x_1, x_2, \ldots, x_n}_{\text{document } d \text{ represented as features } x_1,\, x_2,\, \ldots,\, x_n} \mid c) P(c)$$

- If each feature $x_j \in X$ and class $c \in C$, then $|X|^n \times |C|$ params $P(x_1, x_2, \ldots, x_n \mid c)$:
  - x_j could be word at position j in document d, X could be entire vocabulary.
    - Number of params is **polynomial** in the size of vocabulary!
      - Could only be estimated if a very, very large number of training examples was available.
        - » Unfeasible in practice.

The Naïve Bayes Model

- **NB assumption**: features are conditionally independent given the target class.



$$P(d|c) = P(x_1,\ldots,x_n|c) = P(x_1|c)\,P(x_2|c)\ldots P(x_n|c)$$

$$\xcancel{P(x_1,\ldots,x_n) = P(x_1)\,P(x_2)\ldots P(x_n)}$$

- **BoW assumption**: assume position doesn't matter.

$$P(d|c) = P(x_1|c)\,P(x_2|c)\ldots P(x_n|c) = \prod_{x \in d} P(x|c)$$

- Assuming binary features, i.e. word *w* appears (or not) at position *j*:

  $\Rightarrow$ need to estimate only $|X|\times|C|$ parameters, a lot less than $|X|^n\times|C|$

Multinomial Naive Bayes Classifier

- MAP inference at test time, using Naïve Bayes model:

$$c_{MAP} = \underset{c \in C}{\operatorname{argmax}}\, P(x_1, x_2, \ldots, x_n \mid c)P(c)$$

$$c_{MAP} = c_{NB} = \underset{c \in C}{\operatorname{argmax}}\, P(c) \prod_{x \in d} P(x|c)$$

- Use probabilities over all word *positions* in the document *d*:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}}\, P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Multiplying lots of probabilities can result in floating-point underflow!

Naïve Bayes: Use log-space to avoid underflow

- Instead of this:

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

$$\log(ab) = \log(a) + \log(b)$$

- Work in log-space:

$$c_{\text{NB}} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i|c_j) \right]$$

- This is ok since *log* doesn't change the ranking of the classes:
  - class with highest prob still has highest log prob.
- Model is now just max of sum of weights:
  - A **linear** function of the parameters. So Naïve Bayes is a **linear classifier**.

Learning the Multinomial Naive Bayes Model

- **Maximum Likelihood** estimates:
  - Use the frequencies of features in the data.

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}} \qquad \hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

- Create mega-document for topic *j* by concatenating all docs in this topic:
  - Use frequency of *w* in mega-document.

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears
among all words in documents of topic c_j

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up)***?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace Smoothing for Naïve Bayes

- **Laplace (add-1) smoothing**:
  - |V| "hallucinated" examples spread evenly over all |V| values of w_i, for each class c.

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c)}{\sum_{w \in V} \left(count(w, c)\right)}$$

$$change\ to \ = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Unknown words and Stop words

- **Unknown words** appear in the test data, but not in our training data or vocab.
  - Ignore unknown words.
    - Remove them from the test document, i.e. pretend they weren't there.
      - Don't include any probability for them at all.
  - Why don't we build an unknown word model?
    - It doesn't help; knowing which class has more unknown words is not generally useful to know.

- **Stop words:** very frequent words like *the* and *a*:
  - Sort the vocabulary by frequency in the training, call the top 10 or 50 words the **stopword list**.
  - Remove all stop words from the training and test sets, as if they were never there.
    - But in most text classification applications, removing stop words don't help, so it's more common to **not** use stopword lists and use all the words in naive Bayes.

Text Categorization with Naïve Bayes

- Generative model of documents:
  1) Generate document category by sampling from $p(c_j)$.
  2) Generate a document as a bag of words by repeatedly sampling with replacement from a vocabulary $V = \{w_1, w_2, \ldots, w_{|V|}\}$ based on $p(w_i \mid c_j)$.

When do we stop generating words? Provide two solutions …

- Inference with Naïve Bayes:
  - Input :
    - Document d with n words $x_1, x_2, \ldots x_n$.
  - Output:
    - Category $c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$

$$c_{\mathrm{NB}} = \operatorname*{argmax}_{c_j \in C} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$$

Text Categorization with Naïve Bayes

- Training with Naïve Bayes:
  - Input:
    - Dataset of training documents D with vocabulary V.
  - Output:
    - Parameters $p(C_k)$ and $p(w_i \mid C_k)$.

---

for each category C_k:
    let D_k be the subset of documents in category C_k
    set $p(C_k) = |D_k| / |D|$
    let n_k be the total number of words in D_k
    for each word $w_i \in \mathrm{V}$:
        let n_{ki} be the number of occurrences of w_i in D_k
        set $p(w_i \mid C_k) = (n_{ki}+1) / (n_k + |V|)$

A worked sentiment example

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable ~~with~~ no fun

Prior from training:

P(−) = 3/5
P(+) = 2/5

Drop **unknown words**, i.e. "with”.

Likelihoods from training:

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

Scoring the test document:

$$\boxed{P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Sentiment analysis with Binary Naïve Bayes

- For tasks like sentiment, word **occurrence** is more important than word **frequency**.
  - The occurrence of the word *fantastic* tells us a lot.
  - The fact that it occurs 5 times may not tell us much more.

- **Binary multinominal Naive Bayes**, or **binary NB**
  - Clip word counts at 1.
  - This is different than Bernoulli Naïve Bayes.

- Inference with Naïve Bayes:
  - Input is document d with n words $x_1, x_2, \ldots x_n$.
    - **Remove duplicates from *d*, keep only 1 instance of each word type.**
  - Output:
    - Category $c_{MAP} = \text{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$

Text Categorization with Binary Naïve Bayes

- Training with Naïve Bayes:
  - Input:
    - Dataset of training documents D with vocabulary V.
  - Output:
    - Parameters $p(C_k)$ and $p(w_i \mid C_k)$.

---

1. **for each** category C_k:
2. **let** D_k be the subset of documents in category C_k
   for each document $d \in D_k$, remove duplicates from d.
3. **set** $p(C_k) = |D_k| / |D|$
4. **let** n_k be the total number of words in D_k
5. **for** each word $w_i \in$ V:
6. **let** n_{ki} be the number of occurrences of w_i in D_k
7. **set** $p(w_i \mid C_k) = (n_{ki}+1) / (n_k + |V|)$

Binary multinominal naive Bayes

Four original documents:

- – it was pathetic the worst part was the boxing scenes
- – no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- – it was pathetic the worst part boxing scenes
- – no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

Counts can still be 2, binarization is within doc.

	NB Counts +	NB Counts –	Binary Counts +	Binary Counts –
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Naïve Bayes

- Often has good performance, despite strong independence assumptions:
  - Quite competitive with other classification methods on UCI datasets.

- It does not produce accurate probability estimates when independence assumptions are violated:
  - The estimates are still useful for finding max-probability class.

- Does not focus on completely fitting the data $\Rightarrow$ resilient to noise.

- NB model is sum of weights = a **linear** function of the inputs.

$$c_{\mathrm{NB}} = \operatorname*{argmax}_{c_j \in C} \left[\log P(c_j) + \sum_{i \in \mathrm{positions}} \log P(x_i | c_j) \right]$$

Homework Assignment

- We will be using the MultinomialNB class from sklearn.naive_bayes
  - The training function fit(X, y) takes as input the data matrix X and the list of labels y.
    - The rows of X are training examples represented as *feature vectors*.
      - The number M of columns of X represents the number of features of each example.
    - Each feature vector X[*j*] contains M features, indexed from 0 to M − 1.
      - Thus, X[*j*,*m*] represents the value of feature with index *m* for example number *j*.
- But Naïve Bayes for document classification uses words as features:
  - How do we get from words to numerical feature indexes?
    - word_features(tokens) takes as input a list of tokens and returns a dictionary, where each token is transformed into a feature name by prepending the prefix 'WORD_' and is mapped to its frequency in tokens.
      - if 'like' appears 5 times in tokens, then 'WORD_like' is mapped to 5.

Homework Assignment

– How do we get from words to numerical feature indexes?
  - create_vocab(examples) takes a list of examples, where each example is a dictionary mapping feature names to their counts in the example, e.g. {'WORD_like': 5}, and returns a dictionary feature_vocab where all unique features that appears in examples are mapped to a unique index, e.q. feature_vocab['WORD_like'] = 237.
    – The size M of feature_vocab will be equal to the number of unique features (e.g. unique words) in the set of examples.
    – Thus, each example has a number of features, where each feature has an index in [0, M).
  - features_to_ids(examples, feature_vocab) takes the feature dictionary created above and transforms each example to be a vector of feature values, e.g. example[m] = c means the feature with index *m* appears *c* times in the example.
    – However, most features (e.g. words) do not appear in an example (e.g. document), which means the feature vectors will be very sparse (most entries are 0).
    – To not waste memory for features values that are 0, we use the class spacy. sparse.lil_matrix to store feature vectors in a memory-efficient, *sparse format*.

Recommended Readings

- Section 4, from 4.1 to 4.6, and 4.10, in J & M.
- Section 2, from 2.1 to 2.2, in Eisenstein.
- Nathaniel E. Helwig's Introduction to Probability Theory.