SEMANTIC COMPUTING

Lecture 2: Language Models and Text Classification

Dagmar Gromann
International Center For Computational Logic

TU Dresden, 30 April 2018
Overview

- Language modeling
- Text classification and supervised learning
- Naïve Bayes
Language Modeling
Motivation of Language Modeling

In contrast to formal languages, natural languages:

- emerge naturally
- are not fully specified
- change over time (language evolution)
- change their word usage (e.g. awful used to mean “worth of awe”)

One approach is trying to specify rules of a language using formal grammars and structures

**Alternative**: Specify a language model that is learned from examples.
Motivating example

Toy corpus

You threaded the **eye of the needle**.
...that gave him success, which came in 1978 with the best-selling *Eye of the Needle*.
...honesty is in the **eye of the** beholder, right, so.
...headed to intercept the **eye of the** hurricane.
But when we pass through the **eye of the needle**, we leave our words behind...
thread all of the disparate stories through the eye of one **needle**.
Sickey gave pine **needle** baskets...

\[ P(\text{needle}|\text{eye of the}) = \frac{\text{count}(\text{eye of the needle})}{\text{count}(\text{eye of the})} = ? \]
Probabilistic Language Model

Language Model
Models that assign probabilities to sequences of words are called language models or LMs.

Two interrelated tasks:
1. Assign a probability to a sequence of words in a language
2. Assign a probability of the likelihood that a given word will follow a sequence of words (predict the probability of a word $w_T$ given its $n - 1$ previous words)

A language model computes
1: $P(W) = P(w_1, w_2, w_3, ..., w_n)$ or
2: $P(w_4|w_1, w_2, w_3)$
Application Examples

Language modeling is a crucial component of many real-world applications:

- machine translation
  \[ P(\text{I didn't do anything}) > P(\text{I didn't do nothing}) \]

- speech recognition
  \[ P(\text{I ramble}) > P(\text{I Rambo}) \] also hard for humans:
  \[ P(\text{I've got the power}) > P(\text{Agathe Bauer}) \]

- spelling correction
  \[ P(\text{Please pay before exiting}) > P(\text{Please pai before existing}) \]

- and many more
Probabilities

Maximum Likelihood estimation
Given a word we can estimate a probability distribution as
\[ P(w) = \frac{\text{count}(w)}{\sum_w \text{count}(w')} \]
If we randomly choose a word of a corpus, how likely is it going to be word \( w \)?

Joint Probabilities
The joint probability of two words that occur in sequence with the distribution \( p(w_1, w_2) \) can be estimated similar to the probability distribution over a single variable:
\[ p(w_1, w_2) = \frac{\text{count}(w_1, w_2)}{\sum_{w_1', w_2'} \text{count}(w_1', w_2')} \]
Conditional Probabilities

Probability of a word \( w_2 \) given that a previous word \( w_1 \) co-occurs with it is written as \( p(w_1|w_2) \) and defined as

\[
p(w_2|w_1) = \frac{p(w_1, w_2)}{p(w_1)}
\]

If the two words were completely independent, then

\[
p(w_2|w_1) = p(w_2)
\]
Chain Rule

We can describe probability distributions in terms of conditional probability and the chain rule allows us to calculate any member of a joint distribution of random variables using conditional probabilities.

In other words, we use the definition of the conditional probability to factor the joint probability:

\[
P(w_1, w_2, w_3, \ldots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_n|w_1, \ldots, w_{n-1})
\]

Example

\[
P(\text{“eye of the needle”}) = P(\text{eye}) \times P(\text{of|eye}) \times P(\text{the|of eye}) \times P(\text{needle|eye of the})
\]
Chain Rule continued

- Problem with sparse data: Many English sentences will not have been seen before
- Not much gained yet because $P(w_n|w_1, w_2, w_3, \ldots, w_{n-1})$ is equally sparse

Example

Could we just do the following?

$$P(\text{needle}|\text{eye of the}) = \frac{\text{Count(eye of the needle)}}{\text{Count(eye of the)}}$$

We cannot just count the number of times words occur in sequence to other words because language is creative and a specific sequence might have never occurred before.
Markov Chain

**Markov assumption:**
- extension of finite automata
- only the previous history matters (older words are less useful)
- only the last \( k \) words are considered in this history

Simplifying assumption:
\[ P(\text{that}| \text{eye of the needle}) \approx P(\text{that}|\text{needle}) \]
Or:
\[ P(\text{that}|\text{eye of the needle}) \approx P(\text{that}|\text{the needle}) \]

\[ P(w_1, w_2, w_3, \ldots, w_n) \approx \prod_i P(w_i|w_{i-k}, \ldots, w_{i-1}) \]
N-Gram Language Models

The simplest type of a language model is the N-gram model. An N-gram is a sequence of N words: 2-gram (bigram), 3-gram (trigram), etc.

<s>I live in Dresden</s>
<s>Dresden is a city</s>
<s>I do not like pigeons in the city</s>

Unigram model:

\[
P(w_1, w_2, w_3, \ldots, w_n) \approx \prod_i P(w_i) = \frac{c(w_i)}{\sum_w c(w)}
\]

\[P(\text{live}) = \frac{1}{18} = 0.05\]
\[P(\text{Dresden}) = \]?
\[P(<s>) = \]

Dagmar Gromann, 30 April 2018 Semantic Computing
<s>I live in Dresden</s>  
<s>Dresden is a city</s>  
<s>I do not like pigeons in the city</s>  

Bigram Model

\[ P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})} \]

\[ P(I|<s>) = \frac{2}{3} = 0.67 \quad P(Dresden|<s>) = \frac{1}{3} = 0.33 \quad P(live|I) = 0.5 \]

\[ P(do|I) = 0.5 \quad P(< /s>|city) = 0.33 \]

\[ P(< s > I \ live \ in \ Dresden < /s >) = P(I|<s>)P(live|I)P(in|live) \]

\[ P(Dresden|in)P(< /s>|Dresden) = ? \]
from nltk.corpus import reuters
from nltk import bigrams
from collections import Counter, defaultdict

first_sentence = reuters.sents()[0]
print(first_sentence)
#Output: ['ASIAN', 'EXPORTERS', 'FEAR', 'DAMAGE', 'FROM', 'U', '.', 'S', ...]
print(list(bigrams(first_sentence, pad_left=True, pad_right=True)))
#Output: [(None, 'ASIAN'), ('ASIAN', 'EXPORTERS'), ('EXPORTERS', 'FEAR'), ...]

model = defaultdict(lambda: defaultdict(lambda: 0))

#Generate a dictionary of counts
for sentence in reuters.sents():
    for w1, w2 in bigrams(sentence, pad_right=True, pad_left=True):
        model[w1][w2] += 1

print(model["the"]["economists"])  
# Output: "economist" follows "the" 8 times
print("Example why padding is useful", model[None]["The"])  
# Output: "The" starts a sentence 8839 times
Bigram Model in Python - continued

```
#Transform counts into probabilities
for w1 in model:
    total_count = float(sum(model[w1].values()))
    for w2 in model[w1]:
        model[w1][w2] /= total_count

print(model["the"]["economists"]) #0.00013733669808243634
print(model[None]["The"]) #0.16154324146501936
```
In practice

- Trigrams are more common than bigrams
- Use of log probabilities instead of probabilities to avoid numerical underflow (the more probabilities we multiply the smaller the product becomes)
- Addition in log space is equivalent to multiplication in linear space

\[ p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4) \]
Evaluation

Main two evaluation methods for most computational linguistic models:

- **Extrinsic evaluation**: measure how much a specific application improves by using your model as compared to the standard baseline (time-consuming!)

- **Intrinsic evaluation**: measure the quality of the model independent of any application

For the intrinsic evaluation, the dataset is split into a:

- Training set: data used to train the model
- Test set: data used to test the trained model using a specific accuracy measure

The model that more accurately predicts the test set is the better model.
Perplexity

The perplexity (PP) of a language model on a test set is the inverse probability of the test set, normalized by the number of words. The more information the model gives us, the lower the perplexity.

\[ PP(W) = P(w_1, w_2, \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \ldots w_N)}} \]

**Chainrule:**

\[ PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_1, w_2, \ldots w_N)}} \]

**Forbigrams:**

\[ PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \]

Minimizing perplexity is equivalent to maximizing probability.
Shannon Game example

The higher the probability the model assigns to the word that actually occurs, the better the model.

Variation of “The Shannon Game”:

I always order pizza with cheese and ___ => mushrooms 0.1, pepperoni, 0.1, anchovies, 0.01
Generalization and Smoothing

Problem 1:
- **Unknown words**: words that never occur in the training but in the test set lead to probabilities of zero! (out of vocabulary (OOV))
- **One solution**: assign the token <UNK> to all OOV in the test set and estimate probability just like for any other word (requires the replacement of words in the training set with <UNK>, e.g. all words with a probability lower than n%).

Problem 2:
- **Unseen contexts**: words are in the vocabulary but appear in an unseen context
- **One solution**: smoothing = it is a form of regularization; individual points are reduced while others are increased leading to a smoother signal
Example Smoothing techniques

- Laplace smoothing: add one to all bigram counts before normalizing
- Backoff: use trigram if there is good evidence, otherwise bigram, otherwise unigram
- Interpolation: mix unigram, bigram, trigram
Text classification
Text classification

One of the most important and typical tasks in supervised machine learning; assigning categories to documents, such as

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

Supervised?

Supervised means an algorithm learns from a number of prelabeled examples (learning by “watching” many examples).
Examples problems

Which of the followings are examples of supervised classification problems?

- Analyze bank data for fraudulent transactions
- Recommend music to someone given their previous choices and many features of that music
- Recognize someone in a photo given a repository of tagged photos
- Cluster students into types based on their learning styles
Classification Method

• Input:
  – a document $d$
  – a fixed set of classes $C = c_1, c_2, \ldots, c_m$
  – a training set $m$ or hand-labeled documents
    $(d_1, c_1), \ldots, (d_m, c_m)$

• Output:
  – a learned classifier $y : d \rightarrow c$
Features and labels

Machine learning takes

- **features** as inputs
- and tries to produce **labels**

We extract features from the raw input: e.g. intensity, tempo, genre, gender of music and then assign the individual features a value based on individual examples
Common Classifiers

- Naïve Bayes
- Logistic regression
- Support-Vector Machines (SVM)
- k-Nearest Neighbors
- ...
Naïve Bayes
Bayes Rule

Probability of an event based on prior knowledge of conditions that might be related to the event.

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} \]

A and B are events, e.g. “wrinkles” and “age”, and \( P(B) \) may not be zero

\( P(A|B) \) = conditional probability of the likelihood of event A occurring given that B is true

\( P(B|A) \) = conditional probability of the likelihood of event B occurring given that A is true

\( P(A) \) and \( P(B) \) are the probabilities of observing A and B independently of each other
## Example

<table>
<thead>
<tr>
<th>Weather</th>
<th>No</th>
<th>Yes</th>
<th>Probability</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>2</td>
<td>3</td>
<td>5 / 14</td>
<td>0.36</td>
</tr>
<tr>
<td>Rainy</td>
<td>3</td>
<td>2</td>
<td>5 / 14</td>
<td>0.36</td>
</tr>
<tr>
<td>Cloudy</td>
<td>4</td>
<td>5</td>
<td>4 / 14</td>
<td>0.29</td>
</tr>
<tr>
<td>All</td>
<td>5</td>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: Likelihood of volley players coming to city beach.

**Question:** Probability of players coming when it is sunny?
Naïve Bayes

- simple classification method based on Bayes rule
- relies on very simple representation of documents: Bag of words
Naïve Bayes Example

- 1% of a population have a disease D with the prior probability $P(D) = 0.01$
- A test is positive 90% if someone has D = sensitivity of the test
- The same test is negative 90% if someone does not have D = specificity of the test

Question: Test = Positive; what is the posterior probability to have D?
Objective

In text classification, our objective is to find the best class $C$ for a specific document $d$. The best class in Naïve Bayes is the most likely or maximum a posterior (MAP) class $c_{\text{map}}$:

$$
c_{\text{map}} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c)
$$

Where $d$ is represented as features $P(x_1, x_2, \ldots, x_n|c)P(c)$
Bag-of-Words (BoW)

Represents the number of known words as a vocabulary and measures their occurrence without considering the order of occurrence.

I live in Dresden
Dresden is a city
I really hate pigeons in the city

unique words = [I, live, in, Dresden, is, a, city, really, hate, pigeons, the]
“I live in Dresden” = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
“Dresden is a city” = [0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0]
Text cleaning steps

• lower-case all words in the vocabulary
• ignore punctuation
• ignore frequent words that do not contain much information, so-called stop words, e.g. “the” or “a”
• correct misspellings
• reduce words to lemmas or stems
Naïve Bayes - Conclusion

Given an input, Naïve Bayes calculates the probability or likelihood that the input belongs to a certain class or in other words the output with the highest probability is most likely the class of the input. While it is easy to implement in a big features space and easy to run, however,

- Naïve Bayes is called “naive” because it ignores the word order.
- Tasks which require word order would break with this algorithm

Code examples: Scikit learn Python library
Review of Lecture 4

- What is the difference between a joint probability and a conditional probability?
- Why is the chain rule alone not enough? What does the Markov assumption add?
- What is text classification?
- How can a language model be evaluated?
- Why is supervised machine learning particularly good on text classification?
- What is the Bayes rule and how is it used in the Naïve Bayes algorithm?
- Which weaknesses does the Naïve Bayes algorithm have?