LINFO2263: Text Completion and Categorization

Pierre Dupont

Outline

Outline

1. Text Completion and Ngram Evaluation

2. Text Categorization

Text Completion

Character or word completion in search engines, query systems, text messaging, source code editors, ...

what time ...	is ... it?	$P(is \mid what\ time)$
	has ... the ... least ... traffic?	$P(has \mid what\ time)$
	from ... now?	$P(from \mid what\ time)$

- Ngram models are evaluated by computing how well they perform
  - when predicting the next word by ranking the predictions according to the probability estimates $\hat{P}(is \mid h)$, $\hat{P}(has \mid h)$, $\hat{P}(from \mid h)$ with h the history, or (partial) left context, of the word to be predicted
  - when repeating such a prediction game (= the Shannon's game) throughout a text which serves as test set
- the quality of this prediction depends on the model order (= the N value) and the smoothing as it influences the probability estimates ($\hat{P}(\dots)$)
- the quality of this prediction is computed according to the test set perplexity metric

Data splitting

- **Basic protocol:** split available corpus into training and test files
  - Use training set (typically 90%) to estimate a model (including the definition of a vocabulary)
  - Use test set (typically 10%) to evaluate its quality (perplexity value)
- **Refinement 1**
  - Split training set into 90% actual training and 10% validation set
  - Validation set is used to
    - tune meta-parameters: pseudo-counts for additive smoothing, λ_i's for interpolation, none with back-off models (discounting values are based on count histograms)
    - possibly select an optimal model order
  - Re-train on the whole training set with any meta-parameter being fixed (except possibly the model order)
- **Refinement 2:** repeat the above using 10-fold cross validation

Test set perplexity

- A test set $S = \{w_1, \ldots, w_M\}$ can be viewed as a single sequence:

$$\underbrace{<s> w_1 \ldots w_n </s>}_{sentence_1} \underbrace{<s> w_{1'} \ldots w_{n'} </s>}_{sentence_2} \ldots$$

- Per-word log likelihood LL

$$LL = \frac{1}{M} \log_2 \left(\prod_{i=1}^{M} \hat{P}(w_i|h) \right) = \frac{1}{M} \sum_{i=1}^{M} \log_2 \hat{P}(w_i|h)$$

  - $\hat{P}(w_i|h)$ is a score (between 0 and 1) of how much the model predicts that the word, at position i in the test, actually follows its (partial) left-context h
  - the likelihood $\hat{P}(w_1, \ldots, w_M) = \prod_{i=1}^{M} \hat{P}(w_i|h)$ measures how likely is the whole test set according to the model
  - $\log_2$ avoids numerical underflow, $\frac{1}{M}$ normalizes according to test set length

- Test set perplexity PP (the lower the better)

$$PP = 2^{-LL} = 2^{\left[-\frac{1}{M} \sum_{i=1}^{M} \log_2 \hat{P}(w_i|h)\right]}$$

- Valid measures only if consistent model: $\forall h, \sum_{w \in V} \hat{P}(w|h) = 1$

Perplexity Computation $PP = 2^{\left[-\frac{1}{M}\sum_{i=1}^{M}\log_2 \hat{P}(w_i|h)\right]}$

<s>	I		am		happy	</s>	<s>	You're	<UNK>	</s>
	the	0.002	will	0.002						
	a	0.001	am	0.0015						
	I	0.0005	would	0.001						
	you	0.00025	was	0.0002						
	...	...	...	...						
	house	0.00004	you	0.00003						
	dollar	0.00003	I	0.00003						
	...	...	...	...						

Bigram:

$\hat{P}(\texttt{am} \mid \texttt{I}) = 0.0015$

<s>	I	am	happy	</s>	<s>	You're	<UNK>	</s>
-	0.0005	0.0015	0.0001	0.001	-	0.0002	0.00004	0.00015

- $-\log_2$ scale

<s>	I	am	happy	</s>	<s>	You're	<UNK>	</s>
	10.96	9.38	13.29	9.97		12.29	11.29	12.70

- $M = 7$ predicted tokens

$$\frac{1}{7}(10.96 + 9.38 + 13.29 + 9.97 + 12.29 + 11.29 + 12.70) = 11.41$$

- Perplexity: $PP = 2^{11.41} = 2721$

Checking consistency: $\sum_w \hat{P}(w|h) = 1, \forall h$

<s>	I		am		happy	</s>	<s>	You're	<UNK>	</s>
	the	0.002	will	0.002						
	a	0.001	am	0.0015						
	I	0.0005	would	0.001						
	you	0.00025	was	0.0002						
	...	...	...	...						
	house	0.00004	you	0.00003						
	dollar	0.00003	I	0.00003						
	...	...	...	...						
	$\sum$	1.0	$\sum$	1.0						

- Each sum must run over all word types in the vocabulary, including <UNK> and </s> but excluding <s> (which is never predicted but used as possible history)
- The consistency should ideally be checked on all possible histories ($\forall h$)
  - When lexicon size is $10,000$, each sum requires $10,000$ additions and there are $O(10^8)$ possible histories for a trigram model
  - This check should be run at least on the observed histories in the test set = the histories considered when computing PP

Perplexity : 0-gram case

Predicting uniformly at random

Assume a vocabulary size $|V|$ = 10,000 word types (including </s> and <UNK>)
0-gram: $\hat{P}(w) = \frac{1}{10{,}000} = 0.0001$ for each word w in the vocabulary

<s>	I		am		happy	</s>	<s>	You're	<UNK>	</s>
	the	0.0001	will	0.0001						
	a	0.0001	am	0.0001						
	I	0.0001	would	0.0001						
	you	0.0001	was	0.0001						
	...	...	...	...						
	house	0.0001	you	0.0001						
	dollar	0.0001	I	0.0001						
	...	...	...	...						

0-gram:

$\hat{P}(\mathrm{am}) = 0.0001$

<s>	I	am	happy	</s>	<s>	You're	<UNK>	</s>
-	0.0001	0.0001	0.0001	0.0001	-	0.0001	0.0001	0.0001

- $-\log_2$ scale

<s>	I	am	happy	</s>	<s>	You're	<UNK>	</s>
	13.29	13.29	13.29	13.29		13.29	13.29	13.29

- $M = 7$ predicted tokens

$$\frac{1}{7}\,(13.29 + 13.29 + 13.29 + 13.29 + 13.29 + 13.29 + 13.29) = 13.29$$

Perplexity properties $PP = 2^{\left[-\frac{1}{M}\sum_{i=1}^{M}\log_2 \hat{P}(w_i|h)\right]}$

- Perplexity is a measure of quality of an iterated Shannon game

$$\ldots \underbrace{w_{i-N+1}\ w_{i-N+2}\ \ldots\ w_{i-1}}_{h}\ \underset{?}{w_i} \ldots$$

- Perplexity is a geometric average representing a gambling cost

$$PP = 2^{\left[-\frac{1}{M}\sum_{i=1}^{M}\log_2 \hat{P}(w_i|h)\right]} = 2^{\left[\log_2\left(\left(\prod_{i=1}^{M}\hat{P}(w_i|h)\right)^{-\frac{1}{M}}\right)\right]} = \sqrt[M]{\prod_{i=1}^{M}\frac{1}{\hat{P}(w_i|h)}}$$

  - the model gambles $\hat{P}(w_i|h)$ on word w_i
  - the model receives 1 \$ to gamble on (actually 1 as total probability mass for a consistent model) for each prediction step i
  - it will cost $\frac{1}{\hat{P}(w_i|h)}$: minimal cost = 1 (no loss), maximal cost ∞
  - the total cost is the geometric average over all prediction steps

Perplexity properties (ctd.)

$$PP = \sqrt[M]{\prod_{i=1}^{M} \frac{1}{\hat{P}(w_i|h)}}$$

- the simplest model (0-gram) predicts uniformly at random
  - $\hat{P}(w_i|h) = \hat{P}(w_i) = \frac{1}{|V|}$
  - Its perplexity is equal to the vocabulary size
    $$PP = \sqrt[M]{\prod_{i=1}^{M} \frac{1}{\hat{P}(w_i)}} = \sqrt[M]{|V|^M} = |V|$$
- the PP value, generally lower than $|V|$ for a well smoothed N-gram model, can be interpreted as the number of words among which one predicts **if** one would predict uniformly at random
  - Example: $|V| = 10,000$ and $PP = 100$ means that the uncertainty is the same as if one would predict only among 100 possibilities and an equal probability $\frac{1}{100}$ for each word
  - The model actually predicts among 10,000 possibilities but with unequal probabilities

Perplexity properties (ctd.)

- Better models have a lower perplexity $\Rightarrow$ they are more predictive
- Unsmoothed models can be much worse than random guessing
  $PP = +\infty$ if $\hat{P}(w_i|h) = 0$ for some i
- A test sample $S = \{w_1, \ldots, w_M\}$ viewed as a single sequence:

$$\underbrace{<\!s\!> w_1 \ldots w_n <\!/s\!>}_{sentence_1} \underbrace{<\!s\!> w_{1'} \ldots w_{n'} <\!/s\!>}_{sentence_2} \ldots$$

  - <s> is never predicted because it is trivial to predict it
  - </s> is predicted because end of sentence matters
  - the actual number M of predictions in a test set of K sentences, each one with n_j "real" words, is $M = \sum_{j=1}^{K}(n_j + 1)$
- Training set PP decreases with the model order N (true with no or minimal smoothing) $\Rightarrow$ avoid overfitting!
- Test set PP is minimal for an optimal order (often close to 3)

Vocabulary and *unknown word*

- The vocabulary or lexicon V is defined
  - either from the application domain (i.e. independently of the data)
  - from the observed word types in the training set

  $\Rightarrow$ some word in the test set (assumed representative of new data) may have never been observed before
- A consistent ($\sum_{w \in V} \hat{P}(w|h) = 1$) and smoothed model ($\hat{P}(w|h) > 0, \forall w \in V, \forall h$) assigns a zero probability to any new word:

  $\hat{P}(w_{new}|h) = 0$ if $w_{new} \notin V \Rightarrow w_{new}$ is never predicted
- Define an additional word type <UNK> as part of the vocabulary and relies on smoothing
  - either from some out-of-vocabulary (OOV) words in the training data
  - or purely from the smoothing mechanism:

  $\hat{P}(\texttt{<UNK>}|h)$ is function of $C(h, \texttt{<UNK>})$, possibly corrected with smoothing
  - with interpolation or backoff to a zero-gram, if necessary:

  $\hat{P}(\texttt{<UNK>}) = \frac{1}{|V|}$

Language Modeling Experiments

Corpus: 98 million words from the Wall Street Journal
Alphabet size = lexicon size = 20,000 distinct word types
Average sequence length = 23
Results for training set with 100,000 sentences ($\sim$ 2 million words)
Test set: 2,500 sentences ($\sim$ 50,000 words)

Results

Model Order	Smoothing	Perplexity
0	Uniform model	20,000
2	Add one	832
3	Add one	1,910
2	Linear Interpolation	194
3	Linear Interpolation	119
2	Back-off smoothing	173
3	Back-off smoothing	101

Outline

Text Categorization

Definition

The task of assigning a label or a category to an entire text or document

Examples

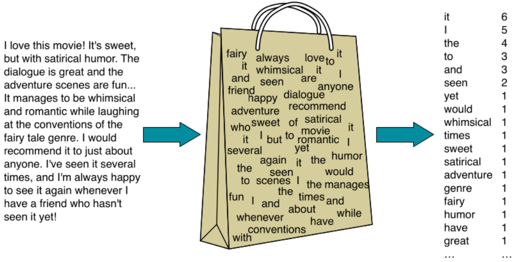
- email spam filtering: email categorization into spam versus ham (= not spam)
- language identification: automatic identification of the language in which a document is written
- topic labeling: assignment of a subject category or topic label to a text
- sentiment analysis: positive or negative orientation of a text (*e.g.* book/movie/restaurant reviews)

  \+ *...awesome caramel sauce and sweet toasty almonds. I love this place!*

  \- *...awful pizza and ridiculously overpriced...*

Bag of Words

In its simplest from, a text document is represented as a bag of words: an unordered set of words with their frequency in the document but ignoring their position



- each word w is a feature and its frequency is the feature value
- the whole document d is represented as a vector made of these feature values $w_1, \ldots, w_n$

Illustration from *Speech and Language Processing, Jurafsky and Martin, 3rd ed.*

Maximum A Posteriori (MAP) classifier

Decision rule

For a document d out of $c \in C$ classes (or categories), this probabilistic classifier returns the class $\hat{c}$ which maximizes the posterior probability

$$\hat{c} = \mathrm{argmax}_{c \in C} P(c \mid d)$$

Bayes rule

$$\hat{c} = \underset{c \in C}{\mathrm{argmax}}\, P(c \mid d) = \underset{c \in C}{\mathrm{argmax}}\, \frac{P(d \mid c)P(c)}{P(d)} = \underset{c \in C}{\mathrm{argmax}}\, P(d \mid c)P(c)$$

Multinomial naive Bayes classifier

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d \mid c)}^{likelihood} \overbrace{P(c)}^{prior} = \underset{c \in C}{\operatorname{argmax}} P(w_1, \ldots, w_n \mid c) P(c)$$

Generative model

An observed document is supposed to come from a random experiment:

1. a class c is drawn at random according to $P(c)$
2. given c, the frequencies of occurrence of words are drawn at random according to $P(w_1, \ldots, w_n \mid c)$

Estimating $P(w_1, \ldots, w_n \mid c)$ is difficult because there are many possible combinations of words and their frequencies, even after ignoring their positions according to the bag of words representation

Naive Bayes assumption

$$P(w_1, \ldots, w_n \mid c) \approx P(w_1 \mid c) \ldots P(w_n \mid c) = \prod_{i=1}^{n} P(w_i \mid c)$$

Conditional independence

The frequency of occurrence of a word is assumed independent of the frequencies of other words, given the class

- this assumption is often violated: *White House, President, . . .*
- yet, it is often satisfactory to categorize documents correctly
  - the probability of occurrence of a word (*e.g. President*) depends on the class (*e.g. Politics* $\neq$ *Computer Science*)

Naive Bayes decision rule

$$c_{NB} = \operatorname*{argmax}_{c \in C} P(c) \prod_i P(w_i \mid c) = \operatorname*{argmax}_{c \in C} \left[\log P(c) + \sum_i \log P(w_i \mid c) \right]$$

Learning the Naive Bayes Classifier

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

- N_c number of documents labeled as class c in a training set
- N_{doc} total number of documents in the training set

$$\hat{P}(w_i \mid c) = \frac{n_{w_i,c}+1}{\sum_{w \in V} n_{w,c}+|V|}$$

- $n_{w_i,c}$ number of occurrences of word w_i in all documents labeled c
- V the vocabulary: by default, all words appearing in the training set
- add-one (Laplace) smoothing to avoid a zero probability if a word in the vocabulary V never appeared in the training set for some class c
- test words not included in the training vocabulary V are simply ignored from the bag of words representation $\Rightarrow$ no need of `<UNK>` when classifying a document

Refinements for Sentiment Analysis

Sentiment analysis is, in its simplest form, a binary classification problem: *positive/negative* sentiment $\Rightarrow$ whether a word occurs or not matters more than its frequency in each document

Binary naive Bayes clips the word counts in each document at 1

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

	NB 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

Illustration from *Speech and Language Processing, Jurafsky and Martin, 3rd ed.*

Refinements for Sentiment Analysis (ctd.)

Negative words (*n't, not, no, never*) modify $+ \Leftrightarrow -$

+	*I really like this movie*
-	*I didn't like this movie*
+	*don't dismiss this movie*

The bag of words (BOW) representation and Naive Bayes consider each word independently of its context

Preprocessing

- prepend the prefix `NOT_` to every word after a token of logical negation till the next punctuation mark

  `didn't like this movie , but I`

  becomes

  `didn't NOT_like NOT_this NOT_movie , but I`
- these additional `NOT_word`s are added to the vocabulary and are part of the Naive Bayes features

Naive Bayes and Language Models

- multinomial naive Bayes is a class conditional unigram: one unigram model for each class (with add-one smoothing)

$$\hat{P}(w_i \mid c) = \frac{n_{w_i,c} + 1}{\sum_{w \in V} n_{w,c} + |V|}$$

- Smoothed class conditional N-grams (with $N > 1$) may be used as features in a MAP classifier. For example, a class conditional 3-gram:

$$\hat{c} = \operatorname*{argmax}_{c \in C} \left[\log \hat{P}(c) + \sum_i \log \hat{P}(w_i | w_{i-1} w_{i-2}; c) \right]$$

  - $i, i-1, i-2$ are now relative positions of the words in the text
  - a distinct model is considered for each class c
  - relative word orders now matters (but not absolute word positions), contrary to the BOW representation, with a possible marginal gain in classification accuracy
  - character N-grams (instead of word N-grams) are often used for language identification

Summary

- Text completion is a common task used in search engines, sms typing, code editors, ...
- N-grams are typically evaluated to perform this task (at the word or character level)
- Test set perplexity measures to which extent a probabilistic model indeed assigns a high probability to a word token occurring in a text when predicting it from its left context
  - the PP of uniform random guessing is the vocabulary size
  - the lower PP the better
  - PP may be larger than the vocabulary size, and even infinite, for a poorly smoothed model
- Bag of Words representation and Naive Bayes assumption are simple yet efficient modeling to categorize text
- Multinomial Naive Bayes is a class conditional unigram model

Perspectives

- Spelling correction is somewhat similar to text completion, except that it needs not be performed from left to right
- Neural methods offer recent alternatives to estimate N-grams and can also be used for
  - text completion
  - sentence generation (= complete iteratively till predicting </s>)
  - text categorization

  ⇒ see lecture on *Deep Learning Methods for Sequence Processing*

Further Reading

Jurafsky D. and Martin J.H.
Speech and Language Processing, 3rd edition (draft),
chapter (3 and) 4.