CE314/CE887 Lecture 3: Language Models and Smoothing

Aline Villavicencio

October, 2017

(Original slides from Annie Louis. Some slides adapted from Sharon Goldwater)

Recap

- A corpus - language samples for use in statistical NLP
- A corpus may have annotations or not
- One very useful statistical model - **language models**
- Probability estimation from corpora is an important component of many NLP models including language models

Language models: assign probabilities to sequences of words in the language

P("all of a sudden I notice three guys standing on the sidewalk")

$>$

P("on guys all I of notice sidewalk three a sudden standing the")

and

P(in | "Please turn your homework")

$>$

P(of | "Please turn your homework")

How to compute the probability of a sentence?

P("all of a sudden I notice three guys standing on the sidewalk")
But most sentences will never be seen even in very large corpora.

1. Decompose probability of a sentence into probability of shorter sequences
2. Estimate probabilities from a corpus (MLE likely to not always work)

1. Decomposing the probability of a sentence or sequence

We want to estimate $P(S = w_1 \dots w_n)$.

- Ex: $P(S = \text{the cat slept quietly})$.
- This is really a joint probability over the words in S: $P(W_1 = \text{the}, W_2 = \text{cat}, W_3 = \text{slept}, \dots W_4 = \text{quietly})$.
- Concisely, $P(\text{the, cat, slept, quietly})$ or $P(w_1, \dots w_n)$.

Using chain rule:
$P(\text{the, cat, slept, quietly}) =$
$= P(\text{quietly}|\text{the, cat, slept})P(\text{slept}|\text{the, cat})P(\text{cat}|\text{the})P(\text{the})$

Decomposing as conditional probabilities does not solve the problem

- More generally, the chain rule gives us:

$$P(w_1, \dots w_n) = \prod_{i=1}^{n} P(w_i | w_1, w_2, \dots w_{i-1})$$

- But many of these conditional probs are just as sparse!
  - If we want
    P(all of a sudden I notice three guys standing on the sidewalk),
  - we still need
    P(sidewalk|all of a sudden I notice three guys standing on the)

So we make an **independence assumption**

- The probability of a word only depends on a fixed number of previous words or **history**
- Generally known as a **Markov assumption**: the probability of a future unit depends on a limited history of previous units
  - 2-word history: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
  - 1-word history: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-1})$
  - no history: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i)$

So we make an **independence assumption**

- The probability of a word only depends on a fixed number of previous words or **history**
- Generally known as a **Markov assumption**: the probability of a future unit depends on a limited history of previous units
  - 2-word history: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
  - 1-word history: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-1})$
  - no history: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i)$

- The probabilities in each (0-word vs 1-word vs 2-word) will be different and the probability of a sentence composed from each may be different
- They are different **models of language**

N-gram Language Models

The difference between the models is the amount of context they capture

- An **n-gram** is a sequence of *n* words.
- The models with limited history define the probability of the last word in an n-gram given previous words

Hence these models are called:

- Trigram model: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
- Bigram model: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-1})$
- Unigram model: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i)$

Trigram independence assumption

- Put another way, trigram model assumes these are all equal:
  - $P(\text{sidewalk}|\text{all of a sudden I notice three guys standing on the})$
  - $P(\text{sidewalk}|\text{I walked on the})$
  - $P(\text{sidewalk}|\text{See the small dog on the})$
  - $P(\text{sidewalk}|\text{They carried out repairs on the})$

  because all are estimated as $P(\text{sidewalk}|\text{on the})$
- Not always a good assumption! But it does reduce the sparse data problem.

Exercise: Sentence probability

What is the equation for
P(See the small dog on the sidewalk) ?

(1) Under a bigram language model
(2) Under a trigram language model

Exercise: Sentence probability

What is the equation for
$P(\text{See the small dog on the sidewalk})$?

(1) Under a bigram language model
(2) Under a trigram language model

Answer:
(1) $= P(\text{See})P(\text{the}|\text{see})P(\text{small}|\text{the})P(\text{dog}|\text{small})P(\text{on}|\text{dog})$
$P(\text{the}|\text{on})P(\text{sidewalk}|\text{the})$

Estimating n-gram probabilities from a corpus

Maximum likelihood estimates (relative frequencies)

Unigram: $P(w_i) = \frac{\text{count}(w_i)}{N}$ where N is the total number of words in the corpus

Bigram: $P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$

Bigram example: $P(\text{“roses”} | \text{“pink”}) = \frac{\text{count(“pink”,”roses”)}}{\text{count(“pink”)}}$

Trigram: $P(w_i | w_{i-2} w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})}$

“Out of k occurrences of a *context*, how often is it followed by the *given* word”

Beginning/end of sequence

What is likely at the beginning and end of sentences?

- To capture behaviour at beginning/end of sequences, we also augment the input:

	w_0	w_1	w_2	w_3	w_4	w_5
(1)	<s>	he	saw	the	cat	</s>
(2)	<s>	i	like	to	swim	</s>

- That is, assume $w_0 =$ <s> and $w_{n+1} =$ </s> so bigram probability:

$$P(w_1, w_2 ... w_n) = \prod_{i=1}^{n+1} P(w_i | w_{i-1})$$

- For trigram estimation we add two dummy start symbols at the beginning of a sentence:

<s><s> he saw the cat </s>

Example bigram estimation

Example bigram estimation

- Consider a corpus with three sentences

$$<s> \text{ I am Sam } </s>$$
$$<s> \text{ Sam I am } </s>$$
$$<s> \text{ I do not like green eggs and ham } </s>$$

Example bigram estimation

- Consider a corpus with three sentences

$$<\text{s}>\ \text{I am Sam}\ </\text{s}>$$
$$<\text{s}>\ \text{Sam I am}\ </\text{s}>$$
$$<\text{s}>\ \text{I do not like green eggs and ham}\ </\text{s}>$$

- Compute the following bigram probabilities

$$P(\text{I}| < s >) = \quad P(\text{Sam}| < \text{s} >) = \quad P(\text{am}|\text{I}) =$$
$$P(< /\text{s} > |\text{Sam}) = \quad P(\text{Sam}|\text{am}) = \quad P(\text{do}|\text{I}) =$$

Example bigram estimation

- Consider a corpus with three sentences

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

- Compute the following bigram probabilities

$$P(\mathrm{I} | <s>) = \quad P(\mathrm{Sam} | <\mathrm{s}>) = \quad P(\mathrm{am} | \mathrm{I}) =$$
$$P(</\mathrm{s}> | \mathrm{Sam}) = \quad P(\mathrm{Sam} | \mathrm{am}) = \quad P(\mathrm{do} | \mathrm{I}) =$$

- Answers

$$P(\mathrm{I} | <s>) = 2/3 \quad P(\mathrm{Sam} | <\mathrm{s}>) = 1/3 \quad P(\mathrm{am} | \mathrm{I}) = 2/3$$
$$P(</\mathrm{s}> | \mathrm{Sam}) = 1/2 \quad P(\mathrm{Sam} | \mathrm{am}) = 1/2 \quad P(\mathrm{do} | \mathrm{I}) = 1/3$$

What was the obvious problem with MLE for language?

If an n-gram is not present in the corpus, its probability is zero

A sentence's probability is composed by multiplying n-gram probabilities. So if any n-gram has probability zero, the probability of the sentence is also zero!

Smoothing: steal some probability mass from seen events and give to unseen events

Laplace smoothing or add-one smoothing
Let's say there are V unique words in the vocabulary (seen + unseen events). Often we may need to come up with an estimate for unseen or **out-of-vocabulary** words.

N is the number of tokens in the corpus

Smoothing: steal some probability mass from seen events and give to unseen events

Laplace smoothing or add-one smoothing

Let's say there are V unique words in the vocabulary (seen + unseen events). Often we may need to come up with an estimate for unseen or **out-of-vocabulary** words.

N is the number of tokens in the corpus

- Just pretend we saw every vocabulary word one more time than we did.

$$\Rightarrow \qquad P_{+1}(w_i) = \frac{C(w_i) + 1}{N} \qquad ?$$

Smoothing: steal some probability mass from seen events and give to unseen events

Laplace smoothing or add-one smoothing

Let's say there are V unique words in the vocabulary (seen + unseen events). Often we may need to come up with an estimate for unseen or **out-of-vocabulary** words.

N is the number of tokens in the corpus

- Just pretend we saw every vocabulary word one more time than we did.

$$\Rightarrow \qquad P_{+1}(w_i) = \frac{C(w_i) + 1}{N} \qquad ?$$

- NO! Sum over possible w_i (in vocabulary V) must equal 1:

Smoothing: steal some probability mass from seen events and give to unseen events

Laplace smoothing or add-one smoothing

Let's say there are V unique words in the vocabulary (seen + unseen events). Often we may need to come up with an estimate for unseen or **out-of-vocabulary** words.

N is the number of tokens in the corpus

- Just pretend we saw every vocabulary word one more time than we did.

$$\Rightarrow \qquad P_{+1}(w_i) = \frac{C(w_i)+1}{N} \qquad ?$$

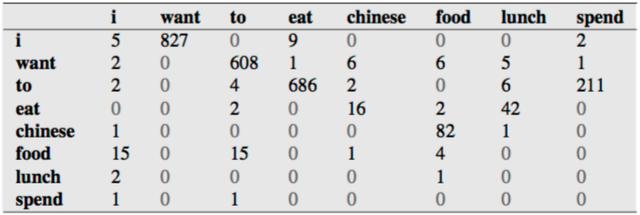
- NO! Sum over possible w_i (in vocabulary V) must equal 1:
- We want:

$$P_{+1}(w_i) = \frac{C(w_i)+1}{N+V}$$

Vocabulary size V

Sometimes we may just assume that there is one unseen vocabulary item `UNK`. Unseen words in the test corpus (when we apply the language model to new data) will be replaced with `UNK` before computing sentence probability.

That does not mean there are whole lot of unseen n-grams! (seen words in unseen contexts)



	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 4.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

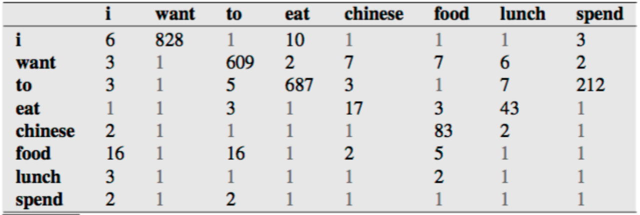
Laplace smoothing for bigrams and trigrams

$$P_{+1}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + 1}{C(w_{i-1}) + V}$$

Similarly for trigrams:

$$P_{+1}(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + 1}{C(w_{i-2}, w_{i-1}) + V}$$

For every context, we assume all vocabulary words occur one more time. Eg. for bigrams,

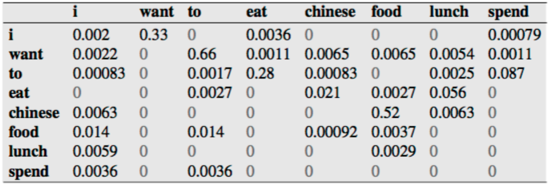


	i	**want**	**to**	**eat**	**chinese**	**food**	**lunch**	**spend**
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Figure 4.5 Add-one smoothed bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Previously-zero counts are in gray.

Laplace smoothing moves too much mass to unseen events

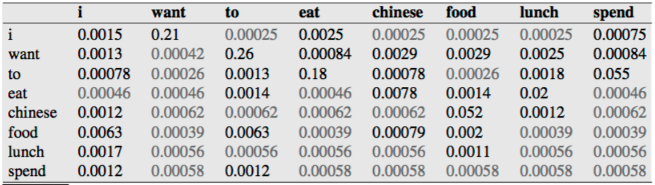
Before smoothing:



	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 4.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

After smoothing:



	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Figure 4.6 Add-one smoothed bigram probabilities for eight of the words (out of $V = 1446$) in the BeRP corpus of 9332 sentences. Previously-zero probabilities are in gray.

Lidstone smoothing or add-k

Rather than add 1, add a value $k < 1$.

$$P_{+1}(w_i) = \frac{C(w_i) + k}{N + kV}$$

$$P_{+1}(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + k}{C(w_{i-1}) + kV}$$

$$P_{+1}(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i) + k}{C(w_{i-2}, w_{i-1}) + kV}$$

How to choose k? Optimize for a certain task, meaning empirically test many values and choose the best one

What measure to use as goodness of a language model? Next class!

Interpolation smoothing

Combine higher and lower order N-gram models, since they have different strengths and weaknesses:

- high-order N-grams are sensitive to more context, but have sparse counts
- low-order N-grams have limited context, but robust counts

If P_N is N-gram estimate (from MLE; $N = 1$ to 3), use:

$$P_{\mathrm{INT}}(w_3|w_1, w_2) = \lambda_1\ P_1(w_3) + \lambda_2\ P_2(w_3|w_2) + \lambda_3\ P_3(w_3|w_1, w_2)$$

$$\begin{aligned} P_{\mathrm{INT}}(\text{food}|\text{want, Chinese}) = \lambda_1\ P_1(\text{food}) + \lambda_2\ P_2(\text{food}|\text{want}) \\ +\lambda_3\ P_3(\text{food}|\text{want, Chinese}) \end{aligned}$$

Interpolation smoothing

- Note that λ_is must sum to 1:

$$\begin{aligned} 1 &= \sum_{w_3} P_{\mathrm{INT}}(w_3|w_1, w_2) \\ &= \sum_{w_3} [\lambda_1\ P_1(w_3) + \lambda_2\ P_2(w_3|w_2) + \lambda_3\ P_3(w_3|w_1, w_2)] \\ &= \lambda_1 \sum_{w_3} P_1(w_3) + \lambda_2 \sum_{w_3} P_2(w_3|w_2) + \lambda_3 \sum_{w_3} P_3(w_3|w_1, w_2) \\ &= \lambda_1 + \lambda_2 + \lambda_3 \end{aligned}$$

The λ_i parameters can again be chosen empirically to optimize evaluation measures on a dataset

Often Laplace or Lidstone are not used

Kneser Ney Smoothing: One of the most powerful smoothing methods

Uses a combination of

- **discounting**: stealing probability mass from higher frequency words
- **a notion of fertility**: Some words are more likely to occur in a new context compared to others. Eg. 'Francisco' (occurs mostly after the word 'San. A word such as 'glasses' can occur in many more contexts). So rather than give the same probability to all unseen events (as in Laplace), we given them unequal probabilities.
- **interpolation**: mix probabilities of higher and lower order n-grams

How are language models used?

It is a flexible framework, you do not always have to think of n-grams in terms of words.

Can also do n-grams of characters, phonemes (sound units) etc.

Use of LM example: Language Identification

You have texts in English, French, German, Italian, Spanish, Latin. Given a new text, which language is it?

- Input: a corpus containing samples of text in each language
- Create **bigram character** models one for each language

- Now, the system is given a new unseen text (not from the corpus used to compute the LM)
- Compute probability of the text under each language model
- Assign the language from the model which gave highest probability to the text

How well do you think this works?

Language and Task Independent Text Categorization with Simple Language Models, Peng et al. NAACL 2003

Use of LM example: Language Identification

You have texts in English, French, German, Italian, Spanish, Latin. Given a new text, which language is it?

- Input: a corpus containing samples of text in each language
- Create **bigram character** models one for each language

- Now, the system is given a new unseen text (not from the corpus used to compute the LM)
- Compute probability of the text under each language model
- Assign the language from the model which gave highest probability to the text

How well do you think this works?
Accuracy is 100%

Language and Task Independent Text Categorization with Simple Language Models, Peng et al. NAACL 2003

Acknowledgements

- These slides were originally prepare by Annie Louis. The Math tutorials are by Sharon Goldwater.