CMPE 409 Machine Translation

Language Model

Murat ORHUN
Adopted from original slides of NIKHIL.P

Istanbul Bilgi University

May 18, 2022

- One essential component of any statistical machine translation system is the language model. Models that assign probabilities to sequences of words are called language models (LMs).
- Which measures how likely it is that a sequence of words would be uttered by an English(may be other languages) speaker.

- Language model typically does much more than just enable fluent output.
- It supports difficult decisions about word order and word translation.
- For instance, a probabilistic language model p LM should prefer correct word order to incorrect word order:
- p LM (the house is small) $>$ p LM (small the is house)
- p LM (I am going home) $>$ p LM (I am going house)

- In the solution of many problems in the natural language processing, statistical language processing techniques can be also used.
    - optical character recognition
    - spelling correction
    - speech recognition
    - machine translation
    - part of speech tagging
    - parsing
- Statistical techniques can be used to disambiguate the input.
- They can be used to select the most probable solution.
- Statistical techniques depend on the probability theory.
- To able to use statistical techniques, we will need corpora to collect statistics.
- Corpora should be big enough to capture the required knowledge.

N-Gram

- The simplest language model that assigns probabilities to sentences and sequences of words is the n-gram.
- An n-gram is a sequence of N words:
    - A 1-gram (unigram) is a single word sequence of words like "please" or " turn".
    - A 2-gram (bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework".
    - A 3-gram (trigram) is a three-word sequence of words like "please turn your", or "turn your homework".
- We can use n-gram models to estimate the probability of the last word of an n-gram given the previous words, and also to assign probabilities to entire word sequences.

Probabilistic language models can be used to assign a probability to a sentenc
many NLP tasks.

Machine Translation:

- P(high winds tonight) > P(large winds tonight)

Spell Correction:

- Thek office is about ten minutes from here
- P(The Office is) > P(Then office is)

Speech Recognition:

- P(I saw a van) >> P(eyes awe of an)

Summarization, question-answering, …

Our goal is to compute the probability of a sentence or sequence of words W ($=w_1,w_2,\ldots w_n$):

- $P(W) = P(w_1,w_2,w_3,w_4,w_5\ldots w_n)$

What is the probability of an upcoming word?:

- $P(w_5|w_1,w_2,w_3,w_4)$

A model that computes either of these:

- $P(W)$ or $P(w_n|w_1,w_2\ldots w_{n-1})$ is called a **language model**.

The intuition of the n-gram model (simplifying assumption):

– instead of computing the probability of a word given its entire history, we can approximate the history by just the last few words.

$P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n)$	**unigram**
$P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n \mid w_{n-1})$	**bigram**
$P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n \mid w_{n-1} w_{n-2})$	**trigram**
$P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n \mid w_{n-1} w_{n-2} w_{n-3})$	**4-gram**
$P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n \mid w_{n-1} w_{n-2} w_{n-3} w_{n-4})$	**5-gram**

In general, **N-Gram** is

$$P(w_n \mid w_1 \ldots w_{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

computing probabilities of word sequences

Unigrams -- $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k)$

Bigrams -- $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1})$

Trigrams -- $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1} w_{k-2})$

4-grams -- $P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1} w_{k-2} w_{k-3})$

computing probabilities of word sequences (Sentences)

Unigram

P(<s> the man from jupiter came </s>) ≈
P(the) P(man) P(from) P(jupiter) P(came)

Bigram

P(<s> the man from jupiter came </s>) ≈
P(the|<s>) P(man|the) P(from|man) P(jupiter|from) P(came|jupiter) P(</s>|came)

Trigram

P(<s> the man from jupiter came </s>) ≈
P(the|<s> <s>) P(man|<s> the) P(from|the man) P(jupiter|man from)
P(came|from jupiter) P(</s>|jupiter came) P(</s>|came </s>)

- In general, a n-gram model is an insufficient model of a language because languages have long-distance dependencies.
    - "The computer(s) which I had just put into the machine room is (are) crashing."
    - But we can still effectively use N-Gram models to represent languages.

Which N-Gram should be used as a language model?

- Bigger N, the model will be more accurate.
  - But we may not get good estimates for N-Gram probabilities.
  - The N-Gram tables will be more sparse.
- Smaller N, the model will be less accurate.
  - But we may get better estimates for N-Gram probabilities.
  - The N-Gram table will be less sparse.
- In reality, we do not use higher than Trigram (not more than Bigram).
- How big are N-Gram tables with 10,000 words?
  - Unigram -- 10,000
  - Bigram – 10,000*10,000 = 100,000,000
  - Trigram – 10,000*10,000*10,000 = 1,000,000,000,000

N-Grams and Markov Models

- The assumption that the probability of a word depends only on the previous word(s) is called Markov assumption.
- Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far into the past.
- A bigram is called a first-order Markov model (because it looks one token into the past);
- A trigram is called a second-order Markov model;
- In general a N-Gram is called a N-1 order Markov model.

Estimating N-Gram Probabilities

Estimating n-gram probabilities is called **maximum likelihood estimation** (or **MLE**)

We get the MLE estimate for the parameters of an n-gram model by ***getting counts from a corpus***, and **normalizing** the counts so that they lie between 0 and 1.

Estimating bigram probabilities:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

where C is the count of that pattern in the corpus

Estimating N-Gram probabilities

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

A Bigram Example

A mini-corpus: We augment each sentence with a special symbol <s> at the beginning of the sentence, to give us the bigram context of the first word, and special end-symbol </s>.

<s> I am Sam </s>
<s> Sam I am </s>
<s> I fly </s>

Unique words: I, am, Sam, fly

Bigrams: <s> and </s> are also tokens. There are 6(4+2) tokens and 6*6=36 bigrams

<table>
<tr><td>P(I|<s>)=2/3</td><td>P(Sam|<s>)=1/3</td><td>P(am|<s>)=0</td><td>P(fly|<s>)=0</td><td>P(<s>|<s>)=0</td><td>P(</s>|<s>)=0</td></tr>
<tr><td>P(I|I)=0</td><td>P(Sam|I)=0</td><td>P(am|I)=2/3</td><td>P(fly|I)=1/3</td><td>P(<s>|I)=0</td><td>P(</s>|I)=0</td></tr>
<tr><td>P(I|am)=0</td><td>P(Sam|am)=1/2</td><td>P(am|am)=0</td><td>P(fly|am)=0</td><td>P(<s>|am)=0</td><td>P(</s>|am)=1/2</td></tr>
<tr><td>P(I|Sam)=1/2</td><td>P(Sam|Sam)=0</td><td>P(am|Sam)=0</td><td>P(fly|Sam)=0</td><td>P(<s>|Sam)=0</td><td>P(</s>|Sam)=1/2</td></tr>
<tr><td>P(I|fly)=0</td><td>P(Sam|fly)=0</td><td>P(am|fly)=0</td><td>P(fly|fly)=0</td><td>P(<s>|fly)=0</td><td>P(</s>|fly)=1</td></tr>
<tr><td>P(I|</s>)=0</td><td>P(Sam|</s>)=1/3</td><td>P(am|</s>)=1/3</td><td>P(fly|</s>)=1/3</td><td>P(<s>|</s>)=0</td><td>P(</s>|</s>)=0</td></tr>
</table>

Uni-Trigram Example

Example

<s> I am Sam </s>
<s> Sam I am </s>
<s> I fly </s>

Unigrams: I, am, Sam, fly

P(I)=3/8 P(am)=2/8 P(Sam)=2/8 P(fly)=1/8

Trigrams: There are 6*6*6=216 trigrams.

- Assume there are two tokens <s> <s> at the begining, and two tokens </s> </s> at the end.

P(I|<s> <s>)=2/3 P(Sam|<s> <s>)=1/3
P(am|<s> I)=1/2 P(fly|<s> I)=1/2
P(I|<s> Sam)=1
P(Sam|I am)=1/2 P(</s>|I am)=1/2
P(</s>|am Sam)=1
P(</s>|Sam </s>)=1

Corpus: Berkeley Restaurant Project Sentences

- There are 9222 sentences in the corpus.
- Raw biagram counts of 8 words (out of 1446 words)

	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

Corpus: Berkeley Restaurant Project Sentences

	i	want	to	eat	chinese	food	lunch	spen
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

Unigram counts:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Normalize bigrams by unigram counts:

	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

Bigram Estimates of Sentence Probabilities

Some other bigrams:

P(i|<s>)=0.25

P(english|want)=0.0011

P(food|english)=0.5

P(</s>|food)=0.68

	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

Compute the probability of sentence **I want English food**

P(<s> i want english food </s>)

= P(i|<s>) P(want|i) P(english|want) P(food|english) P(</s>|food)

= 0.25*0.33*0.0011*0.5*0.68

= 0.000031

Further Reading

- Jurafsky, D. and J. H. Martin. Speech and language processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Second Edition, Upper Saddle River, NJ: Prentice-Hall, 2008.
- Statistical Machine Translation, Philipp Koehn (chapter 7).