Probabilistic Language Modeling
Introduction to N-grams

Mustafa Jarrar

University of Birzeit

Watch this lecture
and download the slides

Course Page: http://www.jarrar.info/courses/AI/

More Online Courses at: http://www.jarrar.info

Acknowledgement: This lecture is largely based on Dan Jurafsky's online course on NLP, which can be accessed at http://www.youtube.com/watch?v=s3kKlUBa3b0

Outline

- ❑ Probabilistic Language Models
- ❑ Chain Rule
- ❑ Markov Assumption
- ❑ N-gram
- ❑ Example
- ❑ Available language models
- ❑ Evaluate Probabilistic Language Models

Keywords: Natural Language Processing, NLP, Language model, Probabilistic Language Models Chain Rule, Markov Assumption, unigram, bigram, N-gram, Curpus

المعالجة الآلية للغات ,مدونة , ماركوف فرضية , تطبيقات لغوية , التحليل اللغوي إحصائيا , الغموض اللغوي , اللسانيات الحاسوبية الطبيعية

Why Probabilistic Language Models

Goal: assign a probability to a sentence ("as used by native speakers")

Why do we need probabilistic language models?

Machine Translation: to generate better translations
P(**high** winds tonite) **>** P(**large** winds tonite)

Spell Correction: to the much more likely to happen(i.e., more correct)
The office is about fifteen **minuets** from my house
P(about fifteen **minutes** from) **>** P(about fifteen **minuets** from)

Speech Recognition
P(I saw a van) **>>** P(eyes awe of an)

+ Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

Goal: given a corpus, compute the probability of a sentence W (or sequence of words w_1 w_2 w_3 w_4 $w_5 \ldots w_n$):

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

P(How to cook rice) = P(How, to, cook, rice)

Related task: probability of an upcoming word. That is, given the sentence (w_1,w_2,w_3,w_4), what is the probability that w_5 will be the next word:

$P(w_5 | w_1,w_2,w_3,w_4)$ //P(rice| how, to, cook)

related to $P(w_1,w_2,w_3,w_4,w_5)$ //P(how, to, cook, rice)

A model that computes:

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

Better: **The grammar = language model**

➔ **Intuition: let's rely on the Chain Rule of Probability**

Reminder: The Chain Rule

Recall the definition of conditional probabilities:

$$P(A|B) = \frac{P(A,B)}{P(B)} \quad \xrightarrow{\text{Rewriting}} \quad P(A|B) \times P(B) = P(A,B)$$

or $P(A,B) = P(A|B) \times P(B)$

More variables:

$$P(A,B,C,D) = P(A) \times P(B|A) \times P(C|A,B) \times P(D|A,B,C)$$

Example: P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)× P(so|its water is)× P(transparent | its water is so)

The Chain Rule in general is:

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

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

How to Estimate Probabilities

Given a large corpus of English (that represents the language), should we just divide all words and count all probabilities?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

No! Too many possible sentences!
We'll never have enough data (the counts of all possible sentences) for estimating these.

Markov Assumption

Based on [2]

Instead, we apply a simplifying assumption:

Andrei Markov
(1856–1922),
Russian mathematician

<u>Markov suggests</u>: Instead of the counts of all possible sentences **it is enough to only count the last few words**

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i \mid w_{i-k} \dots w_{i-1})$$

In other words, approximate each component in the product (this is enough)

$$P(w_i \mid w_1 w_2 \dots w_{i-1}) \approx P(w_i \mid w_{i-k} \dots w_{i-1})$$

Example:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{that})$$

Or maybe better:

$$P(\text{the} \mid \text{its water is so transparent that}) \approx P(\text{the} \mid \text{transparent that})$$

Unigram Model -Simplest case of Markov Model

Estimate the probability of whole sequence of words by the product of probability of individual words:

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

P(its water is so transparent that the) ≈
P(its) x P(water) x P(is) x P(so) x P(transparent) x P(that) x P(the)

Example of some automatically generated sentences from a unigram model, (words are independent):

```
fifth, an, of, futures, the, an, incorporated, a, a,
the, inflation, most, dollars, quarter, in, is, mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

➔ This is not really a useful model

Bigram Model

Condition on the previous word:

Estimate the probability of a word given the entire prefix (from the begging to the pervious word) only by the pervious word.

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

P(its water is so transparent that the) ≈ P(water | its) x P(is | water)) x P(so | is) x P(transparent | so) x P(that | transparent) x P(the | that)

➔ The used conditioning (bigram) is still producing something is wrong/weak!

N-gram models

We can extend to 3-grams, 4-grams, 5-grams

In general this is an insufficient model of language

- because language has **long-distance dependencies**:

Predict: "the computer crashed"!!

"The computer which I had just put into the machine room on the fifth floor crashed."

This means that we have to consider lots of long sentences.

But in practice we can often get away with N-gram model.

Estimating Bigram Probabilities

The Maximum Likelihood Estimate

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

if we have word w_{i-1}, how many times it was followed by word w_i

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Example:

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Sample Corpus

<s> I am Sam </s>

<s> Sam I am </s>

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

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

Estimating Bigram Probabilities

The Maximum Likelihood Estimate

if we have word w_{i-1}, how many times it was followed by word w_i

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Example: Sample Corpus

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

$$P(\texttt{I} \mid \texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam} \mid \texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am} \mid \texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>} \mid \texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam} \mid \texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do} \mid \texttt{I}) = \tfrac{1}{3} = .33$$

Another Example

Given this larger corpus

… can you tell me about any good cantonese restaurants close by
mid priced thai food is what i'm looking for
tell me about chez panisse
can you give me a listing of the kinds of food that are available
i'm looking for a good place to eat breakfast
when is caffe venezia open during the day …

Bigram Counts (Out of 9222 sentences)

	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

Many counts are Zero

Raw bigram probabilities

Normalizing the previous table/counts with the following:

Normalize by unigrams:

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

Result:

	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

P(<s> I want english food </s>) ≈

P(I|<s>)

× P(want|I)

× P(english|want)

× P(food|english)

× P(</s>|food)

= .000031

What kinds of knowledge?

P(english | want) = .0011
P(chinese | want) = .0065
P(to | want) = .66
P(eat | to) = .28
P(food | to) = 0
P(want | spend) = 0
P (i | <s>) = .25

➔ These numbers reflect how English is used in practice (**our corpus**).

Practical Issues

In practice we don't keep these probabilities in the form of probabilities, we keep them in the form of log probabilities. That is, we do everything in log space for two reasons:

- Avoid underflow (as we multiply many small numbers yield arithmetic underflow)
- Adding is faster than multiplying.

$$p_1 \times p_2 \times p_3 \times p_4 = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Available Resources

There are many available Language models that you can try

Google N-Gram Release, August 2006

AUG 3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

…

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google N-Gram Release

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Update (22 Sept. 2006): The LDC now has the data available in their catalog. The counts are as follows:

```
File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens:    1,024,908,267,229
Number of sentences:    95,119,665,584
Number of unigrams:         13,588,391
Number of bigrams:         314,843,401
Number of trigrams:        977,069,902
Number of fourgrams:     1,313,818,354
Number of fivegrams:     1,176,470,663
```

The following is an example of the 3-gram data contained this corpus:

```
ceramics collectables collectibles 55
ceramics collectables fine 130
ceramics collected by 52
ceramics collectible pottery 50
ceramics collectibles cooking 45
ceramics collection , 144
```

Other Models

Google Book N-grams Viewer
http://ngrams.googlelabs.com/

SRILM - The SRI Language Modeling Toolkit
http://www.speech.sri.com/projects/srilm/

How to know a language is model is good?

Does the language model prefer good sentences to bad ones?

- Assign higher probability to "real" or "frequently observed" sentences
  - Than "ungrammatical" or "rarely observed" sentences?

Train parameters of our model on a **training set**.

Test the model's performance on data you haven't seen.

- A **test set** is an unseen dataset that is different from our training set, totally unused.
- An **evaluation metric** tells us how well our model does on the test set.

➔ Two way to evaluate a language model

- ❖ **Extrinsic** evaluation (**in-vivo**)
- ❖ **intrinsic** evaluation (**perplexity)**

Extrinsic evaluation of N-gram models

Best evaluation for comparing models A and B

- Put each model in a task
  - spelling corrector, speech recognizer, MT system
- Run the task, get an accuracy for A and for B
  - How many misspelled words corrected properly
  - How many words translated correctly
- Compare accuracy for A and B

➔ Extrinsic evaluation is time-consuming; can take days or weeks

Intuition of Perplexity (intrinsic evaluation)

– How well can we predict the next word?

I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

….

and 1e-100

A better model of a text

– is one which assigns a higher probability to the word that actually occurs, gives, Gives the highest P(sentence).

Perplexity is the probability of the test set, normalized by the number of words:

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

Minimizing perplexity is the same as maximizing probability

Project

Develop an auto complete web form, based on a 3-gram language model.

Each student need to collect an Arabic corpus of 10000 words (10 documents) at least. Students can use the same corpus, fully or partially.

Tokenize the corpus into tokens/words, then build a tri-gram language model for this corpus. That is, your language = Table that contains word counts + table that contains the probability (or log) of each 3-grams.

Develop an autocomplete web form that is able to uses your language model to autocomplete what users write (no matter how long their queries).

Deadline: ?????

Idea for Graduate Project

Take Curras (a well annotated corpus for the Palestinian dialect, developed at Sina Institute), and build and evaluate a language model for this corpus.

References

[1] Dan Jurafsky:From Languages to Information notes
http://web.stanford.edu/class/cs124

[2] Wikipedia: Andrei Markov
http://en.wikipedia.org/wiki/Andrey_Markov