Language Model

Preetham Reddy Armoor

Department of Mathematics
Hochschule,Mittweida

9 January, 2018

Outline

Introduction

Language Model: Probability distribution over a sequence of words

Introduction

Language Model: Probability distribution over a sequence of words
Application:

- spell correction
  Example
  The office is about fifteen **minuets** from my house
  P(fifteen **minutes**) $>$ P(fifteen **minuets**)

Introduction

Language Model: Probability distribution over a sequence of words
Application:

- spell correction
  Example
  The office is about fifteen **minuets** from my house
  P(fifteen **minutes**) $>$ P(fifteen **minuets**)
- Speech recognition
  P(I saw a van) $>$ P(eyes awe of an)

Introduction

we have some (finite) vocabulary,
V = (the, a, man, saw, Beckham,telescope, two,fan, play, for, Real Madrid)

Introduction

we have some (finite) vocabulary,
V = (the, a, man, saw, Beckham,telescope, two,fan, play, for, Real Madrid)
Example:
START the STOP
START a STOP
START the fan STOP
START the fan saw Beckham STOP
START the fan saw saw STOP
START the fan saw Beckham play for Real Madrid STOP
(infinite) sequence of words $V^{'}$

Language Model

Example:

START the fan saw Beckham STOP

probability of a sentence or sequence of words:

Example:
START the fan saw Beckham STOP
probability of a sentence or sequence of words:

$$P(W) = P(START, the, fan, saw, Beckham, STOP)$$

$$P(W) = P(w_1, w_2, w_3, w_4, w_5, w_6)$$

Related task:probability of an upcoming word:

$$P(W) = P(Beckham|START, the, fan, saw)$$

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

Example:
START the fan saw Beckham STOP
probability of a sentence or sequence of words:

$$P(W) = P(START, the, fan, saw, Beckham, STOP)$$

$$P(W) = P(w_1, w_2, w_3, w_4, w_5, w_6)$$

Related task:probability of an upcoming word:

$$P(W) = P(Beckham|START, the, fan, saw)$$

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

depends on Joint Probability and Conditional Probability

Language Model

Estimate the probability of sentence or sequence of words:

$$p(w_i | w_1, w_2, w_3, ..., w_{i-1}) = \frac{count(w_1, w_2, w_3, ..., w_{i-1}, w_i)}{S}$$

where S is total number of sentence in training data
Not too many possible sentences

Markov Model

simplifying assumption:

$$P(w_5|w_1, w_2, w_3, w_4) \approx P(w_5|w_4)$$

simplifying assumption:

$$P(w_5|w_1, w_2, w_3, w_4) \approx P(w_5|w_3, w_4)$$

or

$$P(w_i|w_1, w_2, w_3, ..., w_{i-1}) \approx P(w_i|w_{i-k}, ..., w_{i-1})$$

Markov Model

simplifying assumption:

$$P(w_5|w_1, w_2, w_3, w_4) \approx P(w_5|w_4)$$

simplifying assumption:

$$P(w_5|w_1, w_2, w_3, w_4) \approx P(w_5|w_3, w_4)$$

or

$$P(w_i|w_1, w_2, w_3, ..., w_{i-1}) \approx P(w_i|w_{i-k}, ..., w_{i-1})$$

In general from chain rule, we can approximate each sentence in the product

$$P(w_1, w_2, w_3, w_4) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)P(w_4|w_1, w_2, w_3)$$

N-gram model

Unigram model

$$P(w_1, w_2, w_3, ..., w_n) \approx \prod_i P(w_i)$$

bigram model

$$P(w_1, w_2, w_3, ..., w_n) \approx \prod_i P(w_i|w_{i-1})$$

We can extend to trigram, 4gram, 5gram

N-gram model

Estimating bigram probability

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

Example:
Training data
START i am sam STOP
START sam i am STOP
START i like traveling STOP

$$P(like|i) = \frac{1}{3}$$

$$P(sam|am) = \frac{1}{2}$$

N-gram model

Bigram estimating sentence probability
P(START i like traveling STOP) =

$$P(i|START) * P(like|i) * P(traveling|like) * P(traveling|STOP)$$

$$= \frac{2}{3} * \frac{1}{3} * \frac{1}{1} * \frac{1}{3}$$

$$= \frac{2}{27}$$

Evaluation of N-gram model:Perplexity

We have some test data, m sentences $s_1, s_2, s_3, ..., s_m$
Perplexity is the probability of a the test set,normalized by the number of words

$$perplexity = 2^{-l}$$

where

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(s_i)$$

and M is the total number of words in the test data

Evaluation of N-gram model:Perplexity

Example
we have a vocabulary V and $N = |V|$
and zero frequency word in test set
estimating a probability

$$p(zerofrequencyword|w_{i-k}, ..., w_{i-1}) = \frac{0}{n}$$

Perplexity will be infinity

Evaluation of N-gram model:smoothing

smoothing:Generalization of perplexity
laplace smoothing(add-one):

- adds one to the zero frequency words

bigram probability

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

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

Evalution of N-gram model:smoothing

Good Turing smoothing:

$$P_{GoodTuring}(wordswithzerofrequency) = \frac{N_1}{N}$$

N_c = the count of words we have seen c times
$N = |v|$

$$P_{GoodTuring}(wordswithfrequency) = \frac{c^*}{N}$$

where

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Reference I

Christopher D.Manning,Prabhakar Raghavan,Hinrich Schuetze.
An Introduction to Information Retrieval.
Cambridge UP, 2009.

Michael Collins.
Course notes for NLP
Columbia University

Favian Contreras .
Python Ngram application
https://github.com/BigFav