Natural Language Processing 1

Lecture 2: Language models and part-of-speech tagging

Katia Shutova

ILLC
University of Amsterdam

28 October 2020

Outline.

video: Language models

Probabilistic language modelling

Part-of-speech (POS) tagging

Q&A and Discussion

Modelling word sequences

this is not true.
in reality, words are dependent
to each other

- We have seen the bag-of-words technique
- where each word is treated as independent from its context
- In reality, word likelihood depends on context
- This lecture introduces shallow syntax:
  language modelling, i.e. modelling word sequences using statistical techniques

Corpora

a collection of text. =trainset
corpus might have annotations,
might not have annotations.
if we use POS tags as annotation
for the text, then we say this
is tagged corpus(=tag+text)

in uncurpervised learning,
when design corpus, we want it to be
balanced. means: it does not have bias. bias
can be caused by domain or genre. we want our
trainset (corpus) to be representitative in
reality

- corpus: text that has been collected for some purpose.
- balanced corpus: texts representing different genres
  genre is a type of text (vs domain)
- tagged corpus: a corpus annotated with e.g. POS tags
- treebank: a corpus annotated with parse trees
- specialist corpora — e.g., collected to train or evaluate particular applications
  - Movie reviews for sentiment classification
  - Data collected from simulation of a dialogue system

parsing trees: 语法分析

Language modelling and word prediction

Guess the missing word:

Wright tells her story with great _____.

Language modelling and word prediction

Guess the missing word:

this word must be a noun,
can be feeling or semantic
meaning

Wright tells her story with great professionalism.

Uses of language modelling

- speech recognition to disambiguate results from signal processing:
  - *have an ice Dave*
  - *heaven ice day*
  - *have a nice day*

speech recognition中，input: voice, output: a sentence. 这三句话都有可能。为了确定哪句话是最有可能的，we use NLP modelling.NLP gives which sentence has the largest prob

language translation中，input: a sentence. output: a sentence. output has several options. we use NLP modelling to rank the prob of each sentence. find: which sentence has largest prob

- word prediction for communication aids:
  e.g., to help enter text that's input to a synthesiser
- text entry on mobile devices

NLP can be used in:
when typing an email, what word is the most probable in the next word

- spelling correction
- ...

n-grams

Bigram: n-gram with N=2

- A probability is assigned to a word **based on the previous word**:

$$P(w_n|w_{n-1})$$

  where w_n is the nth word in a sentence.

- Probability of a sequence of words (assuming **independence**):

$$P(W_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

n-grams

Trigram: n-gram with N=3

- A probability is assigned to a word **based on two previous words**:

$$P(w_n | w_{n-1} w_{n-2})$$

  where w_n is the nth word in a sentence.

- Probability of a sequence of words (assuming **independence**):

$$P(W_1^n) \approx \prod_{k=1}^{n} P(w_k | w_{k-1} w_{k-2})$$

bigrams: probability estimation

约等于 because this equation does not take "beginning" and "end"of sentence" into account.
this equation is true, except for the very last word in the corpus.

Maximum likelihood estimation:

count how often we see the seq where "w_n-1 is followed by w_n" in our corpus

Probability is estimated from counts in a training corpus:

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

?

i.e. count of a particular bigram in the corpus divided by the count of all bigrams starting with the prior word.

start of the sentence occurs by 5 times.

"start of the sentence followed by good"occurs by 3 times.

⟨s⟩ good morning ⟨/s⟩ ⟨s⟩ good afternoon ⟨/s⟩ ⟨s⟩ good afternoon ⟨/s⟩ ⟨s⟩ it is very good ⟨/s⟩ ⟨s⟩ it is good ⟨/s⟩

sequence	count	bigram probability
⟨s⟩	5	
⟨s⟩ good	3	.6
⟨s⟩ it	2	.4
good	5	
good morning	1	.2
good afternoon	2	.4
good ⟨/s⟩	2	.4

...

3/5 = 0.6

2/5 = 0.4

Sentence probabilities

green text: is trainset or corpus

testset. testset is an unseen sentence. but the words in the testset are all seen in the trainset, in this example.we make prediction on test sentence, based on what we learned in trainset (in corpus)

$\langle$s$\rangle$ good morning $\langle$/s$\rangle$ $\langle$s$\rangle$ good afternoon $\langle$/s$\rangle$ $\langle$s$\rangle$ good afternoon $\langle$/s$\rangle$ $\langle$s$\rangle$ it is very good $\langle$/s$\rangle$ $\langle$s$\rangle$ it is good $\langle$/s$\rangle$

Probability of $\langle$s$\rangle$ it is good afternoon $\langle$/s$\rangle$ is estimated as:

$P(\text{it}|\langle\text{s}\rangle)P(\text{is}|\text{it})P(\text{good}|\text{is})P(\text{afternoon}|\text{good})P(\langle/\text{s}\rangle|\text{afternoon})$
$= .4 \times 1 \times .5 \times .4 \times 1 = .08$

prob=0
because in corpus(trainset), we have never seen the bigram"<s> followed by very". this is a very common problem in reality

What about the probability of $\langle$s$\rangle$ very good $\langle$/s$\rangle$?
$P(\text{very}|\langle\text{s}\rangle)$?

Sentence probabilities

problem: sequence in the test sentence does not exist in trainset (corpus)
solution:
smoothing
backoff and interpolation

Problems because of sparse data:

- smoothing: distribute 'extra' probability between rare and unseen events
- backoff and interpolation: approximate unseen probabilities by a more general probability, e.g. unigrams

cf Chomsky: *Colorless green ideas sleep furiously*

smoothing means unseen phrases have a non-zero probability estimate.

use the prob in lower-order gram to replace the prob in higher-order gram.

eg. we never seen the bigram "<s> very" in trainset(corpus), so we dont know its prob, but we know the prob for unigram "very". we can use prob of unigram to replace bigram, for this word sequence

eg, never seen a certain trigram. we use prob of bigram to replace prob of the trigram

Laplace (add 1) smoothing

we add 1 for each count on the bigram

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

size of vocabulary
if a word appears 10 times, what is recorded in vocabulary?

- simple to implement, BUT
- only suitable for problems with few unseen events
- we have a lot of unseen n-grams

in the case that: we have a lot of bigrams which has count=0. +1 on all of them, means we are shifting a lot of prob masks from frequent events to rare events.

so this does not give a good model

But add-1 is used to smooth other NLP models:

- e.g. for text classification
- in domains where the number of zeros isn't so huge

Backoff and Interpolation

- Sometimes it helps to use **less context**
  - Condition on less context for contexts you haven't learned much about
- Backoff
  - use trigram if you have good evidence,
  - otherwise bigram, otherwise unigram
- Interpolation
  - mix unigram, bigram, trigram
  - Interpolation works better

Linear interpolation

is weighted sum.
sum of the prob

we compute this for all trigram
probability

- Combine **different order n-grams**
- by linearly interpolating all the models:

$$\hat{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n),$$

trigram bigram unigram

such that $\sum_i \lambda_i = 1$

- λs are learned from a held-out corpus

More options

Advanced smoothing methods:

- Absolute discounting
- Good Turing smoothing
- Kneser-Ney smoothing
- ...

See Chapter 3 in Jurafsky & Martin (3 edition) for more details

- Neural language models (later in the course)

Handling unknown words

eg unknow word is carrot.
the prob of "carrot" can come
from :
prob of "eat" ,
prob of "buy"
prob of "soup"
prob of "restaurant"
ect

- Most tasks in NLP are open vo
- Test data will contain out of vocabulary (OOV) words
- Create an unknown word token <UNK>
- Train <UNK> probabilities
  - Create a fixed lexicon L of size V
  - in the corpus, replace all words not in L with <UNK>
  - train its probabilities like a normal word
  - use UNK probabilities for any OOV word

Using n-grams to generate sequences

Some Shakespeare...

```
2 gram: does not follow gramma
3 gram: we can see some gramma. 3 is better than 2
4 gram: it is very close to the original shakespere trainset
```

2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Using n-grams to generate sequences

Wall Street Journal

2 gram	Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
3 gram	They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

the output which is generated by 3 gram is better than 2 gram

because the output is a reflection of the trainset

Limitations of n-gram models

n-gram model works well in most cases. but it has a disadvantage:

主语谓语相距太远

- In general this is an insufficient model of language
- because language has long-distance dependencies:

  The computer which I had just put into the machine room on the fifth floor is crashing.

- But we can often get away with N-gram models

Limitations of n-gram models

- In general this is an insufficient model of language
- because language has long-distance dependencies:

  The computer which I had just put into the machine room on the fifth floor is crashing.

- But we can often get away with N-gram models

Evaluation of language models

we test the model on the same task as its trainset.
eg. for task XXX, we have trainset and testset. we test model A on its testset.

1. Intrinsic evaluation
   - evaluate directly on a test set designed for the task at hand
   - using some metric
   - for LMs — perplexity
2. Extrinsic evaluation
   - evaluate in the context of some external task
   - e.g. speech recognition, machine translation

we have model A and B
if we test A and B on the task: speech recognition, A is better than B
if we test A and B on the task: machine translation, B is better than A
model A and B were trained on task X. we test them on a different task Y.
goal: so we know is model A good for solving a different task?
but it is resource consuming and testing is too slow.

Perplexity

we compute perplecity of model A and B.
to compare the perplexity of A and B
find out which model is better

Intuition: The best language model is one that best predicts an unseen test set (i.e. with the highest probability)

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

$$PP(W) = P(w_1, w_2, ..., w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, ..., w_N)}}$$

P: prob of testset (seq of words)
1/P: we take its inverse
we normalize it by the nb of words in the testset

- For bigrams:

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

we want to maximize the prob, means to minimize the 1/prob, minimize perplexity

- **Minimize** perplexity

in case of bigram, we need to compute the prob of bigrams in testset.
in case of trigram, we need to compute the prob of trigrams in testset.

Lower perplexity = better model

- Wall Street Journal corpus
- Train on 38 million words
- test on 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

= index of confusion

the lower, the better the model is

perplexity of unigram is the highest.
means: unigram is the worst model among these 3 models.

because unigram is just the frequency of word, does not take context into account.

Problem with intrinsic evaluation of LMs

the more similar is (trainset is similar as testset), the lower the perplexity is

- depends on how different the test and training set are
- not comparable across datasets
- but useful for pilot experimentation

So extrinsic evaluation is better, but time-consuming