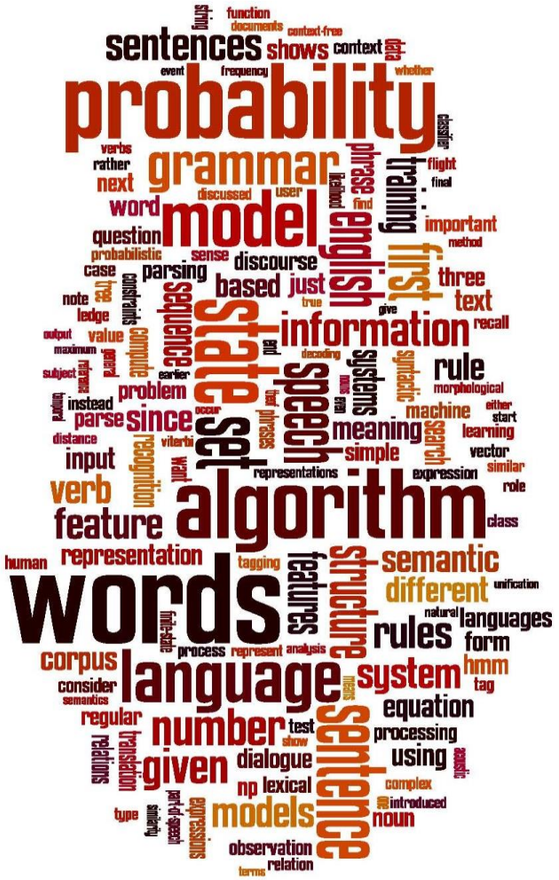
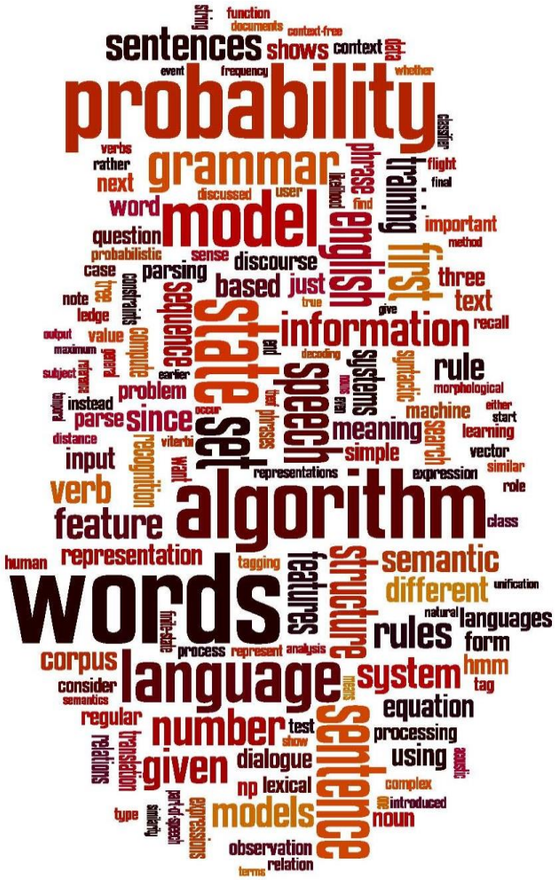


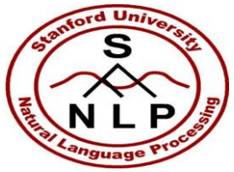
Language Modeling

Introduction to N–grams



Language Modeling

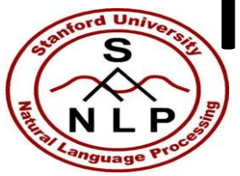
Evaluation and Perplexity



Types of Evaluation

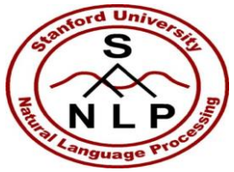
Extrinsic evaluation: Evaluate the performance of a language model by embedding it in an application and measure how much the application improves.

Intrinsic evaluation: Measures the quality of a model independent of any application.



Intrinsic Evaluation: How good is our model?

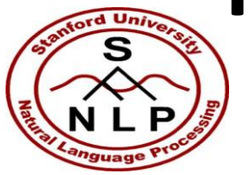
- Does our language model prefer good sentences to bad ones?
  - Assign higher probability to “real” or “frequently observed” sentences
    - Than “ungrammatical” or “rarely observed” sentences?
- We train parameters of our model on a **training set**.
- We test the model’s performance on data we haven’t 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.



Extrinsic evaluation of N-gram models

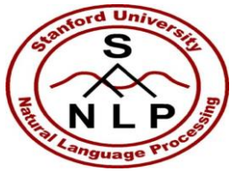
- Best evaluation for comparing models A and B
  - Put each model in a task
    - spelling corrector, speech recognizer, MT system
  - Run the task, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - Compare accuracy for A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

Difficulty of extrinsic (in--vivo) evaluation of N-gram models



- Extrinsic evaluation
  - Time--consuming; can take days or weeks
- So
  - Sometimes use **intrinsic** evaluation: **perplexity**
  - Bad approximation
    - unless the test data looks **just** like the training data
    - So **generally only useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity



- The Shannon Game:
  - 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 _____

  - Unigrams are terrible at this game. (Why?)

mushrooms 0.1

pepperoni 0.1

anchovies 0.01

….

fried rice 0.0001

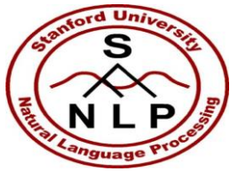
….

and 1e-100

- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

Perplexity

Average branching factor
On avg how many words can come next



The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

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

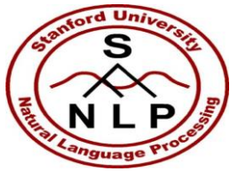
Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}}$$

For bigrams:

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

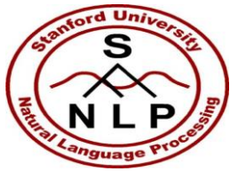
Minimizing perplexity is the same as maximizing probability



Perplexity as branching factor

- Let's suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}^{N}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$



The Shannon Game intuition for perplexity

- From Josh Goodman
- How hard is the task of recognizing digits ‘0,1,2,3,4,5,6,7,8,9’
  - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53
- Perplexity is weighted equivalent branching factor

The Shannon Game intuition for perplexity

A call-routing phone system gets 120K calls and has to recognize

- "Operator" (let's say this occurs 1 in 4 calls)
- "Sales" (1 in 4)
- "Technical Support" (1 in 4)
- 30,000 different names (each name occurring 1 time in the 120K calls)

To get the perplexity of this sequence of length 120K:

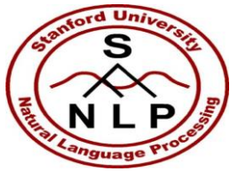
1) multiply 120K probabilities (90K of which are 1/4 and 30K of which are 1/120K)

2) take the inverse 120,000th root:

Perp = (¼ * ¼ * ¼* ¼ * ¼ * …. * 1/120K * 1/120K * ….)^(-1/120K)

Can be arithmetically simplified to just N = 4: operator (1/4), sales (1/4), tech support (1/4), and 30,000 names (1/120,000):

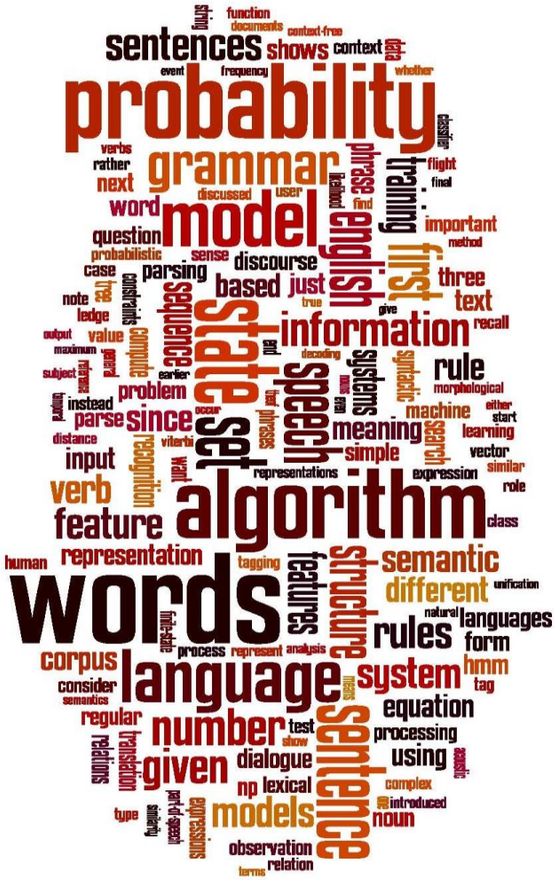
Perplexity= (¼ * ¼ * ¼ * 1/120K)^(-1/4) = 52.6



Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

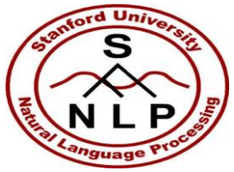
N--gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109



Language Modeling

Evaluation and Perplexity

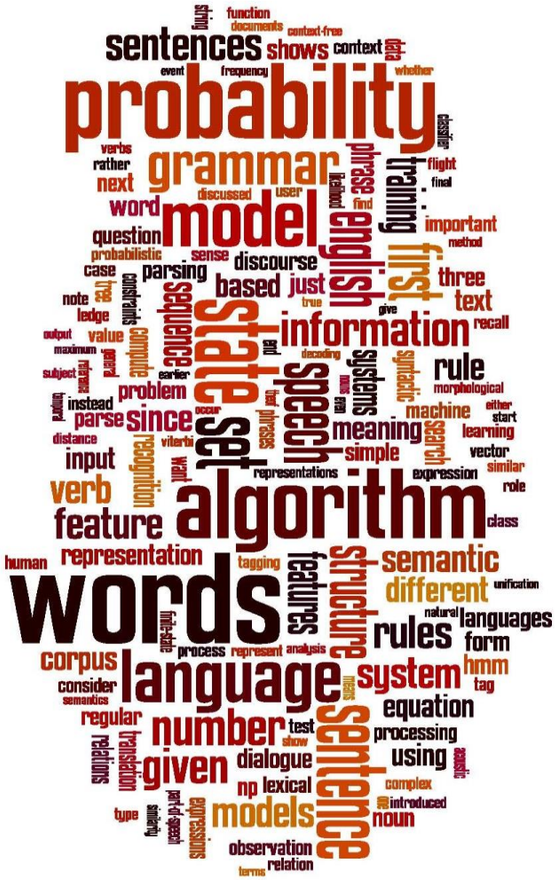
Example:



<s> I want to eat Chinese food </s>
<s> We ate Pakistani food </s>
<s>I ate apples </s>
<s>They ate Chinese food </s>

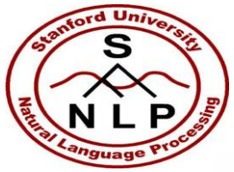
a) Calculate the probability of the following sentence. Include </s> in your counts just like any other token.
<s> I ate Chinese food</s>
Unigram
Bigram
Trigram

b) Calculate the perplexity of test sentence using bigram model.



Language Modeling

Generalization and zeros



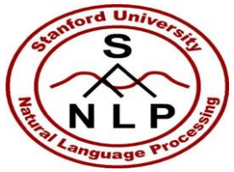
The Shannon Visualization Method

The n-gram model, like many statistical models, is dependent on the training corpus.

One implication of this is that the probabilities often encode specific facts about a given training corpus.

Another implication is that n-grams do a better and better job of modeling the training corpus as we increase the value of N.

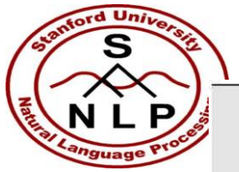
The Shannon Visualization Method



- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                           food  </s>
I want to eat Chinese food
```

Approximating Shakespeare



Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

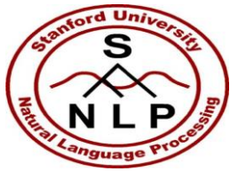
Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

Shakespeare as corpus



- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

The wall street journal is not shakespeare (no offense)

Unigram

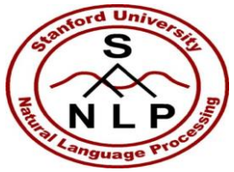
Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

Bigram

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

Trigram

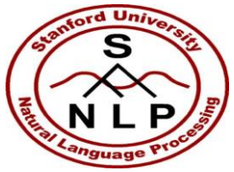
They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions



The perils of overfitting

- N–grams only work well for word prediction if the test corpus looks like the training corpus
  - In real life, it often doesn't
  - We need to train robust models that generalize!
  - One kind of generalization: Zeros!
    - Things that don't ever occur in the training set
      - But occur in the test set

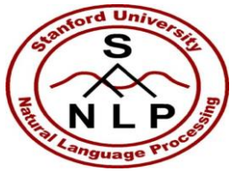
Zeros



- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P(“offer” | denied the) = 0

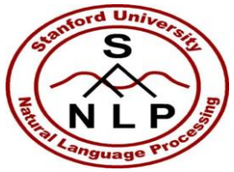
- Test set
  … denied the offer
  … denied the loan



Zero probability bigrams

- Bigrams with zero probability
  - mean that we will assign 0 probability to the test set!
- And hence we cannot compute perplexity (can't divide by 0)!

Example:



<s> I want to eat Chinese food </s>
<s> We ate Pakistani food </s>
<s>I ate apples </s>
<s>They ate Chinese food </s>

a) Calculate the probability of the following sentence. Include </s> in your counts just like any other token.
<s> I ate Chinese food</s>
Unigram
Bigram
Trigram

b) Calculate the perplexity of test sentence using bigram model.

Dan Jurafsky

Unknown words: Open versus closed vocabulary tasks

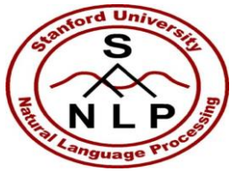
The previously we discussed the problem of words whose bigram probability is zero.

But what about words we simply have never seen before?

It depends on type of system

- Closed Vocabulary System
- Open Vocabulary System

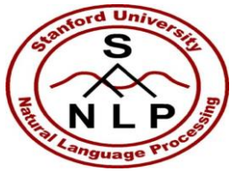
Closed vocabulary System



Sometimes we have a language task in which this can't happen because we know all the words that can occur.

In such a closed vocabulary system the test set can only contain words from this lexicon, and there will be no unknown words.

This is a reasonable assumption in some domains, such as speech recognition or machine translation, where we have a pronunciation dictionary or a phrase table that are fixed in advance, and so the language model can only use the words in that dictionary or phrase table.



Open Vocabulary System

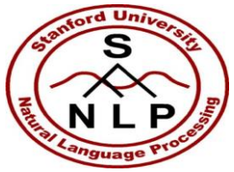
In other cases we have to deal with words we haven't seen before, which we'll call unknown words, or out of vocabulary (OOV) words.

The percentage of OOV words that appear in the test set is called the OOV rate.

An open vocabulary system is one in which we model these potential unknown words in the test set by adding a pseudo-word called <UNK>.

There are two common ways to train the probabilities of the unknown word model <UNK>.

Open Vocabulary System: First Approach



The first one is to turn the problem back into a closed vocabulary one by choosing a fixed vocabulary in advance:

1. Choose a vocabulary (word list) that is fixed in advance.

2. Convert in the training set any word that is not in this set (any OOV word) to the unknown word token <UNK> in a text normalization step.

3. Estimate the probabilities for <UNK> from its counts just like any other regular word in the training set.

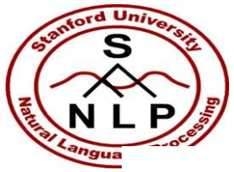
Open Vocabulary System: Second Approach

The second alternative, in situations where we don't have a prior vocabulary in advance, is to create such a vocabulary implicitly, replacing words in the training data by <UNK> based on their frequency.

For example we can replace by <UNK> all words that occur fewer than n times in the training set, where n is some small number, or equivalently select a vocabulary size V in advance (say 50,000) and choose the top V words by frequency and replace the rest by UNK.

In either case we then proceed to train the language model as before, treating <UNK> like a regular word.

Unknown Words

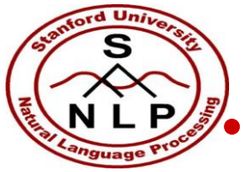


The exact choice of <UNK> model does have an effect on metrics like perplexity.

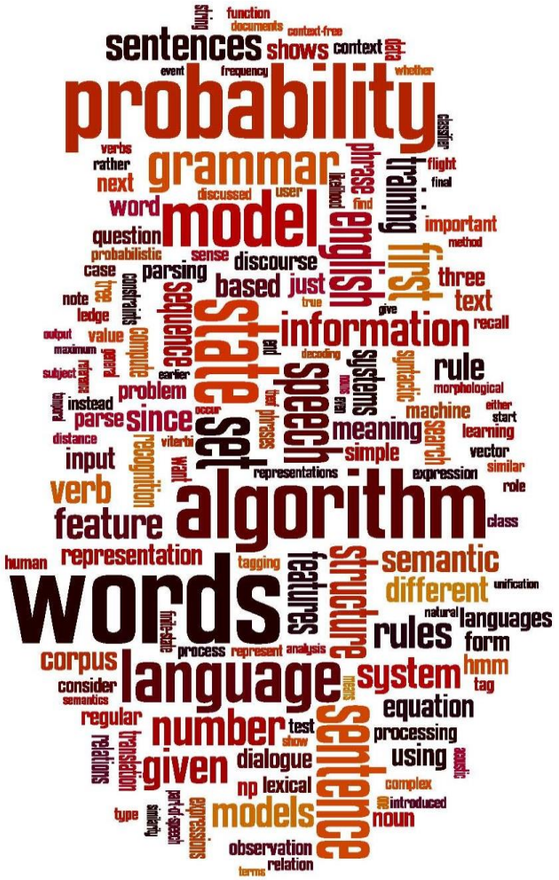
A language model can achieve low perplexity by choosing a small vocabulary and assigning the unknown word a high probability.

For this reason, perplexities should only be compared across language models with the same vocabularies

Unknown words: Open versus closed vocabulary tasks



- If we know all the words in advanced
  - Vocabulary V is fixed
  - Closed vocabulary task
- Often we don't know this
  - **Out Of Vocabulary** = OOV words
  - Open vocabulary task
- Instead: create an unknown word token <UNK>
  - Training of <UNK> probabilities
    - Create a fixed lexicon L of size V
    - At text normalization phase, any training word not in L changed to <UNK>
    - Now we train its probabilities like a normal word
  - At decoding time
    - If text input: Use UNK probabilities for any word not in training



Language Modeling

Generalization and zeros