COMP 3225

Natural Language Processing

5. N-Grams

Les Carr

lac@soton.ac.uk

University of Southampton

In This Lecture You Will …

- Learn how to use language models
- Learn about the predictive power of word sequences.
- Learn how to train ngram models, even in the face of sparse data

The material in this lecture is based on chapter 3 from Jurafsky and Martin, Speech and Language Processing, 3rd edition (online)

Language Models: n-gram

- Language models assign probabilities to sequences of words
- The simplest of these is the n-gram, which assigns probabilities to sentences and sequences of words.
  - 2-gram (bigram) is a two-word sequence of words like “language model”, “models assign”, or ”assign probabilities”,
  - 3-gram (trigram) is a three-word sequence of words like “models assign probabilities”, or “sequences of words”.
  - 1-gram (unigram) are independent words, unconnected to each other, i.e. vocabulary frequency tables
- n-gram models can be used to
  - estimate the probability of the last word of an n-gram given the previous words,
  - assign probabilities to entire sequences.
- In later lectures you will discover more sophisticated language models like RNN LMs.

Importance of Language Models

- Probabilities are essential in any task in which we have to identify words in noisy, ambiguous input, or judge fluency
  - **speech recognition**
    - A language model contains information that *back soonish* is a more probable sequence than *bassoon dish*
  - **spelling correction** or **grammatical error correction**
    - in writing like *Their are two exams*, or *Everything has improve*
    - The phrase *There are* will be much more probable than *Their are*, and *has improved* than *has improve*
  - **machine translation**
    - As part of the translation process we have a set of potential rough English translations:
      - he introduced reporters to the main contents of the statement
      - he briefed to reporters the main contents of the statement
      - he briefed reporters on the main contents of the statement
    - A probabilistic model of word sequences could suggest that *briefed reporters on* is a more probable English phrase (and hence more **fluent**) than *briefed to reporters* or *introduced reporters to*
  - **augmentative and alternative communication systems**
    - People who are physically unable to speak or sign but can instead use eye gaze or other specific movements to select words from a menu to be spoken by the system.
      - word prediction can be used to suggest likely words for the menu.

Unigram (1-gram)

- Vocabulary list
- Assumes appearances of words are independent
  - $P(w_1,w_2,w_3,w_4) = P(w_1) * P(w_2) * P(w_3) * P(w_4)$
- Assumes that the previous words have *no influence* on the next word
  - Given the sequence "the cat sat on the", what is the most probable next word?
  - The cat sat on the the
    - The most probable next word is just the most probable word

word	P(word)
the	4.85%
and	2.94%
to	2.50%
he	2.20%
a	1.94%
was	1.93%
of	1.78%
it	1.72%
in	1.41%
that	1.31%
she	1.25%
they	1.22%
had	1.06%
you	1.03%
but	1.00%
his	0.98%
her	0.97%
i	0.97%
peter	0.83%
not	0.82%
on	0.81%
wendy	0.74%

English language word frequency table as generated from Peter Pan

The The

Matt Johnson

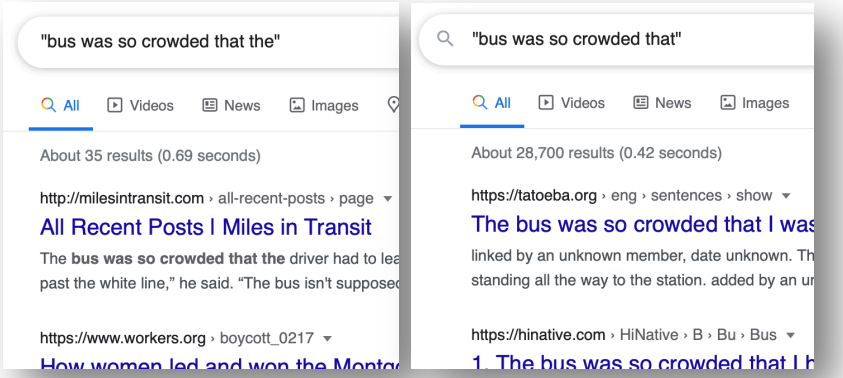
An English musical and multimedia group with singer/songwriter Matt Johnson being the only constant band member of the “The The”. The “The The” has no permanent group line-up, and Johnson has collaborated with a wide range of musicians, changing personnel from project to project. The The The began recording in 1979/1980 and released their first single “Controversial Subject” in 1980.

N-grams

- An n-gram model informs us of the probability of the *next word* in the text, given the *previous n-1 words* in the text.
  - P(w | h) the probability of word *w* given the previous history *h*, where *h* is is sequence of words
  - Out of the times that *h* occurred, how may times was it followed by *w*

$$P(\text{the} \mid \text{bus was so crowded that}) = \frac{Count(\text{bus was so crowded that the})}{Count(\text{bus was so crowded that})}$$

$$= \frac{35}{28700} = 0.12\% \textit{ (Google)}$$



In General

- $P(w_1) = P(w_1)$
- $P(w_1,w_2) = P(w_2 \mid w_1) * P(w_1)$
- $P(w_1,w_2,w_3) = P(w_3 \mid w_1,w_2) * P(w_1,w_2)$
- $P(w_1,w_2,w_3,w_4) = P(w_4 \mid w_1,w_2,w_3) * P(w_1, w_2,w_3)$

  . . .
- $P(w_1, \ldots, w_n) = P(w_n \mid w_1,\ldots,w_{n-1}) * P(w_1, \ldots w_{n-1})$

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

Equation 3.4

This is impossible to compute in general for long texts (n=50000). Instead, assume a locality and restrict n to 2, 3, 4 or 5.

Bigrams n=2

- The bigram model approximates the probability of the next word in a document by using only the conditional probability of the preceding word
  - $P(w_1, \ldots, w_n) = P(w_n \mid w_1,\ldots,w_{n-1}) * P(w_1, \ldots w_{n-1})$
    $\approx P(w_n \mid w_{n-1})$

N-grams and Markov models

- The assumption that the probability of a word depends only on the previous word is called a **Markov assumption**.
  - Markov models are probabilistic models that assume we can predict the probability of some future event from the current state
- We can generalize the bigram (which looks one word into the past) to the trigram (which looks two words into the past) and thus to the n-gram (which looks n−1 words into the past).
- The **N**-gram approximation to the conditional probability of the next word in a sequence is given by
  - $P(w_n \mid w_{n-1}) \approx P(w_{n-\mathbf{N}+1 : n-1} \mid w_{n-1})$
    - where N=2 for bigrams, 3 for trigrams etc

- Given the bigram assumption for the *probability of an individual word*, we can compute the probability of a *complete word sequence*

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

Equation 3.9

This model for the probability of a word sequence is a language model. To make it operative, we need to assign its parameters - ie the individual bigram probabilities…

Maximum Likelihood Estimation (MLE)

- The process of choosing the right set of bigram parameters to make our model correctly predict (maximise the likelihood of) the nth word in the text is called *maximum likelihood estimation*.

- the MLE estimate for the parameters of an n-gram model are obtained by
  - observing the n-gram counts from a representative corpus
  - normalizing them (dividing by a total count) to lie between 0 and 1

Bigram Parameter Estimation

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

Equation 3.11

N-gram Parameter Estimation

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

Equation 3.12

This is the process we saw on slide 7 "the bus was so crowded that"

Example

Bigram frequencies

	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

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

Unigram frequencies

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

Bigram probabilities

	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

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Other misc probabilities

$P(\texttt{i}|\texttt{<s>}) = 0.25$ $P(\texttt{english}|\texttt{want}) = 0.0011$

$P(\texttt{food}|\texttt{english}) = 0.5$ $P(\texttt{</s>}|\texttt{food}) = 0.68$

Example taken from Berkeley Restaurant project, described on p33.

P(I, want) = P(want | I)
= count(I, want) / count (I)
= 827 / 2533 = 0.33

P(<s> i want english food </s>)
= P(i|<s>)P(want|i)P(english|want)
P(food|english)P(</s>|food)
= .25 × .33 × .0011 × 0.5 × 0.68
= .000031 *

P(<s> i want *chinese* food </s>)
= P(i|<s>)P(want|i)P(*chinese* |want)
P(food| *chinese*)P(</s>|food)
= .25 × .33 × *.0065* × *0.52* × 0.68
= .000190 *

Practical Notes

- it's more common to use trigram models, which condition on the previous two words rather than the previous word, or 4-gram or even 5-gram models, when there is sufficient training data.
  - Note that for these larger n-grams, we'll need to assume extra contexts to the left and right of the sentence end.
  - e.g. first trigram in sentence is P(I|<s><s>

- We always represent and compute language model probabilities in log format as log probabilities to avoid numerical underflow.
  - Adding in log space is equivalent to multiplying in linear space, so we combine log probabilities by adding them.
  - convert back into probabilities if we need to report them by taking the exp of the log (prob)
  - $p_1 \times p_2 \times p_3 \times p_4 = \exp^{log\ p1 + log\ p2 + log\ p3 + log\ p4}$

Break

- Explore n-grams by checking your phone's predicative capabilities, starting with these seeds and repeatedly choosing from the most likely next words
  - I went to university because
  - I will be famous for
  - My favourite food

N-gram problems

Shakespearean ngrams

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
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.

Figure 3.3 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

Journalistic ngrams

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

Figure 3.4 Three sentences randomly generated from three n-gram models computed from 40 million words of the *Wall Street Journal*, lower-casing all characters and treating punctuation as words. Output was then hand-corrected for capitalization to improve readability.

- Longer n-grams encode more 'sense'
  - But there is a training problem
  - Even if |V| is large, only a tiny minority of possible n-grams exist in any corpus
    - $P(w_k \mid w_{k-1}) = 0$ for most sequences (w_{k-1}, w_k)
    - Matrix is sparse
  - More severe problem as n increases

Smoothing

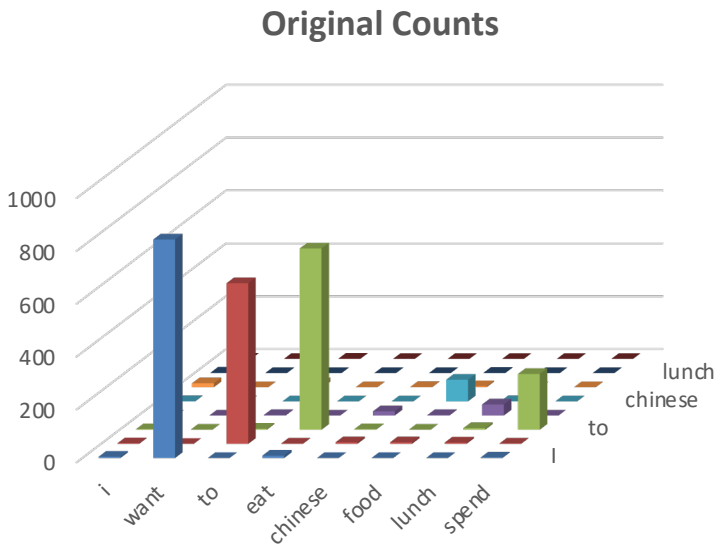
- To keep a language model from assigning zero probability to unseen n-grams, a bit of probability is taken from some more frequent n-grams and given to the unseen ones.
- This modification is called *smoothing* or *discounting*.
  - In **Laplace smoothing**, simply +1 to all the bigram counts, before we normalize them into probabilities
    - All the counts that used to be zero will now have a count of 1, the counts of 1 will be 2, and so on.
    - Also known as add-one smoothing
  - useful for tasks like text classification

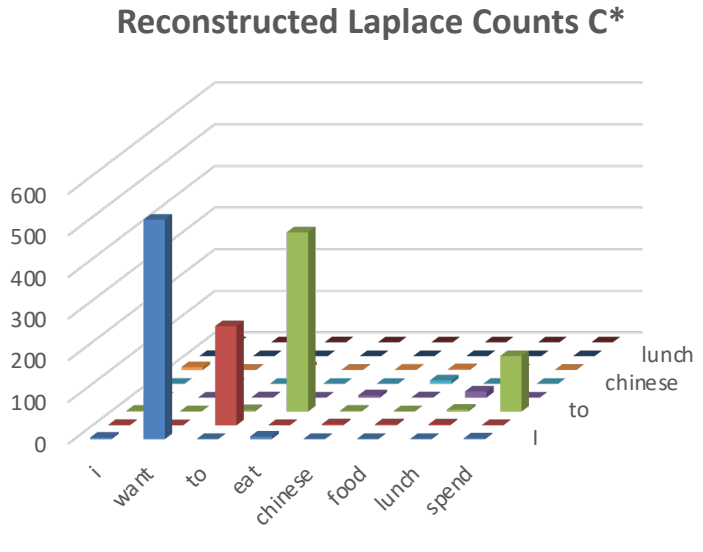
Laplace smoothing

- Apply Laplace smoothing to unigrams.
  - the unsmoothed MLE of word w_i is its count c_i normalized by N, the total number of tokens
    - $P(w_i) = c_i / N$
  - Since there are V words in the vocabulary and each one was incremented, the denominator is also adjusted to take into account the extra V observations
    - $LP(w_i) = (c_i + 1) / (N + V)$
  - Instead of changing both the numerator and denominator, it is convenient to describe how a smoothing algorithm affects the numerator, by defining an adjusted count c^* which is easier to compare directly with the MLE counts
    - $c^*_i = (c_i + 1) N / (N + V)$
  - Normalising by N yields the same expression as $LP(w_i)$ above
  - Also consider the discount rate $d_c = c^* / c$

Problems with Laplace Smoothing

- The “extra V observations” which have to be added to all the normalisations are problematic





C(*want to*) changed from 609 to 238; *P*(*to* | *want*) decreases from .66 to .26
the discount ratio for the bigram *want to* is .39, and for *Chinese food* is .10

Too much probability mass has been moved to the zero-occurrence cases

Add-k Smoothing

- move a bit less of the probability mass from the seen to the unseen events.
- add a fractional count *k* to each event.
  - $P^*_{\text{Add-k}}(w_n \mid w_{n-1}) = \frac{C(w_{n-1}\, w_n) + k}{C(w_{n-1}) + kV}$
  - requires a choice of *k*, perhaps by optimizing on a devset
- Although add-k is useful for some tasks (including text classification), it turns out that it still doesn't work well for language modelling, generating counts with poor variances and often inappropriate discounts

Backoff & Interpolation

- If data is sparse about the appearance of higher-order n-grams, we can fall back to information about lower order n-grams
- to compute $P(w_n \mid w_{n-2}w_{n-1})$ without counts of the trigram $w_{n-2}\ w_{n-1}\ w_n$
  - estimate its probability by using the bigram probability $P(w_n \mid w_{n-1})$
  - if there are no counts of the bigram, estimate using the unigram $P(w_n)$.

- In the absence of detailed information, using less context can be a good strategy. Allows generalization to more for contexts that the model hasn't been trained on about.

- There are two ways to use this n-gram "hierarchy".
  - **Backoff**: only use lower-order n-gram if we have zero evidence for a higher-order n-gram
  - **Interpolation**: always mix the probability estimates from all the n-gram estimators, weighing and combining the trigram, bigram, and unigram counts.

Interpolation

- In simple linear interpolation, we combine different order n-grams by linearly interpolating all the models.
- estimate the trigram probability $P(w_n \mid w_{n-2}w_{n-1})$ by mixing together the unigram, bigram, and trigram probabilities, each weighted by a λ
  - such that the λs sum to 1

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

Equation 3.26

$$\sum_i \lambda_i = 1$$

- Possible to make the λs dependent on the accuracy of the bigram statistics, themselves learned from a held-out corpus.

Problem of Discounting (1)

- The language model is trained on a specific set of texts which cannot include all possible linguistic variations
- Adjustments have to be made to the language model to allow it to accommodate unexpected word adjacencies
  - Discounting of existing probabilities in an equitable fashion
  - Substitution of 0-value probabilities of unseen word sequences with reasonable approximations either by adding a standard proportion
    - Laplace, add-one, add-k
  - or by calculation from lower-order n-gram information. See sections 3.4 –3.6 describe increasing complex ways to calculate appropriate backoff discounting:
    - Katz backoff + Good-Turing estimate of P* and α

$$P_{\text{BO}}(w_n|w_{n-N+1:n-1}) = \begin{cases} P^*(w_n|w_{n-N+1:n-1}), & \text{if } C(w_{n-N+1:n}) > 0 \\ \alpha(w_{n-N+1:n-1})P_{\text{BO}}(w_n|w_{n-N+2:n-1}), & \text{otherwise.} \end{cases}$$

Equation 3.29

Problem of Discounting (2)

- **Interpolated Kneser-Ney** smoothing makes use of two approaches
  - an *absolute discounting* factor of 0.75
  - the probability of a unigram being a *novel continuation* of another word in a bigram

Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Fig 3.8

The size of the set of words v, that w appears after, *ie* where the count of (v,w) is greater than 0.

$$P_{\text{CONTINUATION}}(w) = \frac{|\{v : C(vw) > 0\}|}{|\{(u', w') : C(u'w') > 0\}|}$$

Equation 3.32

The size of the set of adjacent word pairs (u', w') *ie* where the count of (u',w') is greater than 0.

- A frequent word 'Britain' with only a single context 'Great Britain' will have a low continuation probability.
- The **modified Kneser-Ney** algorithm uses three separate discounting factors for n-grams of size 1, 2 and 3.

Problem of Discounting (3)

- **Stupid backoff** is best for very large language models, but abandons need for adjusting the normalization factor to maintain a strict probability distribution $\sum p_i = 1$

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if count}(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

Equation 3.40

Undiscounted probability of the n-gram, if data exists **or…**

…scale the lower-order estimate by a constant factor **λ** *e.g.* 0.4

- See https://www.aclweb.org/anthology/D07-1090.pdf
  - We introduce a new smoothing method, dubbed Stupid Backoff, that is inexpensive to train on large data sets and approaches the quality of Kneser-Ney Smoothing as the amount of training data increases
  - Experimental results were presented showing the effect of increasing the amount of training data to up to 2 trillion tokens, resulting in a 5-gram language model size of up to 300 billion n-grams.
  - State-of-the-art techniques like Kneser-Ney Smoothing or Katz Backoff require additional, more expensive steps. At runtime, the client needs to additionally request up to 4 backoff factors for each 5-gram requested from the servers, thereby multiplying network traffic…
  - The name originated at a time when we thought that such a simple scheme cannot possibly be good. Our view of the scheme changed, but the name stuck.

N-gram variants for data sparsity

- Class-based N-gram (aka Brown clustering)
  - probabilities of words are based on the classes of previous words
  - in a flight reservation system that needs to estimate the likelihood of the unseen bigram *to Shanghai*
    - First, cluster "Shanghai" with other city names
    - Then, make estimate based on the likelihood of matching phrases from that cluster e.g. "to London", "to Beijing" and "to Denver"
- Skip-grams
  - N-grams in which the words do not need to be adjacent in the text, but can be separated by a distance k
  - E.g. in the text *the rain in Spain falls mainly on the plain* the set of 1-skip-2-grams consists of
    - all the bigrams (2-grams) *the rain, rain in, in Spain, Spain falls, falls mainly, mainly on, on the, the plain*
    - plus the non-adjacent words *the in*, *rain Spain*, *in falls*, *Spain mainly*, *falls on*, *mainly the*, and *on plain*
  - Useful for capturing word contexts (locality) with less sparsity than strict n-grams. Useful for word2vec in later lecture.

Google n-gram Dataset

- SemEval-2013 https://www.aclweb.org/anthology/S13-1035/
  - We created a dataset of syntactic-ngrams (counted dependency-tree fragments) based on a corpus of 3.5 million English books. The dataset includes over 10 billion distinct items covering a wide range of syntactic configurations. It also includes temporal information, facilitating new kinds of research into lexical semantics over time. This paper describes the dataset, the syntactic representation, and the kinds of information provided.
  - The dataset is based on the English Google Books corpus. The corpus consists of the text of 3,473,595 English books which were published between 1520 and 2008, with the majority of the content published after 1800.
- Dataset http://commondatastorage.googleapis.com/books/syntactic-ngrams/

End of Lecture Questions

- Panopto Quiz - 1 minute brainstorm for interactive questions

Please spend **1 minute** using Panopto quiz to wrie down two or three questions that you would like to have answered at the next interactive session.

Do it **right now** while its fresh.

Take a screen shot of your questions and **bring them with you** at the interactive session so you have something to ask.