Language Modeling

A model that assigns probabilities to *sequences* of words

$$p(w_N, \ldots, w_1)$$

Arguably the most important tool in NLP.

GPT is just a massive language model.

we | both saw the bear
we both | saw the bear
we both saw| the bear
we both saw the | bear

Conditional language model

$$p(w_N, ..., w_1) = \prod_i p(w_i | w_{i-1}, ..., w_1)$$

in order to compute the probability we both saw
we can find the probability of ‘we’, the probability of ‘we both’, the probability of ‘we both saw’

actually, words are closer are more correlated
so we can relax less and look back 3/4/10 words in the past
also not infinite, but still tractable

these assumption is wrong
if we have ‘Both’ , it is possible to have ‘both are’ ‘both of’ ‘both people’
but if we have ‘We both’, than only the ‘both are’ makes sense

assumption: only depends on one before

Markov property

$$p(w_i | w_{i-1}, \ldots, w_1) = p(w_i | w_{i-1})$$

the conditional probability of a word does not depend on everything that came before it
but only stuff that goes back a certain distance

the conditional probability of Wi given everything that came before
is actually equal to the probability of Wi given one word before it

Markov model

Model stochastic systems: the probability of future states depend only on the current state

A common example models the weather:

$$p(x_i = rain|x_{i-1} = rain) = 0.6$$
$$p(x_i = rain|x_{i-1} = sun) = 0.2$$
$$p(x_i = sun|x_{i-1} = rain) = 0.4$$
$$p(x_i = sun|x_{i-1} = sun) = 0.8$$

We've been saying word before,
now we can imagine doing this with letters too

Markov models in text

Example: letters

$p(c_i = \text{'q'} | c_{i-1} = \text{'q'})$ is low q following q is low, but u following q is high

$p(c_i = \text{'u'} | c_{i-1} = \text{'q'})$ is high

$0 < p(c_i = \text{'n'} | c_{i-1} = \text{'m'}) < p(c_i = \text{'e'} | c_{i-1} = \text{'m'})$

...

n-grams

the noisy channel model by itself is insufficient to capture information in context
but n-grams model or a broader language in general, does give us a tool to look at the context
in which a word occurs

A sequence of n tokens (words, characters, ...).

'Hello world' includes nine 3-grams:

'Hel', 'ell', 'llo', 'lo ', 'o w', ' wo', 'wor', 'orl', 'rld'

3-gram model:
‘Hel’ —> the probability of ‘l’ given ‘He’

n-gram model

$$p(w_i | w_{i-1}, ..., w_1) = p(w_i | w_{i-1}, ..., w_{i-n+1})$$

note: an n-gram model looks n-1 words into the past (a trigram model looks two words into the past, etc.)

Markov text generation (auto-complete)

A Markov model (or n-gram model, more generally) can be used generatively.

Given the beginning of a document, e.g. "He went...", we can develop a probabilistic model of what follows:

$$p(w_3|w_2 = \text{‘went’}, w_1 = \text{‘He’}) = p(w_3|\text{‘He’}, \text{‘went’})$$

We can identify the most likely continuation using maximum likelihood estimation (MLE):

$$\hat{w}_3 = \arg\max_{w_3} p(w_3|\text{‘He’}, \text{‘went’})$$

Representing/generating an n-gram model

- based on data!

the more common a particular n-gram is ,
the hight probability we are assigned to that particular transition

- from a (large) corpus of text, extract n-grams
  - each is evidence that the nth word follows the first n-1

count the number of occurrences where i
happened and normalize the distribu

  - these are samples from the distribution!
- to infer $p(w_n | w_{n-1}, \ldots, w_1)$ by fetching all n-grams with prefix $[w_1, \ldots, w_{n-1}]$

if we have a sufficient large data sets and what comes
out is a languae model that maybe reflects the
language used in this corpus of data

Example n-gram model

`l06_character_bigram_model.py`

we are going to trade off between the expressiveness of the individual n-rams and the amount of data
because when n in n-grams become really large, the autocomplete system only know 1/2 ways of completing that end — less useful
we have information, content, but we don't have enough training data
e.g.: Do you have a monkey? is a normal thing in english, but it's very unlikely to have occurred in out corpus
so, when you're asking the questions what follows, it says, I have no idea because I've never seen that prefix before

Problems

- unknown words
- rare n-grams

Unknown words

token issue — token doesn't exist

english evolve with use, there's the words that we use this year/this decade that just didn't exist before

a.k.a. out-of-vocabulary (OOV) words

this is going to be much more common when the tokens are more complicated

You will encounter words that did not occur in the training data. What to do?

· unknown tokens are likely to occur in similar scenarios For instance named entities. Some person/some place shows up and has a name we haven't seen before

Use a special <UNK> token to represent all of these, treating it just like any other word.

· But we are not going to see a conjunction falling in unknown word bucket because there's pretty much a fixed of conjunctions in Englishxt

Train by assigning all rare words $\boxed{\text{(e.g. frequency < X)}}$ to <UNK>, or all words not in a predefined vocabulary.

if we take all the words that only occur once or twice in our corpus and just lump them in here, that allows us to learn something about the unknown word context.

Then when we encounter something at the application time that we haven't seen before, we also assign this unknown token to it

e.g.: transition issue—- doesn’t exist(our training data is limited)
in our corpus, there is no q -> i, however, it’s a english word
if we want to use that language model in some contexts beyond training data,
we really don’t want to assign a probability zero to that transition.

Rare n-grams

We need all to be non-zero
We haven’t seen it doesn’t mena it doesn’t exist

What is the probability of "if it is in"? Non-zero.

the model can recover from very small, but it can’t recover from zero

What if it does not occur in the training set? Options:

- smoothing
- backoff
- interpolation

This is happening after the transition:
the unknown token is in the matrix, and try to figure out what's the transition from Q to unknown/ from V to unknown
if V -> unknown is 270000 times and unknown -> unknown is 0 times. if we don't like it, we add 1 to everything in the matrix

Smoothing

some are zero some are not, just add 1 to all of those and normalize it (1 = 1 observation)
ideas here: training is useful but is going to miss stuff

- add-1 (Laplace) smoothing
- add-k smoothing

if you feel like one is too much, that's sort of overstating the likelihood that this transition happened, you can do add 0.5/0.1

- Kneser-Ney smoothing

other algorithm

Backoff

如果 it is in没有 就看 is in 在不就看in

if you set up your unknown token correctly, you should have be able to assign a frequency to every token

If there is no data for $p(w_3|w_2, w_1)$, use $p(w_3|w_2)$, or $p(w_3)$. i.e. Instead of trigram, use bigram or unigram model.

Interpolation

because the longer n-gram, the probability that gets assigned to them is less informative

combine the probability of ‘it is in’ ‘is in’ ‘in’

Combine trigram, bigram, and unigram models together, e.g. by weighted average.

Computational considerations

For a vocabulary of size `N` and corpus of size `M`, how much space does a bigram (n=2) model take up?

… a trigram (n=3) model?

Suggestion: compute chunks of the n-gram model on demand.

References

- https://web.stanford.edu/~jurafsky/slp3/3.pdf