language modeling (part 2)

Keith Trnka

cisc 882   2009-10-27

language modeling

- some function P(word | history)
  - gives a probability distribution for words
  - how it represents the history makes it an ngram model or cache model or topic model, etc etcs
- common practice is trigram backoff model and some smoothing method

evaluation

- task-neutral
  - perplexity
  - geometric mean probability
- task-specific
  - keystroke savings
  - word error rate

keystroke savings

- percentage reduction in the number of key presses
- are uppercase letters one or two keystrokes?
  - we assume one - AAC users probably wouldn't capitalize anyway

$$KS = \frac{chars - keystrokes}{chars} \times 100\%$$

keystroke savings

- assume a given number of predictions (window size W)
- simulate the typing of the text and select the completion/prediction of the desired word as early as possible
  - actual users only achieve maybe 94% of this in a good system, maybe 80% of this in a bad system

cross-validation

- split a corpus into *k* sets
- iterate through the sets:
  - for a given set *i*, train the model on all other sets
  - test on set *i*
  - combine the results of testing on all *k* sets afterwards
    - what's the subtle problem here?

cross-validation

- combining the results for all sets (keystroke savings)
  - option A: average the keystroke savings for all sets
  - option B:
    - sum the number of keystrokes when using prediction at window size W
    - sum the number of characters
    - divide one by the other

cross-validation

- why would they be different?
  - sets will have somewhat different numbers of words
- why do I care though?
  - the difference between the two methods may be greater than the difference between your work and someone else's

balancing sets

- imagine a text message corpus
  (very small documents)
- trigram/backoff, ambiguous keyboard task (T9)
- how do we split them up into sets?
  - randomly
  - what if they're clustered by similarity?
  - what if all sets look the same?

balancing sets

- clustered by similarity
  - very high chance in testing that you didn’t see that word in any of your training sets
- all sets very similar
  - very low chance of OOVs
- randomly
  - can’t be sure of anything

balancing sets

- clustered by similarity
  - minimal performance w.r.t. set balance
- all sets very similar
  - maximal performance w.r.t. set balance
- randomly
  - who knows!

balancing sets

- which you choose depends on task
  - topic modeling - set balance is important
  - style modeling - probably better with half-imbalanced sets
  - cache modeling - will be more beneficial with imbalanced sets

how to interpret results

	trained/tested using cross-validation			
Testing corpus	keystroke savings			
AAC Email	48.92%			
Callhome	43.76%			
Charlotte	48.30%			
SBCSAE	42.30%			
Micase	49.00%			
Switchboard	60.35%			
Slate	53.13%			

how to interpret results

	trained/tested using cross-validation			
Testing corpus	keystroke savings	size		
AAC Email	48.92%	27,710		
Callhome	43.76%	48,407		
Charlotte	48.30%	187,587		
SBCSAE	42.30%	237,191		
Micase	49.00%	545,411		
Switchboard	60.35%	2,883,774		
Slate	53.13%	3,902,380		

how to interpret results

	trained/tested using cross-validation			
Testing corpus	keystroke savings	size	testing OOVs	
AAC Email	48.92%	27,710	8.48%	
Callhome	43.76%	48,407	6.86%	
Charlotte	48.30%	187,587	4.49%	
SBCSAE	42.30%	237,191	5.76%	
Micase	49.00%	545,411	4.40%	
Switchboard	60.35%	2,883,774	0.52%	
Slate	53.13%	3,902,380	1.96%	

how to interpret results

	trained/tested using cross-validation			
Testing corpus	keystroke savings	size	testing OOVs	named entities
AAC Email	48.92%	27,710	8.48%	8.92%
Callhome	43.76%	48,407	6.86%	8.23%
Charlotte	48.30%	187,587	4.49%	6.59%
SBCSAE	42.30%	237,191	5.76%	5.67%
Micase	49.00%	545,411	4.40%	3.12%
Switchboard	60.35%	2,883,774	0.52%	2.10%
Slate	53.13%	3,902,380	1.96%	12.03%

register-varied evaluation

- evaluate on many different kinds of text separately
  - sometimes a technique mostly helps for one kind of text
- alternative: diverse corpora (BNC, ANC)
  - problem: they're 90-95% written anyway

domain-varied evaluation

- ngrams are ***very*** sensitive to the difference between training and testing data
  - make a clear distinction between in-domain (trained on same type) and out-of-domain (trained on something else)
  - but it's not so clear...
    - broadcast news transcription
    - newspaper
    - email
    - blogs

domain-varied evaluation

- corpus normalization (and nightmares)
  - do all your corpora capitalize the first word in a sentence?
  - do they all have punctuation?
  - (spoken especially) “don’t know” vs. “dunno”
  - spoken may have speech repairs, written may have typos

how to test

- option A: train on something, test on a bunch of different things
- option B: test on something, train on a bunch of different things
- the end difference is ***perspective***

perspective

- compare the two
  - trained on corpus A, tested on corpus A
  - trained on corpus B, tested on corpus A
- compare the two
  - trained on corpus A, tested on corpus A
  - trained on corpus A, tested on corpus B

perspective

- testing data stays the same
  - intrinsic difficulties of the testing corpus stay the same (apples-to-apples)
    - keystroke savings has a maximum for any given data (at least one keystroke per word)
- training data stays the same
  - intrinsic quality of the language model stays the same

perspective

- in my work, I prefer to keep testing data the same
- upcoming: example of doing cross-validation with register/domain variations

In-domain evaluation

from my thesis work

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

Out-of-domain evaluation

from my thesis work

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

	Set 1	Set 2	Set 3	Set 4	Set 5
Corpus A					
Corpus B					
Corpus C					
Corpus D					

green = training, red = testing

Example evaluation

	Training text		
Testing corpus	In-domain	Out-of-domain	Mixed-domain
AAC Email	48.92%	47.89%	**52.18%**
Callhome	43.76%	52.95%	**53.14%**
Charlotte	48.30%	52.44%	**53.50%**
SBCSAE	42.30%	46.97%	**47.78%**
Micase	49.00%	49.62%	**51.46%**
Switchboard	**60.35%**	53.88%	59.80%
Slate	**53.13%**	40.73%	53.05%

statistical significance

statistical significance

- decide on the smallest unit where the performance of different units is independent
  - words: no!
  - sentences: yes for simple ngrams, no for cache models, no for topic/style models, etc.
  - document: yes
- you'll be doing mean/std deviation over this set

statistical significance

- (most NLP applications) if you can formulate it as "before and after" or "baseline+improvement"
  - create a distribution of differences (the pairing part)
    - e.g., +5 on doc 1, -1 on doc 2, +3 on doc 3
  - null hypothesis: the true mean of the difference distribution is zero
    - try to say that the chance of this is under 0.05 or so
    - use t-test (in this case called *paired* t-test)

statistical significance

- keystroke savings
    - the number of keys used in typing is an invariant
- options for per-document differences
    - difference in keystroke savings
    - difference number of keystrokes used with prediction
    - normalized difference of keystrokes

pros and cons

- difference in keystroke savings
  - pro: simple, natural
  - con: standard deviation will be high with some "easy" documents and some "hard" documents
- normalized difference of keystrokes
  - pro: can be a stronger statistical test
  - con: more work

Part of speech

Markov model taggers

- terminology:
  - markov model tagger: looks like
    P(w | t) * P(t | history)
  - hidden markov model tagger: looks the same, but it's trained on UNLABELED training data
  - P(w | tag) = emission probability
  - P(tag | stuff) = transition probability

Dealing with tag sets

- tag sets are dependent on tokenization...
  - Treebank tagging splits off ‘s, for example, does something funny with “don’t”
  - even if the common case makes sense 10% of your data might be weird

Markov model tagging

- How to actually tag something with it?
  - basic: for each word, pick the tag that maximizes P(w | tag) * P(tag | tag_{-1}, tag_{-2}, etc)
    - susceptible to garden-pathing (finding a local optimum for the sentence)
  - Viterbi: special algorithm to find the optimal sequence of tags for a sentence

Viterbi method

- think about a lattice that contains all possible tags for each word and transitions between them
- you’re searching for an optimal path
- I’m gonna draw on the board cause it’s easier

Viterbi method

- can be slow if you have a large tagset
- modify it to only consider the top N candidates from the previous word (beam or n-best search)

POS + word prediction

- as you process each word, the Viterbi method is updating
- you never really commit to one tagging of the history (until the sentence ends)
    - have a set of taggings of the history, with weights
    - process each possible tagging of the history and weight the contribution
    - process each possible tag of the next word, weighted by the transition probability

Word prediction

- (optimization) compile all possible Viterbi histories with all possible tags of the word into one distribution
- (optimization) pre-sort, pre-filter emission probability lists
- (optimization) sort the possible tags, stop processing once you have “enough”
- and so on...

Personal experiences

- pos ngrams are less susceptible to data sparseness than word ngrams
  - but perform no better on most corpora anyway
- getting it to run efficiently takes effort
- doing Viterbi as a beam search speeds it up A LOT for a little performance cost
  - don't do k-best, do a threshold on the probabilities

Personal experiences

- unknown words - build a decision tree to tag words by their suffixes (TreeTagger)
  - separate tree for /^[a-z]/ from everything else worked better than other approaches
- can tag a dictionary using a suffix tagger
  - then use it along with your transition probabilities to predict them in the right contexts (noticeable improvement in keystroke savings)