# FNLP 2016 Past Paper

## PART A

1. Describe or illustrate the essence of Zipf's Law as it pertains to word frequencies. What are the implications of Zipf's law for developing statistical models in NLP?

   The frequency of any word is inversely proportional to its rank in the frequency table, i.e. $f \times r \approx k$

   There will be a lot of infrequent (and zero-frequency) words, hence leading to the zero probability problem, however this can be solved using various smoothing techniques (Laplace, add-one, Good Turing, etc.)

   Another problem is that very few words have very high frequency.

2. Is it always/sometimes/never better to choose N>3 for an N-gram language model? Why?

   Sometimes better, because higher-order N-grams are sensitive to more context, but have sparse counts.

   Lower-order N-grams have limited context, but robust counts.

   Higher-order N-grams require more training data (exponentially more) is required to build a reliable model. Problem is zero probability if there are not enough data and thus require smoothing.

3. Explain what is meant by the bag-of-words assumption + example of a probabilistic model that uses this + appropriate task for this model.

   Bag-of-words assumption: the words in a document are standalone and treated as if there is no dependency between any of the words, for example, disregarding the grammar and word order.

   Probabilistic model: Naive Bayes

   Appropriate task: document type classification (text classification)

4. Benefit of computing negative log probabilities (costs) instead of actual probabilities and why can't we do this when implementing the forward algorithm for HMMs.

   Multiplying lots of small probabilities quickly gets so tiny we can't represent the numbers accurately, even with double precision floating point.

Actual probabilities are often too small and thus multiplications are more resource intensive.

Saves computing power and easier operation, hence increase the speed.

We can't do this when implementing the forward algorithm because there is no probability multiplication involved, only addition.

5. Suppose we train two different character trigram models on English text.

The first model is trained w/o padding. That is, it defines the probability of a token w consisting of characters $c_1 \ldots c_n$ as:

$$P(w = c_1 \ldots c_n) = P(c_1)\, P(c_2 | c_1) \prod_{i=3}^{n} P(c_i | c_{i-2}, c_{i-1})$$

The second model is trained with padding (i.e. using explicit begin and end of sequence markers).

Using each model, we then compute the per-character cross-entropy of tokens (1) and (2) below:

(1) acknowledge (2) cknowledge

a. For the model with padding, which of these tokens is likely to have a higher per-character cross-entropy? Explain.

Token (2) will have a higher per-character cross-entropy because a lot of English words start with 'ac' but none of them starts with 'ck'.

b. For token (2), which model is likely to have higher per-character cross-entropy? Explain.

The second model with padding is likely to have higher per-character cross-entropy because the average entropy of starting word with the sequence 'ck'.

6. Consider the following two sentences:

(1) I ate soup in the restaurant. (2) I ate soup in the pot.

What type of syntactic ambiguity do these sentences illustrate? Explain how these sentences motivate the use of a lexicalized parsing model. (Hint: what happens if you use a non-lexicalized PCFG to parse these sentences? Compare that to either a lexicalized PCFG or dependency grammar.)

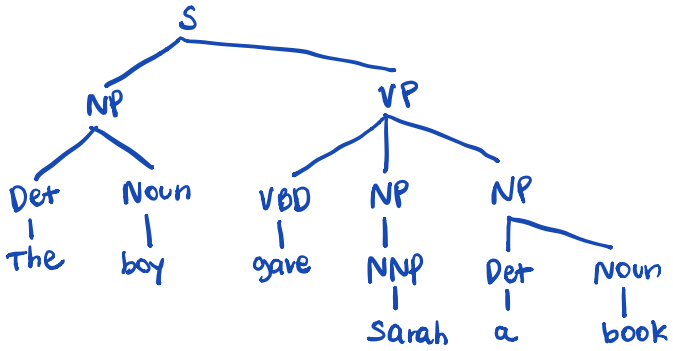
Structural ambiguity → Attachment ambiguity.

A non-lexicalized PCFG will give the same parse for (1) and (2) when it shouldn't. A lexicalized parsing model will take into account the dependency between the words and will be able to give different parses for (1) and (2).

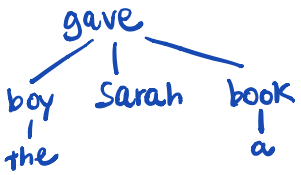
7. What is the difference between a constituency parse and a dependency parse? Provide structural analysis for the following sentence using each method:

The boy gave Sarah a book.

Constituency parse




Dependency parse




?

Difference: dependency parsing only shows relationships between words and their constituents while constituency parsing displays the entire sentence structure and relationships.

8. Fill in the blanks.

a. In terms of lexical semantic relationships, 'toddler' is a hyponym of 'child'.

b. In WordNet, 'child' and 'kid' belong to the same synset because in many contexts, they are synonyms.

# PART B

1. Language models and HMMs.

   a. Suppose we train a bigram language model on a corpus where there are no occurences of the bigram perfume and.

      i. What is the MLE for P(and|perfume)? What problem(s) will this cause when using the LM? Illustrate your answer with examples.

      P(and|perfume) = 0.

      This will cause overfitting the language model.

      MLE gives unseen examples a probability of 0. this will make the whole sentence's probability 0.
      → 'dominate' the probability chain.

      ii. Laplace (add-1) or back-off: which is likely to estimate a higher value for P(and|perfume)? Why?

      Laplace; it has a tendency to assign too much probability to unseen words.

   b. N-gram language model with smoothing, will cross-entropy of training set be higher than test set? Why. (per-word)

   Test set will have higher cross-entropy because there is more uncertainty on data that the language model has never seen.

   c. HMMs are often used for chunking: identifying short sequences of words (chunks) within a text that are relevant for a particular task. For example, if we want to identify all the person names in a text, we could train a HMM using annotated data similar to the following:

   On/O Tuesday/O ,/O Mr/B Cameron/I met/O with/O Chancellor/B Angela/I Merkel/I ./O

   B: beginning of a chunk
   I: words inside a chunk
   O: words outside of chunk
   SS → start of sentence
   ES → end of sentence

      i. Write down an expression for the probability of generating the sentence Sally met Dr Singh tagged with the sequence B O B I.

      $$P(B\ O\ B\ I \mid \text{Sally met Dr Singh}) = P(B|SS) \cdot P(\text{Sally}|B) \cdot P(O|B) \cdot P(\text{met}|O) \cdot P(B|O) \cdot P(\text{Dr}|B) \cdot P(I|B) \cdot P(\text{singh}|I) \cdot P(ES|I)$$

* P(Q|LAMBDA) where $q_n$ = state at time/step n
$o_n$ = word at time/step n
LAMBDA = transition and emission probabilities.

ii. Crucially, the O and SS tags may not be followed by I because we need to have a B first indicating the beginning of a chunk.

What, if any, changes would you need to make to the Viterbi algorithm in order to use it for tagging sentences with this BIO scheme? How can you incorporate the constraint on which tags can follow which others

P(I|O) = 0 and P(I|SS) = 0

2. Spelling correction

a. Formulation of the spelling correction task as a noisy channel model. Let x → word typed, y → intended word, and give the name for each component of the model. Which component is typically more task-specific and which part is more similar across different tasks that use noisy channel models?

$\text{argmax}_y P(y|x) = \text{argmax}_y P(x|y) P(y)$

where $P(y)$ is the LM

$P(x|y)$ is the noise model

The noise model is task-specific (e.g. depends on the keyboard layout), the LM remains well applicable across many tasks.

b. Noise model:

$$P(x|y) = \prod_{i=1}^{n} P(x_i|y_i)$$

observed, intended

individual characters in x and y (including the empty char.) that have been aligned to each other.

What independence assumptions does this noise model make?

Character typed only depends on the intended character, not on any other characters.

c. Twitter text normalization, e.g. convert (1a) to (1b)

(1a) I nate manny ppl but mi twitter ppl lovh ya!

(1b) I hate many people but my twitter people love you!

Use them to explain why the independence assumption in the simple noise model above are too strong to capture the changes needed for twitter normalization

'Manny' and 'many' have a different no. of characters and a simple spelling correction would fail on that.

Ppl is short for people. A normalization should take this into account.

d. Consider trying to normalize the following:

(2) @SwiftOnaRampage lol no comment bro... can't say if I disagree or agree. lol

(3) Really wish 1 of the #sixxtards won. Haha mayb a book? Wld b my 3rd copy. My 1st copy got worn out had to buy a new1

Give two examples of difficult annotation decisions from this dqia: that is, tokens where annotators might disagree about the correct normalised form unless very specific guidelines are given. For each example explain briefly why it is difficult (ie provide two or more alternative normalization and say why each might be reasonable).

3rd → it's a valid form but annotators might want to keep everything in characters and no digits.

Bro → actual slang valid word used in direct speech, annotators might want to change into brother, the original form.

3. Lexical Semantics

'Jobs' → ambiguous between the employment sense and former Apple CEO Steve Jobs.

We want to classify tweets that mention 'Jobs' in the Steve Jobs sense. Example:

(+) Apple had a third founder besides Jobs and Wozniak.
(-) I had two Jobs working like 30 hrs a week while attending school.

a. Off-the-shelf entity recognizer / supersense tagger: which tag would uniquely ID all the cases where the tagger thinks "Jobs" → Steve Jobs?

Person.

b. Why won't the off-the-shelf system work very well for this task?

NER typically uses some form of feature-based sequence tagging like capitalization so if Jobs isn't capitalized, the system might not work.

tweets are very short; it may well be that the NER was trained on bigger contexts. The off-the-shelf NER is also unlikely to make use of hints like 'Apple' if it was trained to just recognise persons in general (supersense).

thus, in NER, richer features are sometimes used, like capitalisation. we might need to use some Domain Adaptation techniques if the NER was trained on a very different corpus.

c. Train your own classifier. Need (+)/(-)-marked tweet collection. Is crowdsourcing reasonable for annotation? If not, why + examples. If yes, explain why + 2 annotation strategies that would ensure the dataset is useful.

Yes → this task is simple, does not require any pre-training for ordinary human.

Two annotation strategies:
1. Redundancy to combat noise: elicit 5+ annotations per data point.
2. Embed datapoints with known answers, reject annotators who get them wrong.

d. Supervised classification model to train the labeled dataset + corresponding technique to avoid overfitting.

MaxEnt with regularization (such as MLE).

e. Treat problem as an unsupervised distributional clustering task.

i. Show non-zero entries of a sparse context vector for the word 'Jobs' in the following sentences:

a. Apple had a third founder besides Jobs and Wozniak. Ronald Wayne had a 10% stake. He forfeited his share for a total of $2300.

b. It was amazing. I had two Jobs working like 30hrs a week while attending school raised my grades 10%.

Lowercase the words and use a window of 5 words on each side (ignoring punctuation). Circle discriminative words.

a. [had, a, third, (founder), besides, and, wozniak, ronald, wayne, had]

b. [was, amazing, i, had, two, (working), like, 30hrs, a, week]

ii. Compute cosine similarity between two vectors.

| | had | a | third | founder | besides | and | wozniak | ronald | wayne |
|---|---|---|---|---|---|---|---|---|---|
| a | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| b | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | was | amazing | i | two | working | like | 30hrs | week |
|---|---|---|---|---|---|---|---|---|
| a | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| b | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

$\vec{a} = [2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$

$\vec{b} = [1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]$

$\vec{a} \cdot \vec{b} = 3$

$\|\vec{a}\| = \sqrt{2^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2} = \sqrt{12}$

$\|\vec{b}\| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2} = \sqrt{10}$

$$\text{sim}(a, b) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|} = \frac{2}{\sqrt{12} \cdot \sqrt{10}} = \frac{2}{\sqrt{120}} \approx 0.1826$$