— Solution notes —

Practice Exam: Natural Language Processing 1 – 2025 – Paper 1

1 ()

(*a*) Name two limitations of n-gram models as a model of language. [1 point]

Answer: Only 2 answers are required, that could be one of the following:

(*i*) N-gram models are unable to capture long-distance dependencies.

(*ii*) N-gram models are unable to take word similarities into account: tokens are atomic and information that is learned about a token is not transferred to similar tokens.

(*iii*) N-gram models are unable to generalise from training data to new events.

(*iv*) In terms of memory an n-gram model theoretically grows exponentially $O(|V|^n)$.

(*v*) data sparsity - without back off and interpolation poor estimates for higher order n-grams.

Note that students may propose other valid answers, this is in principle an open-ended question.

(*b*) You are given the following training corpus, where each word is annotated with its part-of-speech (POS) tag. The POS tagset includes: DT (determiner), IN (conjunction, preposition), NN (noun, singular or mass), NNS (noun, plural), VB (verb, base form), VBN (verb, past participle), MD (verb, modal auxiliary), and PUN (punctuation mark). The corpus will be used to train a part-of-speech tagger based on a Hidden Markov Model (HMM).

```
the_DT government_NN will_MD cut_VB taxes_NNS on_IN income_NN ._PUN
the_DT cut_NN should_MD be_VB blocked_VBN ._PUN
```

(*i*) Compute the estimates that would be obtained for lexical probabilities (emission probabilities) and tag sequence probabilities (transition probabilities) under a bigram model. You should not separate the sentences and you should not use start-of-sentence and end-of-sentence symbols. [3 points]

Answer:

(A) Emission probabilities:

- P(the|DT) = 1.0
- P(government|NN) = 0.33
- P(will|MD) = 0.5
- P(cut|VB) = 0.5
- P(taxes|NNS) = 1.0
- P(on|IN) = 1.0
- P(income|NN) = 0.33
- P(.|PUN) = 1.0
- P(cut|NN) = 0.33

- P(should|MD) = 0.5
- P(be|VB) = 0.5
- P(blocked|VBN) = 1.0

(B) Transition probabilities:

- P(NN|DT) = 1.0
- P(MD|NN) = 0.67
- P(VB|MD) = 1.0
- P(NNS|VB) = 0.5
- P(IN|NNS) = 1.0
- P(NN|IN) = 1.0
- P(PUN|NN) = 0.33
- P(DT|PUN) = 1.0
- P(VBN|VB) = 0.5
- P(PUN|VBN) = 1.0

(*ii*) Compute the probability of the following sequence of part-of-speech tags assigned to the sentence, first under the unigram and then under the bigram model, given the probabilities obtained in *(b)(i)*. Provide the formulae you used to compute sequence probabilities under the unigram and bigram models, defining all variables. You may assume p(t_1='DT')=1.0.

```
the_DT cut_NN will_MD be_VB blocked_VBN ._PUN
```

[1 point]

Answer:

(A) Unigram probability: 0.00000053. Given by $\prod_{i=1}^{N} P(w_i|t_i)P(t_i)$.

(B) Bigram probability: 0.028. Given by $P(w_1|t_1) \cdot \prod_{i=2}^{N} P(w_i|t_i)P(t_i|t_{i-1})$

(*c*) You are given the following probabilistic context-free grammar (PCFG):

```
(1.0) S  -> NP VP
(0.42) NP -> PRP N
(0.58) NP -> PRP
(0.2) VP -> V S
(0.5) VP -> V NP
(0.3) VP -> V
```

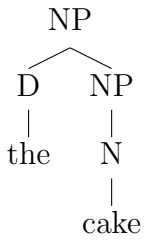
```
(0.25) PRP  -> We
(0.75) PRP  -> her
(1.0) N    -> duck
(0.18) V    -> duck
(0.82) V    -> saw
```

(*i*) Show all possible sentence trees that can be produced for the following example sentence with this grammar.

```
We saw her duck
```

[1 point]

Note that a sentence tree can be represented compactly in bracketed structure, for instance:



⇒ `(NP (D the) (NP (N cake)))`

You can choose what representation to use in your answer.

Answer: Either a sentence tree or a bracketed structure is acceptable as an answer. Note that the non-terminal nodes should be labeled correctly. I show bracketed structures corresponding to two possible trees:

We saw her duck

```
(S (NP (PRP We)) (VP (V saw) (NP (PRP her) (N duck))))

(S (NP (PRP We)) (VP (V saw) (S (NP (PRP her)) (VP (V duck)))))
```

(*ii*) Find the most probable parse and its probability for the following sentence (the same sentence as in question *(c)(i)*). You can use either dynamic programming or enumeration, but clearly describe all steps of your answer.

```
We saw her duck
```

[1 point]

Answer: Using enumeration, and multiplying from left to right in order of the bracketed structure, we obtain a probability of 0.01872675 for sentence tree 1:

```
1.0 * 0.58 * 0.25 * 0.5 * 0.82 * 0.42 * 0.75 * 1.0 = 0.01872675
```

For sentence tree 2 we obtain a probability of 0.0005585922:

```
1.0 * 0.58 * 0.25 * 0.2 * 0.82 * 1.0 * 0.58 * 0.75 *
0.3 * 0.18 = 0.0005585922
```

This means that the first sentence tree corresponds to the most likely parse.

With the Viterbi algorithm we obtain the following probabilities at the bottom layer where we already encounter an ambiguity:

```
(PRP We) = 0.25
(V saw) = 0.82
(PRP her) = 0.75
(N duck) = 1.0
(V duck) = 0.18
```

Then, the probabilities at the next layer are given by:

```
(NP (PRP We)) = 0.58 * 0.25 = 0.145
(NP (PRP her) (N duck)) = 0.42 * 0.75 * 1.0 = 0.315
(NP (PRP her)) = 0.58 * 0.75 = 0.435
(VP (V duck)) = 0.3 * 0.18 = 0.054
```

The probability at the third layer is:

```
(S (NP (PRP her)) (VP V duck)) = 1.0 * 0.435 * 0.054 = 0.02349
```

Now on the fourth layer an ambiguity arises:

```
max((VP (V saw) (NP (PRP her) (N duck))),
(VP (V saw) (S (NP (PRP her)) (VP V duck)))) =
max(0.5 * 0.82 * 0.315, 0.2 * 0.82 * 0.02349)
= max(0.12915, 0.00385236) = 0.12915
```

And the final probability of the first sentence tree, which is the most probable parse:

```
(S (NP (PRP We)) (VP (V saw) (NP (PRP her) (N duck)))) =
1.0 * 0.145 * 0.12915 = 0.01872675
```

Which is the same answer as we obtained with enumeration.

(*d*) Which of the following techniques is more sensitive to the differences in genre when trained on one corpus (e.g. news data) and evaluated on another (e.g. a corpus of scientific articles)? Briefly explain and motivate your answer.

- NGram language models
- PCFG language models.

[1 point]

Answer: NGram language models are more sensitive to changes in genre than PCFGs. This is because PCFGs provide a level of generalisation over the data in the form of phrase categories and syntactic rules over them. NGrams on the other hand use atomic categories, i.e. words, and are therefore dependent on the specific vocabulary used in the corpus. This vocabulary may vary across genres quite substantially.

(*e*) Consider the following three phrases: *chocolate cake*, *birthday cake*, *blue collar*. (Note on the meaning of *blue collar*: *blue collar* workers do work needing strength or physical skill rather than office work.)

(*i*) Give two reasons why such phrases can be problematic for semantic composition models. Make sure to mention all examples in your answer at least once. [1 point]

Answer:

(A) In noun-noun compounds, the relationships holding between the compound's head and modifier can be very diverse and idiosyncratic. The meaning of such a compound cannot be entirely derived from the meaning of the two nouns in isolation; additional meaning arises through composition. For example, the modifier *chocolate* in the compound *chocolate cake* refers to a key ingredient in the cake (*a cake containing chocolate*) whereas the modifier *birthday* in *birthday cake* refers to the occasion on which the cake is served (*a cake served at a birthday party*).

(B) Unlike in the first two examples, the meaning of the phrase *blue collar* isn't conveyed by the head. The meaning of this expression cannot be directly derived from the meaning of the two words that compose it; it is external to the literal meaning of the compound. Moreover, although the meaning of the expression cannot be derived compositionally, the compositional interpretation is still possible.

(*ii*) Which semantic relation holds between *chocolate cake* and *cake*? Which semantic relation holds between *blue collar* and (the denim shirt of) a working class member? [1 point]

Answer:

(A) *Chocolate cake* is a hyponym of *cake* and *cake* is a hypernym of *chocolate cake*. They are in a *is-a* relationship.

(B) A *blue collar* is a part of (the denim shirt of) a working class member, so the relation holding between them is meronomy. (The figure of speech that gives rise to the expression blue collar is metonymy or, more precisely, synecdoche. Though the latter part is not required to get full points for this question.)

(*f*) Processing sentences with a bidirectional LSTM model allows us to model context in both directions at each time step. Can this model be used to distinguish between different word senses of the same word in two different sentences? Explain why or why not. [1 point]

Answer: BiLSTMs naturally produce "sense-aware" representations of each word (at each time step). This is because the hidden representations at each time step are computed based on the input embedding of the word and the hidden representation at the previous time step (or the next time step in case of the backward LSTM). Therefore, they naturally model word meanings in context and allow us to discriminate between different word senses of a given word in different contexts. Of course, these representations are typically not general-purpose representations of the corresponding word sense, but rather they capture task-specific properties of word senses, that are relevant for the task at hand.

(*g*) List two advantages of neural co-reference resolution over a classification method that uses manually-engineered linguistic features. [1 point]

Answer: Open-ended question. Possible answers include:

(1) neural coreference resolution has access to word meanings through the word embeddings.

(2) the neural model of Lee et al. (2017) discussed in the lecture allows us to model co-reference chains (as opposed to individual pairs) and use information from the previous links for subsequent classification.

(*h*) Explain how the attention mechanism is used in neural co-reference resolution method of Lee et al. (2017) discussed in the lecture. What kinds of linguistic information does this mechanism help the model to capture? [1 point]

Answer: The attention mechanism is used to compute span representations. Specifically, the model computes a weighted sum of word representations in the span. It allows the model to identify the words in the span which provide the most information for the task of co-reference resolution. For instance, syntactic heads in the noun phrases might receive higher attention weights.

(*i*) Interpretation of word and sentence meanings is often influenced by the surrounding discourse, for instance, the immediately preceding sentences. Discuss one way in which discourse information can be incorporated into neural sentence representation learning using an attention mechanism. You can use sentence-level sentiment classification as an example task and assume that the documents containing the sentence of interest are provided. [3 points]

Answer: Open-ended question, which combines ideas from sentence representation learning (lecture 6) and document representation learning (lecture 7). One (likely) answer is: when performing sentence-level sentiment analysis, maintain a representation of the surrounding discourse in your model. The representation of the surrounding discourse can be computed using e.g. an LSTM with word- or sentence-level attention. This discourse representation can then be concatenated with the representation of the sentence of interest (also computed e.g. using an LSTM), and this discourse-aware representation can then be fed to a classification head. The model is trained end-to-end and the task-specific discourse representations are learned as a result.

(*j*) A neural model of abstractive summarisation can be thought of as a conditional language generator. The architecture encodes the available text and predicts a neural parameterisation of a distribution over summaries. The typical model for this task factorises the probability of a summary, given the input text, autoregressively using a bidirectional LSTMs encoder and an LSTM decoder with an attention mechanism in between the two (so that the encoder states can be interpolated differently for each generation step). Here's a sketch: let $x_{1:L}$ be the input text and $y_{<j}$ be the prefix summary before the jth word is generated; the probability distribution of the next token Y_j in the summary is Categorical(softmax($\mathbf{s}_j$)) with $\mathbf{s}_j \in \mathbb{R}^V$ defined as follows:

$$\mathbf{s}_j = \text{linear}_V(\mathbf{h}_j; \theta_{\text{out}}) \tag{1a}$$
$$\mathbf{h}_j = \text{decoderstep}(\mathbf{h}_{j-1}, \mathbf{c}_j, \mathbf{t}_{j-1}; \theta_{\text{dec}}) \tag{1b}$$
$$\mathbf{c}_j = \text{attention}(\mathbf{e}_{1:L}, \mathbf{h}_{j-1}; \theta_{\text{att}}) \tag{1c}$$
$$\mathbf{e}_{1:L} = \text{encoder}(x_{1:L}; \theta_{\text{enc}}) \tag{1d}$$
$$\mathbf{t}_{j-1} = \text{embed}(y_{j-1}; \theta_{\text{in}}) \tag{1e}$$

with V being vocabulary size, and for some adequately initialised decoder state $\mathbf{h}_0$.

Describe a mechanism to incorporate ideas from *extractive* summarisation into this model: propose a modification motivating its connection to extractive summarisation, do explain the architecture/components you introduce (there's no need for formulae, instead, you should be able to explain how these components combine with the baseline model, what are the relevant inputs and outputs and what roles they are expected to play). [2 points]

Answer: In extractive summarisation, we use passages from the input text as part of the summary. To incorporate this strategy as part of the model we can use a *pointer network*. Essentially, we employ an attention mechanism (which doesn't have to be the one already in the architecture) to produce normalised attention coefficients over the input tokens. We then regard these coefficients as an expression of tendency to 'copy' from the input into the output. We can combine these probabilities with the token probabilities we get when we compute $\text{softmax}(\mathbf{s}_j)$, for example, we can interpolate them, or sum and renormalise (other strategies are also possible, so long as the result remains a valid probability distribution per time step). The intuition is that the pointer network's probabilities will potentially make tokens that are already in the input more probable to be generated, adding a bias towards copying from input, rather than generating new tokens from the vocabulary.
