Week 4 Discussion: Morphology and Language Models

Edoardo Ponti (School of Informatics, University of Edinburgh)

Goals

Discussion group meetings will mostly focus on exam questions from previous years. In particular, they will include open-ended questions whose solution requires creative thinking and the ability to put together different pieces of information learned throughout the ANLP course. This will prepare you to tackle similar open-ended questions during the final exam.

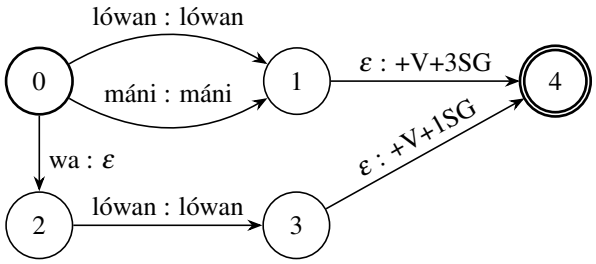
Discussion Questions

Problem 1

In Lakhota, an indigenous language of North America, some verbs, such as *lówan* ('sing'), are inflected for subject agreement by adding a prefix. Alone, *lówan* agrees with a third person singular subject, but with the prefix *wa*, it agrees with a first person singular subject. However, other verbs, such as *máni* ('walk'), are inflected by adding an infix. Alone, *máni* agrees with a third person singular subject, but with the infix *wá*, and a change in the pronunciation of the first vowel from *á* to *a*, it agrees with a first person singular subject. The table below summarizes these words, their meaning, and their morphological analysis. Affixes are underlined.

third person			first person		
word	meaning	analysis	word	meaning	analysis
lówan	'he sings'	lówan+V+3SG	walówan	'I sing'	lówan+V+1SG
máni	'he walks'	máni+V+3SG	mawáni	'I walk'	máni+V+1SG

Now consider the following finite state transducer.



Q 1.1 Is this transducer deterministic or non-deterministic? Why?

Q 1.2 Does this finite state transducer correctly analyse the words *lówan*, *walówan*, and *máni*? If not, which word(s) does it get wrong, and why?

Q 1.3 Add states and transitions to the transducer so that it correctly analyses the word *mawáni*. You do not need to redraw the transducer above, but you should identify any states that you reuse via their associated number; new states need not be numbered. It is not important whether the result is deterministic or non-deterministic.

Problem 2

For this question, imagine I have a jar containing some unknown number of balls. Each ball is labelled with one of four different words: *red*, *blue*, *happy*, or *sad*.

I repeatedly draw out a ball, write down the word, and then replace the ball in the jar. In 5 draws, I get: *red*, *happy*, *happy*, *blue*, *happy*.

Q 2.1 Given the data I observed and using maximum-likelihood estimation (MLE) and a unigram (1-gram) language model,

- What is the estimated probability of getting happy on the next draw?
- What is the likelihood of the original 5-word sequence (rounded to two significant digits)?

Q 2.2 Given the data I observed and using add-alpha smoothing with $\alpha = 0.1$, what is the estimated probability of getting *happy* on the next draw (rounded to two significant digits)?

Q 2.3 Suppose I record a large number of draws **from this jar** to create a much larger "corpus" of observations, which I split into a training, development, and test set. I then build two models of the data: a unigram model and a bigram model. If I optimise the smoothing method and hyperparameters appropriately, which model would you expect to have lower perplexity on the test set? Justify your answer.

Does the answer change if the unigram model and the bigram model are estimated from real texts from Wikipedia instead?

Q 2.4 Unlike the scenario described above, natural language has no fixed vocabulary. However, some language models are built using a fixed vocabulary, which consists of the *N* most common words in the training data. All other words are replaced by the token *UNK*. For a unigram language model built in this way, will smoothing always be needed? Justify your answer.

Problem 3

You are training a neural language model for Polish, a morphologically rich language with many different inflectional forms for each lemma. For instance, *jabłko* (apple) can be inflected as: *jabłko, jabłka, jabłek, jabłku, jabłkom, jabłkiem, jabłkami, jabłkach*. As a consequence, many words appear very infrequently (if at all) in your training data.

Q 3.1 What kind of problem does this long tail of infrequent words give rise to in neural language models? What are possible solutions?

Q 3.2 Polish has a grammatical agreement between a verb and the person and number of its subject. How do fixed-window feed-forward neural networks handle long-distance grammatical agreement?

Q 3.3 You compare the performance of two neural architectures: the log-linear model introduced by Mnih and Hinton (2007) and the multi-layer perceptron introduced by Bengio et al. (2000).

- Which model is more expressive? In other words, which model is equivalent to a larger family of functions?
- Similar to Mnih and Hinton (2007), after training on a very small dataset, you find the log-linear model to perform better than the multi-layer perceptron. Can you hypothesise possible reasons why?

References

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems*, 13.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648.

Week 4 Discussion Solutions: Morphology and Language Models

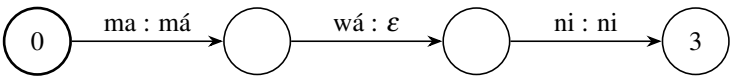
Edoardo Ponti (School of Informatics, University of Edinburgh)

Problem 1

S 1.1 Non-deterministic, due to epsilon transitions

S 1.2 Yes

S 1.3 There are two things that the solution should do: it should change the phonology of the first syllable; and it should remember the infix and return its analysis at the end of the string. One possible solution is given below; another is to rewrite *máni* to *mawáni* in a single transition from state 0 to state 3. The solution does not need to reuse state 0 or state 3 to be correct.



Problem 2

S 2.1

- 0.6 (or 3/5)
- 0.0086

S 2.2 0.57. A possible common incorrect answer would be 0.58 (from 3.1/5.3), based on counting the number of types actually observed (3), rather than the number of possible types (4).

S 2.3 A good answer needs to point out that the data are generated from a unigram distribution in the scenario where sequences are sampled from a jar. In this case, a bigram model would never be expected to do better on the test set than a unigram model, regardless of optimisation. A good answer might conclude that with the appropriate use of development set, the unigram and bigram models would have similar perplexity on the test set. An excellent answer would indicate a better understanding of some subtleties: for example, they might point out that the bigram model is more likely to overfit the development set, especially if the dataset is not that large, so if anything the bigram model is likely to have higher perplexity on the test set (but given enough data, the two models would have similar perplexity).

If the sequences are sampled from Wikipedia instead, the bigram model will perform better. This is because longer contexts better approximate the full history.

S 2.4 No, smoothing will not always be needed for such a model. Since there is a fixed vocabulary, if the training data is large enough it would be possible to have enough observations of even the lowest frequency items so that MLE would be a good estimate. (Saying that it would be possible to "avoid zeros" is not enough for full credit, the counts actually need to be large enough to get a good estimate, not just a non-zero estimate.) Note that the answer depends on the fixed vocabulary, not on the order of the N-gram—the answer would be the same for a bigram model, just that the data size would need to be larger. Answers that imply otherwise will get little to no credit.

Problem 3

S 3.1 The main problem is that unknown words at test time have no word embeddings in the look-up table. In addition, the word embedding of infrequent words is updated only around as many times as that word occurs in the training data (in single-epoch training); hence, their representations are likely close to random initialisation. In terms of solutions, unknown words can be replaced by a special <UNK> (unknown) symbol. In addition, the student should mention alternative tokenizers (e.g. BPE) that yield a smaller vocabulary more robust to data sparsity.

S 3.2 They can't. Similarly to most *n*-gram models, fixed-window neural language models also make a Markovian assumption about conditional probabilities. Hence, they are not equipped to model long-distance dependencies such as subject–verb–object agreement. Recurrent Neural Networks (RNNs) and Transformers instead are capable of modelling arbitrary sequence lengths.

S 3.3 Multi-layer Perceptrons are universal approximators, i.e., they can model non-linear functions (Note: in the limit of infinite width). Hence, they are more expressive than log-linear models. The reason why Mnih and Hinton (2007) reported better results could be related to the possible overfitting of the multi-layer perception (given the limited data size). Another hypothesis is that they relied on a more stable optimisation algorithm. Finally, the bilinear model re-uses the input embeddings to obtain the scores in output, which reduces overfitting.

References

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648.

ANLP Tutorial / Week 6

Shay Cohen (School of Informatics, U of Edinburgh)

Goals

This week's tutorial exercises focus on HMMs and tagging. By completing it, you will learn to hand simulate some of the important HMM algorithms, and gain a better understanding of the strengths and weaknesses of this type of model (i.e., what kinds of information it can and cannot use). Some of this tutorial is based on a tutorial from 2021 (by Sharon Goldwater).

Problem 1

Suppose we want to train an HMM tagger for the task of Named Entity Recognition (NER). We are interested in only two kinds of named entities: persons (`PER`) and organizations (`ORG`), which include corporate and political entities. We have the following training data:[1]

David/`PER` William/`PER` Donald/`PER` Cameron/`PER` (born 9 October 1966) is a British politician who has served as the Prime Minister of the United/`ORG` Kingdom/`ORG` since 2010 , as Leader of the Conservative/`ORG` Party/`ORG` since 2005 and as the Member of Parliament for Witney/`ORG` since 2001 .

Cameron/`PER` studied Philosophy , Politics and Economics at Brasenose/`ORG` College/`ORG` , Oxford/`ORG` .

He then joined the Conservative/`ORG` Research/`ORG` Department/`ORG` and became special adviser , first to Norman/`PER` Lamont/`PER` and then to Michael/`PER` Howard/`PER` .

He was Director of Corporate Affairs at Carlton/`ORG` Communications/`ORG` for seven years .

Cameron/`PER` first stood for Parliament in Stafford in 1997 .

In this data we show only the tags for the tokens belonging to the person and organization categories. Assume all other tokens have the tag `OTH`, which is not shown. Tokens are separated by whitespace. *Note:* There are a total of 73 types and 105 tokens in the text.

[1] The text is a lightly edited version of the start of the Wikipedia article on David Cameron, from Oct 2015.

a) Give the transition probability matrix estimated from this training data using maximum-likelihood estimation. Don't forget to include beginning and end of sentence markers.

b) Now do the same but using add-one smoothing. Assume that all sentences must contain at least one word (i.e., $P(\texttt{</s>}|\texttt{<s>})$ is zero even in the smoothed model).

c) Again using add-one smoothing, what are the estimates for $P(\text{Cameron}|\texttt{PER})$ and $P(\text{Cameron}|\texttt{ORG})$?

Problem 2

This question also deals with NER and HMMs, but asks you to consider the nature of the problem and proposed solution, rather than working through the mathematical details.

a) Suppose we used *four* tags for this task: the three already mentioned, plus a `LOC` tag for locations. In a general text, will context always be able to disambiguate between the `LOC` and `ORG` tags? Justify your answer.

b) POS taggers are normally evaluated using *accuracy*: (the percentage of the tags assigned by the tagger that agree with the gold standard). Does this evaluation measure also make sense for NER, or are there other evaluation measures we've discussed in class that you think would make more sense? Why?

Problem 3

Consider a simple HMM POS tagger with only five tags (plus the beginning and end of sentence markers, `<s>` and `</s>`). The transition probabilities for this HMM are given by the first table below, where cell $[i,j]$ is the probability of transitioning from state i to j (i.e., $P(\text{state}_j|\text{state}_i)$). A subset of the output probabilities are given by the second table below, where cell $[i,j]$ is the probability of state i outputting word j (i.e., $P(\text{word}_j|\text{state}_i)$). We assume there are other possible output words not shown in the table, and that the `<s>` and `</s>` states output `<s>` and `</s>` words, respectively, with probability 1.

	CD	PRP	NN	VB	VBD	</s>
<s>	.5	.2	0	.3	0	0
CD	.2	0	.3	.2	.2	.1
PRP	.1	.1	0	.3	.4	.1
NN	.05	.15	.2	.25	.3	.05
VB	0	.2	.6	0	0	.2
VBD	0	.1	.6	0	0	.3

	one	cat	dog	bit	...
CD	.1	0	0	0	
PRP	.02	0	0	0	
NN	.05	.03	.04	.007	
VB	0	0	.03	0	
VBD	0	0	0	.06	

a) In the Penn Treebank tag scheme, what do the five different tags mean? Give example sentences illustrating the use of each word in the output matrix with each of its possible tags. (Your sentences should be real English, not limited to just the words/tags used in our tiny HMM.)

b) Using the HMM probability matrices, compute $P(\vec{w}, \vec{q})$ (the joint probability of words and tags) for the sentence $\vec{w}$ = `<s> one dog bit </s>` with tags $\vec{q}$ = `<s> CD NN NN </s>`.

c) Now, hand-simulate the Viterbi algorithm in order to compute *highest probability* tag sequence $\vec{q}'$ for the given sentence, and the joint probability $P(\vec{q}', \vec{w})$, without enumerating all possible tag sequences. That is, fill in the cells in the following table, where cell $[j, t]$ should contain the Viterbi value for state j at time t, and you should also use backpointers to keep track of the best path. The rows of the table are already labeled with the different states, and the columns are already labeled with the observations at each time step.

Hint: For this particular HMM, a lot of the cells will have zeros in them. Try to work out ahead of time which these are, so you only need to do the Viterbi computations for the other cells.

	<s>	one	dog	bit	</s>
<s>					
CD					
PRP					
NN					
VB					
VBD					
</s>					

d) As you've seen, Viterbi probabilities get very small very fast. In practice, the algorithm is normally implemented using log probabilities to avoid underflow. The value in each cell is now a *negative log probability* (or *cost*), and we end up computing $-\log P(\vec{w}, \vec{q})$. Work out what the equations need to be in this version of the algorithm. That is, what do we compute to get the value in cell (j, t)?

Problem 4

a) We learned about the Viterbi algorithm, which aims to find a tag sequence that *maximizes* $P(T, S)$ (based on the notation in class). The Viterbi algorithm is similar to the *forward algorithm*, which replaces the max operator with addition (+).[2] The forward algorithm recursive formula, therefore, is defined as

$$\alpha(t+1, j) = \sum_i \alpha(t, i) P(j \mid i) P(w_{t+1} \mid j),$$

where w_t is the word in position t, and i and j range over the states (tags). The initial case of the recursive formula stays the same as with Viterbi.

Show (using mathematical induction or otherwise) that this recursive formula computes the prefix probability:

$$\alpha(t, j) = P(w_1, \ldots, w_t, T_t = j).$$

The interpretation of this is the algorithm computes the joint distribution of the prefix of the string it analyses up to position t $(w_1, \ldots, w_t)$ together with the probability of the last state j given as a parameter.

b) Given what you have showed in the first part of the exercise, how would you calculate $P(S)$ based on the α calculations?

Mathematical Induction: What is meant by "proof by induction"? In its simplest form, it is a mathematical argument that enables you to establish the validity of an equation (or statement) involving an integer parameter by following two fundamental steps. Assume that the statement or equation you aim to prove is denoted as $\Gamma(t)$, and you intend to prove its truth for any integer value of t $(\forall t\, \Gamma(t))$. In the above context, $\Gamma(t)$ represents the statement $\alpha(t, j) = P(w_1, \ldots, w_t, T_t = j)$.

The two steps to be taken are as follows:

- You need to demonstrate the truth of $\Gamma(1)$ (for the case when $t = 1$; sometimes the base is $t = 0$).
- You should establish that *if* $\Gamma(t)$ is true, then $\Gamma(t+1)$ is also true $(\forall t\, \Gamma(t) \rightarrow \Gamma(t+1))$.

In the second step, you are not confirming the truth of $\Gamma(t)$ itself. Instead, you are showing that if it were true, you could progress to the next step in the "ladder" and prove the truth of $\Gamma(t+1)$. This condition, combined with the initial or "base" step, guarantees the overall validity of the general statement. (Mathematical induction is a bit like climbing a ladder, where climbing each step prepares you for the next.)

[2] This can be more generally viewed as changing the "semiring" of the dynamic programming algorithm.

Conditional Independence: In class, we briefly discussed the conditional independence assumptions that we make with HMMs. The salient part of these assumptions is that a given generated word w_t is conditionally independent of anything else in the sequence given the tag (or "state") it was generated from (denoted by the random variable T_t).
Formally speaking, conditional independence in this case means that for any "event" A based on the HMM it holds that

$$P(A, w_t \mid T_t = i) = P(A \mid T_t = i)P(w_t \mid T_t = i),$$

as long as the event A depends only on information that is embodied before T_t and w_t. For example, A can be $w_1, \ldots, w_{t-1}$ or A can be $T_{t-1} = j$ or even both of those events together.
This kind of conditional independence and the way it is formulated is central to HMMs and the reason it is possible to run, for example, the Viterbi algoritm with full correctness. It also highly central to solving this exercise.

ANLP Tutorial / Week 6

Shay Cohen (School of Informatics, U of Edinburgh)

Goals

This week's tutorial exercises focus on HMMs and tagging. By completing it, you will learn to hand simulate some of the important HMM algorithms, and gain a better understanding of the strengths and weaknesses of this type of model (i.e., what kinds of information it can and cannot use). Some of this tutorial is based on a tutorial from 2021 (by Sharon Goldwater).

Problem 1

Suppose we want to train an HMM tagger for the task of Named Entity Recognition (NER). We are interested in only two kinds of named entities: persons (`PER`) and organizations (`ORG`), which include corporate and political entities. We have the following training data:[1]

David/`PER` William/`PER` Donald/`PER` Cameron/`PER` (born 9 October 1966) is a British politician who has served as the Prime Minister of the United/`ORG` Kingdom/`ORG` since 2010 , as Leader of the Conservative/`ORG` Party/`ORG` since 2005 and as the Member of Parliament for Witney/`ORG` since 2001 .

Cameron/`PER` studied Philosophy , Politics and Economics at Brasenose/`ORG` College/`ORG` , Oxford/`ORG` .

He then joined the Conservative/`ORG` Research/`ORG` Department/`ORG` and became special adviser , first to Norman/`PER` Lamont/`PER` and then to Michael/`PER` Howard/`PER` .

He was Director of Corporate Affairs at Carlton/`ORG` Communications/`ORG` for seven years .

Cameron/`PER` first stood for Parliament in Stafford in 1997 .

In this data we show only the tags for the tokens belonging to the person and organization categories. Assume all other tokens have the tag `OTH`, which is not shown. Tokens are separated by whitespace. *Note:* There are a total of 73 types and 105 tokens in the text.

[1] The text is a lightly edited version of the start of the Wikipedia article on David Cameron, from Oct 2015.

a) Give the transition probability matrix estimated from this training data using maximum-likelihood estimation. Don't forget to include beginning and end of sentence markers.

b) Now do the same but using add-one smoothing. Assume that all sentences must contain at least one word (i.e., $P(\texttt{</s>}|\texttt{<s>})$ is zero even in the smoothed model).

c) Again using add-one smoothing, what are the estimates for $P(\text{Cameron}|\texttt{PER})$ and $P(\text{Cameron}|\texttt{ORG})$?

Solutions

a) Note that punctuation are separate tokens, and tagged `OTH`, so all words before `</s>` are `OTH`. The probabilities are therefore

	PER	ORG	OTH	</s>
<s>	3/5	0	2/5	0
PER	5/10	0	5/10	0
ORG	0	6/13	7/13	0
OTH	2/82	7/82	68/82	5/82

b) These can be obtained from the MLE probability matrix as follows: in the `<s>` row, add 1 to each numerator and 3 to each denominator in the first three columns, leaving a 0 in the fourth column. (We added a 1 to each of three different options for the next tag, so we need to add 3 in the denominator for a valid distribution.) In all other rows, add 1 to each numerator and 4 to each denominator. (Similar reasoning.)

c) MLE are 3/10 and 0/13, so we get 4/83 and 1/86. (In this case, we assume the vocabulary is just what we've seen in the corpus, so there are 73 different words, thus 73 possible outputs for each tag. Adding a count of one for each in the numerator means adding 73 in the denominator.)

Problem 2

This question also deals with NER and HMMs, but asks you to consider the nature of the problem and proposed solution, rather than working through the mathematical details.

a) Suppose we used *four* tags for this task: the three already mentioned, plus a `LOC` tag for locations. In a general text, will context always be able to disambiguate between the `LOC` and `ORG` tags? Justify your answer.

b) POS taggers are normally evaluated using *accuracy*: (the percentage of the tags assigned by the tagger that agree with the gold standard). Does this evaluation measure also make sense for NER, or are there other evaluation measures we've discussed in class that you think would make more sense? Why?

Solutions

a) Arguably, no: for example, Witney can be tagged here as an organization, as it's the political constituency (thus political entity, i.e., organization) that Cameron represents. But it is also a location, and this sentence doesn't make clear which meaning the writer intended. The same ambiguity holds for the other locations mentioned in the text. One might ask whether disambiguating these cases even matters, as the meaning of the sentence is basically the same either way. This is a good example of a case where humans don't normally notice (or care) about ambiguity, but it can make life difficult for corpus annotators and computer systems.

   Notice that I used a specific example to justify my claim here. In general, a good justification should include specific evidence, which often means concrete examples. This is especially true if you can find a good counterexample. That is, if you want to argue that claim X does not hold in general, then a single good counterexample can prove that point. Similarly, if you are trying to argue that X is possible, an example will provide evidence for that claim. Examples are not sufficient to prove that a claim holds in general, but thinking about them might still give you something to say about the kinds of cases in which the claim is likely to be true.

b) This will of course depend on exactly what you plan to do with the system, but it seems likely that in most texts, there will be many more words that are non-NEs (the `OTH` class) than NEs (all other classes). That is, the classes are very *unbalanced*. So just predicting `OTH` for every word is likely to achieve relatively high accuracy, but is not very useful. One alternative would be to compute the precision and recall (and/or F-score) of the system on each tag. (Can you see how to do this?) Or, if we mainly care about whether we found the NEs (but not which type), we could just treat the task as a binary classification (is this word part of an NE or not?) and compute precision and recall on that basis.

Problem 3

Consider a simple HMM POS tagger with only five tags (plus the beginning and end of sentence markers, `<s>` and `</s>`). The transition probabilities for this HMM are given by the first table below, where cell $[i,j]$ is the probability of transitioning from state i to j (i.e., $P(\text{state}_j|\text{state}_i)$). A subset of the output probabilities are given by the second table below, where cell $[i,j]$ is the probability of state i outputting word j (i.e., $P(\text{word}_j|\text{state}_i)$). We assume there are other possible output words not shown in the table, and that the `<s>` and `</s>` states output `<s>` and `</s>` words, respectively, with probability 1.

	CD	PRP	NN	VB	VBD	</s>
<s>	.5	.2	0	.3	0	0
CD	.2	0	.3	.2	.2	.1
PRP	.1	.1	0	.3	.4	.1

	CD	PRP	NN	VB	VBD	</s>
NN	.05	.15	.2	.25	.3	.05
VB	0	.2	.6	0	0	.2
VBD	0	.1	.6	0	0	.3

	one	cat	dog	bit	...
CD	.1	0	0	0	
PRP	.02	0	0	0	
NN	.05	.03	.04	.007	
VB	0	0	.03	0	
VBD	0	0	0	.06	

a) In the Penn Treebank tag scheme, what do the five different tags mean? Give example sentences illustrating the use of each word in the output matrix with each of its possible tags. (Your sentences should be real English, not limited to just the words/tags used in our tiny HMM.)

b) Using the HMM probability matrices, compute $P(\vec{w}, \vec{q})$ (the joint probability of words and tags) for the sentence $\vec{w}$ = `<s> one dog bit </s>` with tags $\vec{q}$ = `<s> CD NN NN </s>`.

c) Now, hand-simulate the Viterbi algorithm in order to compute *highest probability* tag sequence $\vec{q}'$ for the given sentence, and the joint probability $P(\vec{q}', \vec{w})$, without enumerating all possible tag sequences. That is, fill in the cells in the following table, where cell $[j, t]$ should contain the Viterbi value for state j at time t, and you should also use backpointers to keep track of the best path. The rows of the table are already labeled with the different states, and the columns are already labeled with the observations at each time step.

Hint: For this particular HMM, a lot of the cells will have zeros in them. Try to work out ahead of time which these are, so you only need to do the Viterbi computations for the other cells.

	<s>	one	dog	bit	</s>
<s>					
CD					
PRP					
NN					
VB					
VBD					
</s>					

d) As you've seen, Viterbi probabilities get very small very fast. In practice, the algorithm is normally implemented using log probabilities to avoid underflow. The value in each cell is now a *negative log probability* (or *cost*), and we end up computing $-\log P(\vec{w}, \vec{q})$. Work out what the equations need to be in this version of the algorithm. That is, what do we compute to get the value in cell (j, t)?

Solutions

a) cardinal number, personal pronoun, singular/mass noun, base verb, past tense verb. Examples include:

- One/`CD` person is here. One/`PRP` can see Arthur's Seat from here. The one/`NN` you are looking for is here.
- I like the cat/`NN`.
- The dog/`NN` is brown. His past mistakes dog/`VB` him to this day.
- Give me a bit/`NN` of cake. The dog bit/`VBD` me.

b) $(.5)(.3)(.2)(.05)(.1)(.04)(.007) = 4.2 \times 10^{-8}$.

c) The probabilities are as follows:

	<s>	one	dog	bit	</s>
<s>	1	0	0	0	0
CD	0	.05	0	0	0
PRP	0	.004	0	0	0
NN	0	0	6×10^{-4}	1.26×10^{-6}	0
VB	0	0	3×10^{-4}	0	0
VBD	0	0	0	1.08×10^{-5}	0
</s>	0	0	0	0	3.24×10^{-6}

If we number rows and columns starting from 0, then the backpointers are:

- in column `one`, both non-zero cells came from `<s>`.
- in column `dog`, both non-zero cells came from `CD`.
- in column `bit`, the `NN` cell came from `VB` and the `VBD` cell came from `NN`.
- in column `</s>`, the non-zero cell came from `VBD`.

Following the backpointers shows that the highest probability path is:

`<s> CD NN VBD </s>`

which is in this case the correct tag sequence.

d) The only difference is subtracting log probs instead of multiplying probs and taking a min instead of a max, because we're now using costs. For state j at time t, let $c(j,t)$ be the cost in cell $[j,t]$ of the chart. You'd need to compute:

$$\begin{aligned}
c(j,t) &= -\log v(j,t) \\
&= -\log \left[\max_{i=1}^{N} \left(v(i,t-1) \cdot a_{i,j} \cdot b_j(o_t)\right)\right] \\
&= -\max_{i=1}^{N} \log \left[v(i,t-1) \cdot a_{i,j} \cdot b_j(o_t)\right] \\
&= \min_{i=1}^{N} -\log \left[v(i,t-1) \cdot a_{i,j} \cdot b_j(o_t)\right] \\
&= \min_{i=1}^{N} \left[-\log v(i,t-1) - \log a_{i,j} - \log b_j(o_t)\right] \\
&= \min_{i=1}^{N} \left[c(i,t-1) - \log a_{i,j} - \log b_j(o_t)\right]
\end{aligned}$$

Problem 4

a) We learned about the Viterbi algorithm, which aims to find a tag sequence that *maximizes* $P(T,S)$ (based on the notation in class). The Viterbi algorithm is similar to the *forward algorithm*, which replaces the max operator with addition (+).[2] The forward algorithm recursive formula, therefore, is defined as

$$\alpha(t+1,j) = \sum_i \alpha(t,i) P(j \mid i) P(w_{t+1} \mid j),$$

where w_t is the word in position t, and i and j range over the states (tags). The initial case of the recursive formula stays the same as with Viterbi.

Show (using mathematical induction or otherwise) that this recursive formula computes the prefix probability:

$$\alpha(t,j) = P(w_1, \ldots, w_t, T_t = j).$$

The interpretation of this is the algorithm computes the joint distribution of the prefix of the string it analyses up to position t $(w_1, \ldots, w_t)$ together with the probability of the last state j given as a parameter.

b) Given what you have shown in the first part of the exercise, how would you calculate $P(S)$ based on the α calculations?

[2]This can be more generally viewed as changing the "semiring" of the dynamic programming algorithm.

Mathematical Induction: What is meant by "proof by induction"? In its simplest form, it is a mathematical argument that enables you to establish the validity of an equation (or statement) involving an integer parameter by following two fundamental steps. Assume that the statement or equation you aim to prove is denoted as $\Gamma(t)$, and you intend to prove its truth for any integer value of t ($\forall t\, \Gamma(t)$). In the above context, $\Gamma(t)$ represents the statement $\alpha(t, j) = P(w_1, \ldots, w_t, T_t = j)$.

The two steps to be taken are as follows:

- You need to demonstrate the truth of $\Gamma(1)$ (for the case when $t = 1$; sometimes the base is $t = 0$).
- You should establish that *if* $\Gamma(t)$ is true, then $\Gamma(t + 1)$ is also true ($\forall t\, \Gamma(t) \rightarrow \Gamma(t + 1)$).

In the second step, you are not confirming the truth of $\Gamma(t)$ itself. Instead, you are showing that if it were true, you could progress to the next step in the "ladder" and prove the truth of $\Gamma(t + 1)$. This condition, combined with the initial or "base" step, guarantees the overall validity of the general statement. (Mathematical induction is a bit like climbing a ladder, where climbing each step prepares you for the next.)

Conditional Independence: In class, we briefly discussed the conditional independence assumptions that we make with HMMs. The salient part of these assumptions is that a given generated word w_t is conditionally independent of anything else in the sequence given the tag (or "state") it was generated from (denoted by the random variable T_t).
Formally speaking, conditional independence in this case means that for any "event" A based on the HMM it holds that

$$P(A, w_t \mid T_t = i) = P(A \mid T_t = i)P(w_t \mid T_t = i),$$

as long as the event A depends only on information that is embodied before T_t and w_t. For example, A can be $w_1, \ldots, w_{t-1}$ or A can be $T_{t-1} = j$ or even both of those events together.
This kind of conditional independence and the way it is formulated is central to HMMs and the reason it is possible to run, for example, the Viterbi algoritm with full correctness. It also highly central to solving this exercise.

Solutions

Note: Even if you do not fully understand the derivation here, try to understand the notion of conditional independence of the w_i from the rest of the sequence given the tag at position i. This is the basis for this derivation.

Assume that $w_0 =$`<s>`. The proof by induction works as follows. For the base case, we know that $\alpha(0, j) = 1$ if and only if j is the state `<s>`, as necessary (because $p($`<s>`$,$ `<s>`$) = 1$).

Assume that $\alpha(t, i)$ computes $P(w_1, \ldots, w_t, T_t = i)$ (the induction hypothesis). What does $\alpha(t+1, i)$ compute? Based on the induction hypothesis, we want to show it computes:

$$P(w_1, \ldots, w_{t+1}, T_{t+1} = i). \tag{1}$$

Consider the expression

$$P(w_1, \ldots, w_t, w_{t+1}, T_t = i, T_{t+1} = j). \tag{2}$$

According to the chain rule and conditional independence assumptions that HMMs make, another way to write it is as:

$$P(w_1, \ldots, w_t, T_t = i \mid T_{t+1} = j) P(w_{t+1} \mid T_{t+1} = j) P(T_{t+1} = j). \tag{3}$$

We can also rewrite the first and last term in the above (by the definition of conditional probability):

$$P(w_1, \ldots, w_t, T_t = i \mid T_{t+1} = j) P(T_{t+1} = j) \tag{4}$$

as

$$P(w_1, \ldots, w_t, T_t = i, T_{t+1} = j). \tag{5}$$

This can be further rewritten as

$$P(w_1, \ldots, w_t, T_t = i) P(T_{t+1} = j \mid T_t = i). \tag{6}$$

because $w_1, \ldots, w_t$ are conditionally independent of T_{t+1} given T_t.

We therefore get that

$$\begin{aligned} &P(w_1, \ldots, w_t, w_{t+1}, T_t = i, T_{t+1} = j) && (7) \\ &\quad = \underbrace{P(w_1, \ldots, w_t, T_t = i)}_{\alpha(t,i)} P(T_{t+1} = j \mid T_t = i) P(w_{t+1} \mid T_{t+1} = j). && (8) \end{aligned}$$

The righthand side in the above is exactly the term for the sum in the definition of $\alpha(t+1, j)$ because by the induction hypothesis $\alpha(t, i) = P(w_1, \ldots, w_t, T_t = i)$ (as marked by curly brackets below the term). On the lefthand side, if we follow the rule of total probability, we get that summing it over i gives

$$P(w_1, \ldots, w_t, w_{t+1}, T_{t+1} = j). \tag{9}$$

Therefore, summing both sides of Equation 8 over i on both sides gives the equality (because Equation 8 is true for any i):

$$\sum_i P(w_1, \ldots, w_t, w_{t+1}, T_t = i, T_{t+1} = j) \tag{10}$$

$$= \sum_i \alpha(t, i) P(T_{t+1} = j \mid T_t = i) P(w_{t+1} \mid T_{t+1} = j). \tag{11}$$

On the lefthand side above, we exactly get by the law of total probability $P(w_1, \ldots, w_i, w_{i+1}, T_{t+1} = j)$, because we are summing out T_t. On the right, by definition of the forward recursive formula, this is $\alpha(t+1, j)$. Therefore:

$$P(w_1, \ldots, w_i, w_{i+1}, T_{t+1} = j) = \alpha(t+1, j), \tag{12}$$

which is what we needed to show.

It is important to note that all throughout, we rely on that

$$P(T_{t+1} = j \mid T_t = i)$$

takes the probability $p(j \mid i)$ from the transition matrix (from state i to state j).

For calculating $P(S)$, what you would need to do is consider the last time step $\alpha(t, j)$ values, and sum over j. This sums out the hidden tag in the last position, and according to the law of total probability, will give $P(T)$.

ANLP Tutorial Week 8

Shay Cohen (School of Informatics, University of Edinburgh)

Goals

This week's tutorial exercises focus on probabilistic parsing and transition-based parsing. After working through the exercises and working with your discussion groups, you should be able to:

- Simulate the CYK algorithm on a PCFG, where rules are associated with probabilities. You will understand the main inductive step in CYK in the context of non-binary rules.
- Estimate the probabilities of a PCFG from a small corpus.
- Explore ideas regarding dependency grammars and parsing and understand better transition-based parsing.

Problem 1

Consider the following probabilistic context free grammar:

S →	Subj VP (1.0)
VP →	V Obj (0.5) | V Obj Obj (0.3) | V Small (0.2)
Small →	Obj V (1.0)
Subj →	I (0.3) | NP (0.7)
Obj →	her (0.2) | NP (0.8)
NP →	N (0.5) | Det N (0.5)
V →	make (0.6) | duck (0.4)
N →	duck (0.5) | goose (0.5)
Det →	her (1.0)

Use a probabilistic CYK-style algorithm to find the most probable parse for the sentence below and determine its probability.

I make her duck

N.B. Do not convert the grammar to Chomsky Normal Form — instead, you should draw up a CYK-*style* parse chart for the grammar by combining cells from more than two other

cells. For small examples this is perfectly feasible, but you should check that you understand why a CYK-style algorithm has difficulties with non-CNF grammars in general. (Hint: think about what additional inferences you might need to make in order to work with grammars in this form.)

Problem 2

a) Derive a PCFG grammar from the corpus below. Write down the rules of the grammar and calculate their probabilities. Assume for the purpose of the grammar that the corpus is lowercased (e.g. *The* has been replaced by *the*).

1.

```
(S
    (NP She)
    (VP
        (VP
            (V saw)
            (NP (Det the) (N man)))
        (PP
            (P from)
            (NP (Det a) (N distance)))))
```

2.

```
(S
    (NP Here)
    (VP
        (V is)
        (NP (Det a) (N telescope))))
```

3.

```
(NP
    (NP (Det the) (N man))
    (PP (P with) (NP (Det the) (N guitar))))
```

4.

```
(S
    (NP He)
    (VP
        (V saw)
        (NP (Det the) (N girl))))
```

5.

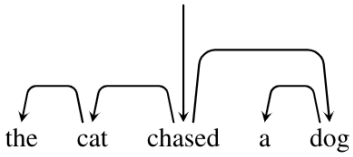
```
(S
    (NP (Det The) (N man))
    (VP
        (V saw)
        (NP
            (NP (Det the) (N girl))
            (PP
                (P with)
                (NP (Det the) (N flowers))))))
```

b) Draw all possible parse trees for the following sentence, and compute their probabilities using the probabilistic grammar you have created.

He saw the man with the telescope

Problem 3

Consider the following dependency-annotated sentence. (For simplicity, we leave out the relation labels in this exercise).



By hand-simulating the algorithm for arc-standard transition-based parsing, show that there is more than one sequence of transitions that can lead to the correct parse of this sentence. How does this fact motivate the need for the procedure described in JM3 section 18.2.1 (generating the training oracle)? What is the sequence produced by the training oracle?

ANLP Tutorial Week 8

Shay Cohen (School of Informatics, University of Edinburgh)

Goals

This week's tutorial exercises focus on probabilistic parsing and transition-based parsing. After working through the exercises and working with your discussion groups, you should be able to:

- Simulate the CYK algorithm on a PCFG, where rules are associated with probabilities. You will understand the main inductive step in CYK in the context of non-binary rules.
- Estimate the probabilities of a PCFG from a small corpus.
- Explore ideas regarding dependency grammars and parsing and understand better transition-based parsing.

Problem 1

Consider the following probabilistic context free grammar:

S →	Subj VP (1.0)
VP →	V Obj (0.5) | V Obj Obj (0.3) | V Small (0.2)
Small →	Obj V (1.0)
Subj →	I (0.3) | NP (0.7)
Obj →	her (0.2) | NP (0.8)
NP →	N (0.5) | Det N (0.5)
V →	make (0.6) | duck (0.4)
N →	duck (0.5) | goose (0.5)
Det →	her (1.0)

Use a probabilistic CYK-style algorithm to find the most probable parse for the sentence below and determine its probability.

I make her duck

N.B. Do not convert the grammar to Chomsky Normal Form — instead, you should draw up a CYK-*style* parse chart for the grammar by combining cells from more than two other

cells. For small examples this is perfectly feasible, but you should check that you understand why a CYK-style algorithm has difficulties with non-CNF grammars in general. (Hint: think about what additional inferences you might need to make in order to work with grammars in this form.)

Solutions

The probabilistic CKY-style chart is as follows (note this is not 'pure CYK' as the grammar is not in Chomsky Normal Form). The derivations of cell probabilities are shown in selected cases.

	I	make	her	duck
I	Subj(0.3)		S(1.0x0.3x0.06=0.018)	S(1.0x0.3x0.06=0.018)
make		V(0.6)	VP(0.5x0.6x0.2=0.06)	VP(0.5x0.6x0.2=0.06)
her			Det(1.0),Obj(0.2)	NP(0.25),Obj(0.2),Subj(0.175), Small(1.0x0.2x0.4=0.08)
duck				V(0.4),N(0.5),NP(0.25), Obj(0.2),Subj(0.175)

The main interest comes in the cell (make,duck). There are three competing analyses of 'make her duck' as a VP, with probabilities as follows:

a) Rule VP → V Obj: 0.5x0.6x0.2 = 0.06
b) Rule VP → V Obj Obj: 0.3x0.6x0.2x0.2 =0.0072
c) Rule VP → V Small: 0.2x0.6x0.08 = 0.0096

Rule a is the most probable, and is the only one we need record in the chart. Note that for the purpose of computing the most probable analysis of the whole sentence, this is the only analysis of 'make her duck' that need be considered.

Thus, the most probable overall parse is

```
(S
    (Subj I)
    (VP
        (V make)
        (Obj
            (NP
                (Det her)
                (N duck)))))
```

with probability 0.018.

Problem 2

a) Derive a PCFG grammar from the corpus below. Write down the rules of the grammar and calculate their probabilities. Assume for the purpose of the grammar that the corpus is lowercased (e.g. *The* has been replaced by *the*).

1.

```
(S
    (NP She)
    (VP
        (VP
            (V saw)
            (NP (Det the) (N man)))
        (PP
            (P from)
            (NP (Det a) (N distance)))))
```

2.

```
(S
    (NP Here)
    (VP
        (V is)
        (NP (Det a) (N telescope))))
```

3.

```
(NP
    (NP (Det the) (N man))
    (PP (P with) (NP (Det the) (N guitar))))
```

4.

```
(S
    (NP He)
    (VP
        (V saw)
        (NP (Det the) (N girl))))
```

5.

```
(S
    (NP (Det The) (N man))
    (VP
        (V saw)
        (NP
            (NP (Det the) (N girl))
```

```
(PP
    (P with)
    (NP (Det the) (N flowers))))))
```

b) Draw all possible parse trees for the following sentence, and compute their probabilities using the probabilistic grammar you have created.

He saw the man with the telescope

Solutions

a) Here are the rules and their probabilities (which are written as rational fraction)

Grammar rules	
S → NP VP (4/4)	V → saw (3/4)
VP → VP PP (1/5)	V → is (1/4)
VP → V NP (4/5)	Det → the (7/9)
NP → Det N (9/14)	Det → a (2/9)
NP → NP PP (2/14)	N → man (3/9)
NP → he (1/14)	N → girl (2/9)
NP → she (1/14)	N → distance (1/9)
NP → here (1/14)	N → guitar (1/9)
PP → P NP (3/3)	N → telescope (1/9)
P → from (1/3)	N → flowers (1/9)
P → with (2/3)	

b) Although not asked for by the question, here is the parse chart (as a matrix):

	He	saw	the	man	with	the	telescope
He	NP(1/14)			S(1/140)			S(1/18900)
saw		V(3/4)		VP(1/10)			VP_1(1/1350),VP_2(1/1890)
the			Det(7/9)	NP(1/6)			NP(1/1134)
man				N(1/3)			
with					P(2/3)		PP(1/27)
the						Det(7/9)	NP(1/18)
telescope							N(1/9)

Here VP_1 →VP PP and VP_2 →V NP. Since VP_1 has the higher probability, strictly speaking the chart should only include this one, and this is compute the probability of S in the top-right cell.

Thus, the most likely parse is

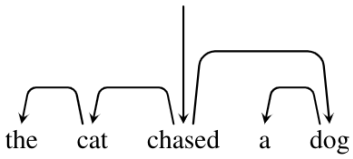
```
(S
    (NP he)
```

```
(VP
    (VP
        (V saw)
        (NP
            (Det the)
            (N man)))
    (PP (P with)
        (NP
            (Det the)
            (N telescope))))
```

with probability 1/18900. The other parse has probability 1/26460.

Problem 3

Consider the following dependency-annotated sentence. (For simplicity, we leave out the relation labels in this exercise).



By hand-simulating the algorithm for arc-standard transition-based parsing, show that there is more than one sequence of transitions that can lead to the correct parse of this sentence. How does this fact motivate the need for the procedure described in JM3 section 18.2.1 (generating the training oracle)? What is the sequence produced by the training oracle?

Solutions

Here are two possible sequences (you might find others). The first is the training oracle sequence, which chooses LeftArc as soon as possible in all cases.

Step	Stack	Word list	Action	Relation added
0	[root]	[the, cat, chased, a, dog]	*Shift*	
1	[root, the]	[cat, chased, a, dog]	*Shift*	
2	[root, the, cat]	[chased, a, dog]	*LeftArc*	(the ← cat)
3	[root, cat]	[chased, a, dog]	*Shift*	
4	[root, cat, chased]	[a, dog]	*LeftArc*	(cat ← chased)
5	[root, chased]	[a, dog]	*Shift*	
6	[root, chased, a]	[dog]	*Shift*	
7	[root, chased, a, dog]	[]	*LeftArc*	(a ← dog)

Step	Stack	Word list	Action	Relation added
8	[root, chased, dog]	[]	$RightArc$	(chased → dog)
9	[root, chased]	[]	$RightArc$	(root → chased)
10	[root]	[]	$DONE$	

Step	Stack	Word list	Action	Relation added
0	[root]	[the, cat, chased, a, dog]	$Shift$	
1	[root, the]	[cat, chased, a, dog]	$Shift$	
2	[root, the, cat]	[chased, a, dog]	$LeftArc$	(the ← cat)
3	[root, cat]	[chased, a, dog]	$Shift$	
4	[root, cat, chased]	[a, dog]	$Shift$	
5	[root, cat, chased, a]	[dog]	$Shift$	
6	[root, cat, chased, a, dog]	[]	$LeftArc$	(a ← dog)
7	[root, cat, chased, dog]	[]	$RightArc$	(chased → dog)
8	[root, cat, chased]	[]	$LeftArc$	(cat ← chased)
9	[root, chased]	[]	$RightArc$	(root → chased)
10	[root]	[]	$DONE$	

The training oracle is needed in order to define a set of actions that will lead to a correct parse, and are also as consistent as possible. In other words, when we train the classifier to decide an action, we want the training data (sequences of configurations from the training oracle) to be as consistent as possible about what action is taken given a particular configuration or partial configuration, because consistent patterns are easier to learn than random ones.

Week 4 Discussion: Exam Practice

Edoardo Ponti (School of Informatics, University of Edinburgh)

Goals

Discussion group meetings will mostly focus on exam questions from previous years. In particular, they will include open-ended questions whose solution requires creative thinking and the ability to put together different pieces of information learned throughout the ANLP course. This will prepare you to tackle similar open-ended questions during the final exam. The questions below were adapted from previous exams.

Before you attend the tutorial, we strongly advise that you practice the following questions **under exam conditions**. This means that you should give yourself a time limit (we suggest 45 minutes), and **write your answers on paper**, without using reference materials. In the exam, each of these questions should be answered with, at most, a few sentences — not multiple paragraphs, and definitely not pages. It may nevertheless take a few minutes for you to think through what those sentences should be for each answer. During your tutorial session, you should discuss your answers with your tutorial group; this will give you an opportunity to think through strengths and weaknesses of your answers and uncover new perspectives on them.

Discussion Questions

Problem 1

You plan to develop a shift-reduce dependency parser. You found a publicly available treebank, containing examples where each sentence is associated to a dependency tree. You also have word embeddings for each of the words in the input to the parser.

Recall that a shift-reduce parser includes a function that must choose among a set of n actions (where $n \geq 2$) given the current stack and input buffer.

1. Which of the following models is **not** suitable for implementing the function that chooses the next parser action? Justify your answer.

   A. Multi-variate logistic regression;

   B. Hidden Markov Model;

   C. Multi-layer Perceptron;

   D. Naïve Bayes;

2. To help choose the correct action, the parser should be able to model *morphosyntactic agreement* between two words. For instance, in some languages an adjective is inflected according to the morphosyntactic features of the noun it modifies. One example is Ancient Greek, where adjectives and noun inflections must agree in gender, number, and case.

You want to know whether the word embeddings alone are enough to model this type of agreement, or whether additional features would also help. Please give *one* other type of feature that you think might help the parser choose the correct action at each step, how you would obtain it, and why you think it would help model morphosyntactic agreement.

3. Parsers often benefit from semantic knowledge, such as the fact that the direct object of *drink* is much more likely to be *water* than *shelf*. Embeddings help model semantics, because words with similar meanings often have similar embeddings. But they are known to fail in modeling one aspect of semantics: the difference between antonyms (words with opposite meaning) and synonyms (words with the same meaning). For example, the embedding for *love* is similar to the embeddings for *cherish* and *adore*, but also to the embedding for *hate*.

   (a) Explain why antonyms and synonyms often occur near each other in the embedding space.

   (b) Will this issue with the embeddings be a problem for your dependency parser? Explain why or why not.

Problem 2

Medical coding is a widespread practice of classifying medical reports by assigning codes (labels) to them. For every diagnosis or medical procedure that might occur in a hospital, there is a corresponding code. For example, a medical record and corresponding procedure code might look like the following.

medical record:

Patient is a 67-year-old female presents to the emergency room with sharp, shooting pain in her lower abdomen and pronounced swelling. Patient is nauseous, has vomited, and has a fever. Abdomen is firm and slightly distended. Patient states she has no history of abdominal problems, disease, or hernia.

Physical examination suggests appendicitis. Ultrasound test ordered, and diagnosis of appendicitis is confirmed. Patient is rushed to surgery and is prepped for general anesthesia. Once anesthetized, patient receives appendectomy via laparoscopy.

procedure code: `44970`
(44970 is the code for laparoscopic surgery to remove the appendix)

Every procedure has a cost, so procedure codes determine medical bills, and there are over ten thousand possible codes. Your company develops medical billing software, and your manager wants you to automate medical coding for a hospital in Birmingham, which uses standard international codes. Through a partnership with NHS Scotland, which also uses standard international codes across all of its hospitals, your company has obtained tens of thousands of medical records and corresponding codes from a hospital in Inverness. Your manager suggests that you use these to develop the system, and reminds you of the golden rule of medical coding: "*Do not code it or bill for it if it's not documented in the medical record.*"

1. What data resources would you need in order to evaluate the system and how would you obtain them?

2. What metric would you use to evaluate the system, and why? If your metric is common, you do not need to give a formula for it or explain how it works; but you do need to justify why it is appropriate for this problem.

3. You need to design a model for the task. What should the *input* and *output* of the model should be? What kind of model would you need to handle this kind of input and output?

4. Describe two ways in which you might expect the training data to be inadequate for the intended task, leading to a poor and / or biased system.

Week 4 Discussion Solutions: Exam Practice

Edoardo Ponti (School of Informatics, University of Edinburgh)

Problem 1

1. Give 1 point for indicating Hidden Markov Model (HMM) as an answer, and 1 point for explaining this is because HMM is not a classifier.

2. Full marks if the feature is well-chosen and all 3 points are hit clearly, less if some parts are vague or missing.

   To model morpho-syntactic agreement, the most relevant features are parts of speech (POS), which can be obtained from an automatic tagger (e.g., an HMM), and morphological features. For instance, in the sentence "*The ancestors of his mother were nobles*", the subject dependency between *were* and *nobles* can be ascertained from the fact that they are a verb and a noun, respectively. Moreover, they agree in their morphosyntactic features, as they are both plural.

3. (a) Assign 1.5 points for clearly explaining the distributional hypothesis and 1.5 points for clearly explaining how it impacts antonym conflation. (Fewer marks for vaguer answers.)

   The reason why word embeddings (both sparse and dense) conflate synonyms and antonyms is due to the distributional hypothesis: words are assigned representations based on the terms they co-occur with. This implies also that words found in similar contexts have similar representations. This is the case for antonyms such as *love* and *hate*, which share all semantic dimensions (e.g. they are human emotions, have strong intensity, etc.) except one. For instance, in review databases they will co-occur with the same set of words, such as *movie*, *critic*, etc.

   (b) No, because selectional preferences are mostly about the kind of things involved (eg animate versus object versus abstract), and synonyms and antonyms are typically the same kind of thing.

Problem 2

The goal of this question is for you to think critically about system design and potential problems in advance of attempting to build a system. There are many possible approaches to each subquestion, but some are better matched to the problem than others. For this reason, you should not regard the discussion below as a comprehensive guide to the "correct" answer; instead, it points out things that you may want to think about. You may have equally good replies that are not covered below.

1. You need a separate test set (or sets) to evaluate your system. Although you already have data from Inverness, and could hold aside some data for this purpose,

you should prefer to evaluate your system using coded examples from the hospital where you intend to deploy the system in Birmingham. Otherwise, you could easily miss problems that arise from differences in the data distribution between the two hospitals (question 4).

Weaker answers to this question are those that do not engage with the question content. For example: discussing evaluation in general terms, or describing general procedures for evaluating systems, e.g. train-test-split or cross-fold validation. In an exam, answer like these would likely receive partial credit for demonstrating knowledge of the course material; but not full credit, since they don't engage with the problem at hand.

2. Some things to think about: The distribution of procedures will naturally be imbalanced. Classification accuracy or micro-averaged F would focus accuracy on most frequent classes, but we also care about infrequent classes, so might prefer macro-averaged F. But note that the question suggests that not all errors are equal: we don't want to bill for things not in the report. This suggests that a biasing towards precision is preferable.

   Weaker answers would include giving the definitions of common metrics and the types of problems that they are used on; or talking about intrinsic and extrinsic evaluation in general terms.

3. This question asks you to identify the general *form* of the problem. In this case, it's a text classification problem: text input, and a class label as ouput (the medical code). If you consider the setting for a moment, you may also recognise that it is likely to be a *multi-label classification* problem, i.e. one in which an input document may receive multiple output labels. This is because a medical report may describe more than one procedure! Though we didn't explicitly discuss multi-label output in class, the algorithms are similar to those for text classification that we did discuss.

   There are wrong answers to this question: it isn't a tagging or parsing problem, for instance.

4. Some possible sources of problems / bias:

   - Different cities will have different demographic patterns, which will in turn affect the distribution of health issues that are common in the population, and in turn the distribution of procedures that are commonly used. For instance, some procedures are more commonly used on children, and others, on the elderly. (Although we wouldn't expect you to know this, Inverness and Birmingham have very different demographics: Birmingham has a much younger and much more racially diverse population.)
   - Many procedures will be rare and thus some codes will have relatively few examples. At the scales described in the question, many codes will be rare and some may not be represented at all.
   - A twist on the above answer is that some hospitals may specialise in certain procedures, which will skew the results.
   - A classifier could pick up on language *about* patients that correlates with identity, e.g. "female" in the example. This may be helpful for classifying certain procedures that correlate with, e.g. age or gender; but it could

also lead to spurious associations with others, resulting in biased outcomes. (Real example: studies have found that in US hospitals, African Americans are less likely to receive treatment for certain conditions, and this bias is replicated in automated systems.)

- It may be that the local dialect used in medical reports varies from hospital to hospital, and thus some associations found by a classifier in one hospital would be less useful for another.