Exam IN4080 2019 Solution proposal

1(a) Part-of-speech tags

- What do we mean by "part-of-speech"?
- What does it mean that a text is tagged with part-of-speech?
- Why is it useful in natural language processing to part-of-speech-tag a text?

Fill in your answer here

Max 5 points

A part of speech (POS) is a category of words, also called *word class*. Each word is assigned one part of speech. Words that are assigned the same part of speech have similar distributions and share grammatical and semantical properties. Exactly how many---and which---parts of speech one recognizes, vary somewhat from language to language (and according to theoretical framework), but there are typically around 10 different of them. The most familiar (and stable across frameworks) are noun, verb, adjective, preposition, pronoun.

In a POS-tagged text each word is assigned one POS which is assumed to be the correct POS for this token in this context. The tags ascribed to the words in a text may contain more information than just the POS, including information regarding subcategory or form; the number of different tags might be higher than the number of POS.

A POS-tagged text is less ambiguous than a raw text. Hence, it may give improved results for text classification tasks, like sentiment classification. It may also serve as first steps towards further analysis, e.g. syntactic parsing and named-entity recognition.

1(b) Hidden Markov models

- Given a sentence $w_1, w_2, \ldots, w_n$ and a tag set, what is the goal of a part-of-speech tagger?
- How does a hidden Markov model use the Bayes formula to solve this task?
- Which simplifying assumptions are made by the hidden Markov model?
- Give the formula for the hidden Markov model!

Max 10 points

- The goal of a part of speech tagger is to assign a part-of-speech tag t_i to each token w_i such that the sequence of tags $t_1, t_2, \ldots t_n$ is optimal given the token sequence $w_1, w_2, \ldots, w_n$.
- We will write w_1^n for the sequence $w_1, w_2, \ldots, w_n$, etc. The hidden Markov model (HMM) is probabilistic, and identifies the optimal tag sequence with the most probable tag sequence, i.e. it tries to find the t_1^n which maximizes $P(t_1^n \mid w_1^n)$ which by Bayes' formula is the same as $\frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$
- HMM makes two assumptions:
  - That each word in the sequence only depends on its corresponding tag, $P(w^i|w_1^{(i-1)}t_1^n) \approx P(w_i|t_i)$.
  - That each tag only depends on its immediate preceding tag, $P(t_i|t_1^{(i-1)}) \approx P(t_i|t_{(i-1)})$ This defines a bigram HMM-tagger. In a trigram tagger, a tag may depend on its two preceding tags. In principle there might also be *n*-gram taggers for longer *n*-s.
- $\hat{t_1^n} = \mathbf{argmax}_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i)P(t_i \mid t_{i-1})$

1(c) Training and applying an HMM-tagger

We will train an HMM-tagger and use it to tag the sentence

1) *they gave her flowers*

As training data, we will use the Brown corpus with the universal POS-tag set. All the relevant counts for sentence (1) can be found in the enclosed pdf-file. How will the HMM-tagger go about to tag the sentence, and which tag sequence will it choose?

Max 10 points

```
Absolute frequencies of words
=============================
flowers     57
gave       285
her       3036
they      3620


Absolute frequencies of tags
============================
DET       137019
NOUN      275558
PRON       49334
VERB      182750
```

```
Absolute frequencies of words with their tags
=============================================
         DET NOUN PRON VERB
flowers    0   57    0    0
   gave    0    0    0  285
    her 1929    0 1107    0
   they    0    0 3620    0


Bigram tag frequencies
======================
       DET  NOUN  PRON  VERB  <\s>
 <s> 12238  8093  9157  2588     0
 DET   809 85838  1358  8861    18
NOUN  4270 41144  5460 43763   914
PRON   864   437   404 34838     5
VERB 29784 17819 10058 33667   102

For example, the tag NOUN is followed by the tag DET 4270 times.
```

Answer:

We see that all words in the sentence are unambiguous except *her*. There are two possible tag sequences

i) PRON VERB PRON NOUN
ii) PRON VERB DET NOUN

In choosing between the two tag sequences, we only have to consider the parts of the HMM-equation where they differ

i) P(PRON | VERB) * P(*her* | PRON) * P(NOUN | PRON) =
10,058/182,750 * 1,107/49,334 * 437/49,334
ii) P(DET | VERB) * P(*her* | DET) * P(NOUN | DET) =
29,784/182,750 * 1,927/137,019 * 85,838/137,019

We see that the expression (ii) is much bigger than expression (i). Hence, it will choose *DET*.

The probabilities are estimated from counting occurrences in the corpus, e.g.

P(PRON | VERB) = C(VERB PRON)/C(VERB) = 10,058/182,750
P(*her* | PRON) = C(*her* : PRON)/C(PRON) = 1,107/49,334

In theory, a HMM-tagger compares all possible tag sequences for a sentence. To make the task tractable, the tagger may use dynamic programming and the Viterbi algorithm.

2(a) Test sets

In the mandatory assignments, we used two different test sets, one during development, and another one for final testing? Why did not we use only one and the same test set?

Max 3 points

We tried several different settings for the classifiers and chose the one that performed best on the development test set. There is a chance of overfitting to the development test set, that the settings that were optimal on this set, is not optimal for other test sets. By doing a final evaluation on a separate test set, we get a more realistic impression of what we can expect with respect to results in general.

2(b) N-fold cross-validation

Explain the principles for *n*-fold cross-validation, e.g. 10-fold cross-validation.

Max 6 points

The development set is partitioned into *n* equally sized sets. One then performs *n* experiments, one for each of the subsets, where this subset serves as test set and the other *n-1* sets are merged into a training set.

2(c) Motivation

- Why does one apply cross-validation?
- What do you think are the reasons we used cross-validation in mandatory assignment 2A text classification, but not in mandatory assignment 2B tagging?

Max 6 points

Cross-valisdation is typically used when the development set is not very big. Then the development test set cannot be very big either, and by testing on small test sets, there will be large variations depending on how one splits the development set into training and testing.

In the test classification assignemnt, we considered 1800 items in the development set. The test set couldn't consist of more than a couple of hundred and when we ran cross-validation we saw a large variationin accuracy between the various runs. By using cross-validation we evaluate on larger test data, in the text classification task, we evaluate on 1800 items.

In the tagging experiment, however, we considered sets with many more items. We counted accuracy on tokens, and the whole Brown corpus contains in the order of 1 million tokens.

A pragmatic reason is that it would have taken too long to run cross-validation when training a logistic regression-based tagger on the whole Brown corpus.

3 Generative and discriminative

- What is the difference between a generative and a discriminative classifier?
- Which of the following are generative and which of them are discriminative?
  - Bernoulli Naive Bayes for text classification
  - Multinomial Naive Bayes for text classification
  - Logistic regression for text classification
  - An hidden Markov model (HMM) tagger
  - A maximum entropy (logistic regression) tagger

Max 5 points

A generative classifier estimates a probability distribution over all pairs of classes and items, i.e. P(c, x) for each class c and possible observation x. In addition it estimates a probability distribution over the classes and uses these two to estimate the conditional probability of the classes for the observations, P(c | x) = P(c, x)/P(x)
A discriminative classifier estimates P(c | x) directly.

Generative:

- Bernoulli Naive Bayes for text classification
- Multinomial Naive Bayes for text classification
- An hidden Markov model (HMM) tagger

Discriminative:

- Logistic regression for text classification
- A maximum entropy (logistic regression) tagger

4(a) Unlabeled scores

The goal of this exercise is to evaluate a named-entity recognizer. The pdf-document shows the results of a named-entity system trained and tested on the Spanish data from the conll2002 shared task, distributed with NLTK. The second column shows the tokens and the third column shows POS-tags. The POS-tags were used by the NER-system but are without interest for this exercise. The fourth column shows the gold IOB-tags and the fifth column shows the predicted IOB- tags. These two columns are the basis for the evaluation.

There are two ways to evaluate the NER-system, labeled or unlabeled. We will first consider the unlabeled scores. With the unlabeled score, one only evaluates whether the system have localized the named entity spans correctly, e.g. the span (15, 16) in sentence 554 is counted as a true positive. What are the unlabeled recall, precision and f-measure for the named entity spans? Explain how you find the numbers.

Max 10 points

```
Sentence nr: 525
  0   Gari                VMI   B-PER      B-PER
  1   Kasparov            AQ    I-PER      I-PER
  2   (                   Fpa   O          O
  3   RUS                 NC    B-ORG      B-LOC
  4   )                   Fpt   O          O
  5   4,5                 Z     O          O
  6   .3                  Z     O          O
  7   .                   Fp    O          O
Sentence nr: 528
  0   Michael             VMI   B-PER      B-PER
  1   Adams               AQ    I-PER      I-PER
  2   (                   Fpa   O          O
  3   ING                 NP    B-ORG      B-ORG
  4   )                   Fpt   O          O
  5   3,5                 Z     O          O
  6   .6                  Z     O          O
  7   .                   Fp    O          O
Sentence nr: 554
  0   En                  SP    O          O
  1   los                 DA    O          O
  2   próximos            AQ    O          O
  3   meses               NC    O          O
  4   serán               VSI   O          O
  5   concedidos          VMP   O          O
  6   los                 DA    O          O
  7   cinco               DN    O          O
  8   premios             NC    O          O
  9   restantes           AQ    O          O
 10   ,                   Fc    O          O
 11   Letras              NC    B-MISC     O
 12   ,                   Fc    O          O
 13   Artes               NC    B-MISC     B-PER
 14   ,                   Fc    O          O
 15   Cooperación         NC    B-MISC     B-PER
 16   Internacional       AQ    I-MISC     I-PER
 17   ,                   Fc    O          O
 18   Concordia           NC    B-MISC     O
 19   y                   CC    O          O
 20   Deportes            NC    B-MISC     O
 21   .                   Fp    O          O
Sentence nr: 668
  0   El                  DA    O          O
  1   Madrid              NC    B-ORG      B-LOC
  2   se                  P0    O          O
  3   refugia             VMI   O          O
  4   en                  SP    O          O
  5   Versalles           NC    B-LOC      O
  6   .                   Fp    O          O
```

We first count the true positives (TP), the false positives (FP), and the false negatives (FN). Some examples. In the first example (sentence 525) Both (0,1): 'Gary Kasparov' and (3,3): 'RUS' are TP. In sentence 554, there are 3 FN, (11, 11): 'Letras', (18, 18): 'Concordia' and (20, 20): 'Deportes', while there are 2 TPs. All in all, we see

Sentence	TP	FP	FN
525	2		
528	2		
554	2		3
668	1		1
Sum	7		4

This gives the measures

R(ecall) = TP/(TP+FN) = 7/11
P(recision) = TP/(TP+FP) = 1
F-score = 2*R*P/(P+R) = (2*7/11)/(7/11 + 1) = 14/18 = 7/9

4(b) Labeled scores

We will calculate the labeled scores from the same four example sentences. With labeled scores, the span (15, 16) in sentence 554 is not counted as a true positive since it gets different labels in the gold set and the predicted set. Count the true positives, false positives and false negatives for each of the four classes of named entities, and report in a table!

Calculate the precision, recall and f-measure for each of the four classes and explain how you find them.

Max 9 points

Sentence	PER			ORG			LOC			MISC		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
525	1					1		1				
528	1			1								
554		2										5
668						1		1	1			
Sum	2	2		1		2		2	1			5

PER:
R(ecall) = TP/(TP+FN) = 1
P(recision) = TP/(TP+FP) = 1/2
F-score = 2*R*P/(P+R) = (2*1/2)/(1 + 1/2) = 2/3

ORG:
R(ecall) = TP/(TP+FN) = 1/3
P(recision) = TP/(TP+FP) = 1
F-score = 2*R*P/(P+R) = (2*1/3)/(1 + 1/3) = 2/4 = 1/2

LOC
R(ecall) = TP/(TP+FN) = 0
P(recision) = TP/(TP+FP) = 0
F-score = 2*R*P/(P+R) = 0

MISC
R(ecall) = TP/(TP+FN) = 0
P(recision) = TP/(TP+FP) = 0
F-score = 2*R*P/(P+R) = 0

4(c) Micro and macro scores

Calculate the macro and micro recall, precision and f-measure across the four classes and show how you find them!

Max 6 points

For the macro-scores we take the average over the 4 classes

Macro-R = (PER-R + ORG-R + LOC-R + MISC-R)/4 = (1 + 1/3 + 0 + 0)/4 = 1/3
Macro-P = (1/2 + 1 + 0 + 0)/4 = 3/8
Macro-F = (2/3 + ½ + 0 + 0)/4 = 7/24

For the Micro-scores we accumulate the numbers for TP, FP, FN over the 4 classes

Sentence	PER		
	TP	FP	FN
PER	2	2	
ORG	1		2
LOC		2	1
MISC			5
Pooled	3	4	8

R(ecall) = TP/(TP+FN) = 3/11
P(recision) = TP/(TP+FP) = 3/7
F-score = 2*R*P/(P+R) = (2* 3/11 * 3/7)/(3/11 + 3/7) = 18/54 = 1/3

5(a) Fundamental frequency

What is the fundamental frequency (F0) in acoustics? And why might it be useful for a spoken dialogue system to measure it?

Max 5 points

1) the fundamental frequency F_0 is lowest frequency of the sound wave
2) it corresponds to the speed of vibration of the vocal folds
3= the fundamental frequency correlates with the pitch, which gives us the intonation of the speech signal. And intonation is important for a spoken dialogue system - for instance, a rising intonation at the end of a signal will indicate an interrogative utterance in many language

5(c) MDP

1) What is a Markov Decision Process (MDP)? Give a formal definition.
2) How can MDPs be used in dialogue management?

Max 5 points

Markov Decision Processes

► A MDP is defined as a tuple **<S,A,T,R>**, where:

- **S** is the *state space* (space of possible states in the domain)
- **A** is the *action space* (possible actions for the agent)
- **T** is the *transition function*, defined as T(s, a, s') = P(s'|s, a). It is the probability of arriving to state **s'** after executing action **a** in state **s**.
- **R** is the *reward function*, defined as R : S × A → R. It is a real number encoding the utility for the agent to perform action **a** while in state **s**.

We can use MDP to represent dialogue policies, which are mappings between dialogue states to system actions. The idea is that:

the state represent the current dialogue state (for instance as slot-values)
the action space represent the possible actions of the dialogue system
the transition function represent the dialogue dynamics
and the rewards encode what we want the dialogue system to achieve, for instance satisfy a user request, minimise the number of turns, etc.
Given this formalisation, one can apply various reinforcement learning methods to automatically optimise dialogue policies from experience (which may be real dialogues or simulated ones).

5(b) **IR-based chatbots**

Assume you wish to develop an IR-based chatbot. To make simple, you decide to use cosine similarity over TF-IDF-weighted vectors. You collect a small corpus of 10 utterances:

1 *Hi Charles!*
2 *Hello Elsa!*
3 *How are you?*
4 *Fine, and you?*
5 *A bit tired.*
6 *Why?*
7 *Busy at work.*
8 *But you are doing great!*
9 *Thanks.*
10 *See you later!*

1) Compute the TF-IDF values for the tokens in the utterances 3, 4 and 8. Don't forget to include the punctuation in your computations.

2) Now assume that you are given the following user input:
How are you doing?

What will be the answer selected by the chatbot trained on the corpus above? Describe your calculation steps.

Max 15 points

Question 1: TF-IDF values:

utterance 3:
- "How": log (10/1) = 1 (assuming a base 10)
- "are": log(10/2)
- "you": log (10/4)
- "?": log(10/3)

utterance 4:
- "Fine": log(10/1)
- ",": log(10/1)
- "and": log(10/1)
- "you": log(10/4)
- "?": log(10/3)

utterance 8:
- "But": log(10/1)
- "you": log(10/4)
- "are": log(10/2)
- "doing": log(10/1)
- great": log(10/1)
- "!": log(10/4)

Note: I assume here a simple version of TF that simply express the number of times the token appears in the utterance (and since all tokens appear once, the TF is then 1). One can also rely on definition of TFs that divide the counts by the total number of tokens in the document.

Question 2: Response selection

The formula to use is the following

1. Return the response to the most similar turn: Given user query q and a conversational corpus C, find the turn t in C that is most similar to q (for example has the highest cosine with q) and return the *following* turn, i.e. the human response to t in C:

$$r = response\left(\underset{t \in C}{\operatorname{argmax}} \frac{q^T t}{||q||t||}\right) \tag{26.1}$$

We start by looking at the IDF values for the query:

- "How": log(10/1)
- "are": log(10/2)
- "you": log(10/4)
- "doing": log(10/1)
- "?": log(10/3)

The euclidian norm $\|q\|$ is $\sqrt{1^2 + (1-\log 2)^2 + (1-\log 4)^2 + 1^2 + (1-\log 3)^2)} = 1.709$ (although it doesn't really matter to search for the most relevant utterance)

And we can then compute the cosine similarity for the three sentences:

- utterance 3:
  - $q^T t = 1^2 + (1-\log 2)^2 + (1-\log 4)^2 + (1-\log 3)^2 = 1.920$
  - $\|t\| = \sqrt{1^2 + (1-\log 2)^2 + (1-\log 4)^2 + (1-\log 3)^2} = 1.386$
  - which means the cosine similarity is $\frac{1.920}{1.386 * 1.709} = 0.811$
- utterance 4
  - $q^T t = (1-\log 4)^2 + (1-\log 3)^2 = 0.432$
  - $\|t\| = \sqrt{1^2 + 1^2 + 1^2 + (1-\log 4)^2 + (1-\log 3)^2} = 1.852$
  - which means the cosine similarity is $\frac{0.432}{1.852 * 1.709} = 0.136$
- utterance 8:
  - $q^T t = (1-\log 4)^2 + (1-\log 2)^2 + 1^2 = 1.647$
  - $\|t\| = \sqrt{1^2 + (1-\log 4)^2 + (1-\log 2)^2 + 1^2 + 1^2 + (1-\log 4)^2} = 1.950$
  - which means the cosine similarity is $\frac{1.647}{1.950 * 1.709} = 0.494$

In other words, utterance 3 is the most similar to the query. Based on the mechanism described above, the chatbot will therefore select the utterance coming after utterance 3, which is "Fine, and you?".

Important note: I do not expect the students to necessarily arrive at the exact same numerical values (as their answer may vary according to the TF formula they decide to use, the tokenisation scheme, and the base of the logarithm). What is most important is that they show an understanding of the main concepts behind TF-IDF and IR-based chatbots.