EDAN20
Final Examination

Pierre Nugues

October 29, 2018

The examination is worth 255 points. The distribution of points is indicated with the questions. You need 40% to have a mark of 4 and 60% to have a 5.

1 Closed Book Part: Questions

In this part, no document is allowed. It is worth 118 points.

Chapter 1. Cite three applications that use natural language processing to some extent. 3 points

Chapter 1. Annotate each word of the sentence:

EU leaders shelve next month's special summit[1]

with its part of speech. You will use parts of speech you learned at school: common noun, proper noun, verb, and adjective. There are two adjectives, four nouns, and one verb. You will annotate the "'s" suffix as a possessive marker (POS). 3 points

Chapter 1. Draw the syntactic graph of the sentence:

EU leaders shelve next month's special summit

using dependency relations. You will first identify the main verb, then, the subject, the object, and a time modifier inside the object. You can first use groups of two words, for instance *EU leaders* instead of *EU* and *leaders*. Your graph should show what word the time modifier modifies. Then you will determine what is the head inside these groups, for instance is *EU* or *leader* the head of *EU leaders*? 5 points

Chapter 1. Identify the proper noun (or named entity) in the sentence:

EU leaders shelve next month's special summit

and describe what entity linking is on this example and why it can be difficult. 8 points

Chapter 1. Represent the sentence:

[1]Retrieved on October 18, 2018 from `www.ft.com`

EU leaders shelve next month's special summit

with a predicate–argument structure. You should identify one predicate (the verb) and two arguments: `predicate(arg1, arg2)`. 4 points

Chapter 2. Describe what a concordance is and give all the **case-insensitive** concordances of the string *skad* with one word before and one word after in the text below (excluding the title): 3 points

Fler skadades på byggen i somras

Antalet anmälda skador i samband med arbetsplatsolyckor inom byggbranschen ökade med omkring tio procent i somras, jämfört med motsvarande period förra året, rapporterar Ekot i Sveriges Radio.

Under juni, juli och augusti anmäldes 802 allvarliga olyckor på byggarbetsplatser till Arbetsmiljöverket. Det handlar bland annat om fallskador, personer som skadades med verktyg och belastningsskador. Samtliga anmälda olyckor medförde sjukskrivningar. Under samma period i fjol anmäldes 725 olyckor.

– Det är allvarligt överhuvudtaget att det är så höga siffror på antalet skadade inom byggbranschen, säger Tomas Blombäck, regionchef på Arbetsmiljöverket region Öst, till Ekot.

Sydsvenskan.se, Retrieved October 18, 2018. Author TT and Maria Repitsch

Chapter 2. Identify what the regular expressions in the list below match in the text above (identify all the matches and just write one word before and after, or write no match if there is no match): 15 points

List of case-sensitive regular expressions:

1. `skador`
2. `\sskador\s`
3. `skad(or)+`
4. `skad(or)?`
5. `olyck\p{L}+`
6. `[a-z]olyck\p{L}+`
7. `[^a-z]olyck\p{L}+`
8. `olyckor.`
9. `olyckor\.`
10. `(.)(.)\2\1`

Chapter 2. What will be the output of the command: 3 points

```
tr -cs '0-9' '\n' <text
```

when applied to the text above (*Fler skadades på byggen i somras...*).

Chapter 2. What will be the output of the command: 3 points

```
tr -cs '0-9' '\n' <text | sort -n
```

when applied to the text above (*Fler skadades på byggen i somras...*).

Chapter 2. Write a simple regular expression that matches all the words of a text: lower and uppercase. 1 point

You will use a Unicode regular expression of the form `\p{}`.

Chapter 2. Write a regular expression that matches alternate sequences of three identical letters in a text, for instance in the text:

```
Fler skadades aaabbbaaabbb på byggen i somras
```

your regular expression should identify `aaabbbaaabbb`. 8 points

You will proceed in three steps:

1. Write a regular expression, where you match sequences of three identical characters like `bbb`. You will use backrefences, `\nn`, where `nn` is a number. As it will contain a backslash, you must use the `r` prefix as in `r'my_regex'` so that Python does not interpret it as a number.
2. Write a second regular expression, where you extend your regular expression so that it can match alternate sequences of three identical characters, like `xxxyyyxxxyyy`.
3. In a third step, extend your regular expression so that the number of triples of each sequence do not need to be identical, like for instance: `xxxyyyxxx` (two `xxx` and one `yyy`) and `xxxyyyxxxyyy` (two `xxx` and two `yyy`).

With the program:

```
import regex as re

regex = # your regex
string1 = 'Fler skadades aaabbbaaa på byggen i somras xxxyyyxxx'
string2 = 'Fler skadades aaabbb på byggen i somras dddcccdddccc'

for string in [string1, string2]:
    matches = [match.group() for match in re.finditer(regex, string)]
    print(matches)
```

your `regex` should print:

```
['aaabbbaaa', 'xxxyyyxxx']
['aaabbb', 'dddcccdddccc']
```

Chapter 2. In this exercise, you will compute the edit distance between *table* and *cables*. You will use three edit operations: deletions, substitutions, and insertion, and you will represent the computation using a matrix.

s	6	7	6			
e	5	6	5			
l	4	5	4	3		
b	3	4	3	2	3	
a	2	3	2	3	4	5
c	1	2	3	4	5	6
start	0	1	2	3	4	5
	start	t	a	b	l	e

Figure 1: Edit operations

$$\begin{tikzcd}
i-1, j \arrow[r, "delete"] & i, j \\
i-1, j-1 \arrow[ur, "replace"] & i, j-1 \arrow[u, "insert"']
\end{tikzcd}$$

1. The algorithm to compute the edit distance is given by this recursive relation: 3 points

$$edit\_distance(i,j) = \min \left(\begin{array}{l} edit\_distance(i-1,j) + del\_cost \\ edit\_distance(i-1,j-1) + subst\_cost \\ edit\_distance(i,j-1) + ins\_cost \end{array} \right).$$

   What are the typical costs associated with deletions, substitutions, insertion, and copies;

2. The cells in Table 1 are filled from adjacent left and lower cells with the edit operations. Explain how to use Fig. 1 to apply these edit operations; 3 points

3. Fill the remaining empty cells in Table 1. You will copy the whole table on your exam sheet or use the one in the Appendix; 6 points

Chapter 3. What do the `\p{Latin}+`, `\p{Cyrillic}+`, and `\p{Arabic}+` Unicode regular expressions match? Please, provide one answer for each regular expression. 3 points

Chapter 4. Describe what a supervised classifier is. What is the typical notation of the input matrix of a classifier (predictors) and of the output vector (response). Give the name of two classifiers: The one you used for the assignments and a second one. 4 points

Chapter 5. Give the probabilistic language model of the sentence:

Fler skadades på byggen i somras

using no n-gram approximation. You will ignore possible start and end of sentence symbols. 2 points

Chapter 5. Using a bigram approximation, give the probabilistic language model of the sentence:

Fler skadades på byggen i somras

You will ignore possible start and end of sentence symbols. 2 points

Chapter 5. Using a trigram approximation, give the probabilistic model of the sentence:

Fler skadades på byggen i somras

You should have exactly the same number of terms as in the previous question. You can include a start of sentence symbol or not. 2 points

Chapter 5. Using the counts in Table 1, you will compute the probability of the sentence:

Fler skadades på byggen i somras

using a bigram and a trigram approximations. You will use fractions to represent the terms in the product and you will not try to reduce them, *i.e.* you will write $\frac{1}{3}$ and not 0.33. 8 points

The count of *fler* in the `.se` web domain is 103,000,000. To compute its probability, you would need an estimate of the total number of words in the Swedish web:

$$P(\text{fler}) = \frac{Count(\text{fler})}{\#\text{words in the Swedish web}}.$$

You will use the Selma corpus instead that consists of about one million words and where you have 32 *fler*.

Table 1: Bigram and trigram counts retrieved from Google.com on October 22, 2018 with the search limited to the Swedish domain (`site:se`).

Bigrams	**Bigram counts**	**Trigrams**	**Trigram counts**
fler skadades	16 900		
skadades på	31 200	fler skadades på	13 700
på byggen	37 900	skadades på byggen	13 200
byggen i	37 900	på byggen i	17 000
i somras	3 640 000	byggen i somras	17 200

Chapter 8. In this exercise, you will analyze the annotation of named entities. Table 2 shows a sentence from the CoNLL 2003 corpus of English (Tjong Kim Sang and De Meulder, 2003). It contains annotations for the parts of speech, noun groups, verb groups, and prepositions (the chunks), as well as named entities.

The chunk annotation in Table 2 uses the IOB tag set, where I stands for inside, O, for outside, and B for between. We use this B to start of a new chunk when two adjacent chunks are of the same category. This annotation is different from the IOB2 tag set that we used in the chunking laboratory, where B stands for begin (the beginning of a chunk).

1. Describe in which way the annotations IOB (the one in Table 2) and IOB2 (the one you used in the lab 3) are different and use the example of

   $\boxed{\text{South African provincial side}}\ \boxed{\text{Boland}}$

   to contrast them. 2 points
2. Write the sentence in Table 2 on your exam sheet and draw a box around each chunk. Label each box with its category: NP, VP, or PP. You will ignore the outside words (O). 2 points
3. Write the sentence in Table 2 and draw box around each named entity. You will label the boxes with their category. 2 points

Table 2: An excerpt from the CoNLL 2003 corpus of English using the IOB tagset(Tjong Kim Sang and De Meulder, 2003).

Words	**Parts of speech**	**Chunks**	**Named entities**
South	JJ	I-NP	I-MISC
African	JJ	I-NP	I-MISC
provincial	JJ	I-NP	O
side	NN	I-NP	O
Boland	NNP	B-NP	I-ORG
said	VBD	I-VP	O
on	IN	I-PP	O
Thursday	NNP	I-NP	O
they	PRP	B-NP	O
had	VBD	I-VP	O
signed	VBN	I-VP	O
Leicestershire	NNP	I-NP	I-ORG
fast	JJ	I-NP	O
bowler	NN	I-NP	O
David	NNP	I-NP	I-PER
Millns	NNP	I-NP	I-PER
on	IN	I-PP	O
a	DT	I-NP	O
one	CD	I-NP	O
year	NN	I-NP	O
contract	NN	I-NP	O
.	.	O	O

Chapter 8. In this exercise, you will describe how to build a simple named entity recognizer using a linear classifier. This exercise is very similar to the laboratory on chunking.

1. Describe what the training step is. Describe how you would use the data in Table 2 to build the input matrix $\mathbf{X}$ (predictors) and output $\mathbf{y}$ (response); 3 points

2. Build this input matrix using features consisting of the current word, the previous word, and the next word, the current part of speech, the previous part of speech, and the next part of speech. You will write the four first lines of the matrix and output vector; 3 points
3. Describe what the test step is and what you would use as data; 3 points
4. Name the two scikit-learn methods you would use to carry out these two steps (training a model and predicting a vector); 2 points
5. The training step only accepts numerical matrices. Describe what a vectorization (or one-hot encoding) is and how you would transform a matrix of symbols (strings) into a matrix of numbers. Name the scikit-learn class and methods for it. 5 points
6. Give the definition of the precision and recall metrics to measure the performance of your recognizer. 2 points

Chapter 14. Table 3 shows two sentences from the CoNLL 2011 corpus (Pradhan et al., 2011). The Chain column contains the coreference chains of entities. Give the coreference chains 0, 6, 8, 15, and 23 appearing in these two sentences as well as their list of mentions. You will use the notation:

$$\begin{aligned}
CorefChain(0) &= \{Mention1, Mention2, ...\},\\
CorefChain(6) &= \{Mention1, Mention2, ...\},\\
CorefChain(8) &= \{Mention1, Mention2, ...\},\\
CorefChain(15) &= \{Mention1, Mention2, ...\},\\
CorefChain(23) &= \{Mention1, Mention2, ...\},
\end{aligned}$$

You will ignore chain 16. 5 points

Table 3: Simplified annotation of two sentences in the CoNLL 2011 corpus. After Pradhan et al. (2011).

Document	Inx	Word	POS	Parse bit	Type	Chain
wsj_0771	0	“	“	(TOP(S(S*	*	-
wsj_0771	1	Vandenberg	NNP	(NP*	(PERSON)	(8|(0)
wsj_0771	2	and	CC	*	*	-
wsj_0771	3	Rayburn	NNP	*)	(PERSON)	(23)|8)
wsj_0771	4	are	VBP	(VP*	*	-
wsj_0771	5	heroes	NNS	(NP(NP*)	*	-
wsj_0771	6	of	IN	(PP*	*	-
wsj_0771	7	mine	NN	(NP*))))	*	(15)
wsj_0771	8	,	,	*	*	-
wsj_0771	9	”	”	*)	*	-
wsj_0771	10	Mr.	NNP	(NP*	*	(15
wsj_0771	11	Boren	NNP	*)	(PERSON)	15)
wsj_0771	12	says	VBZ	(VP*	*	-
wsj_0771	13	,	,	*	*	-
wsj_0771	14	referring	VBG	(S(VP*	*	-
wsj_0771	15	as	RB	(ADVP*	*	-
wsj_0771	16	well	RB	*)	*	-
wsj_0771	17	to	IN	(PP*	*	-
wsj_0771	18	Sam	NNP	(NP(NP*	(PERSON*	(23
wsj_0771	19	Rayburn	NNP	*)	*)	-
wsj_0771	20	,	,	*	*	-
wsj_0771	21	the	DT	(NP(NP*	*	-
wsj_0771	22	Democratic	JJ	*	(NORP)	-
wsj_0771	23	House	NNP	*	(ORG)	-
wsj_0771	24	speaker	NN	*)	*	-
wsj_0771	25	who	WP	(SBAR(WHNP*)	*	-
wsj_0771	26	cooperated	VBD	(S(VP*	*	-
wsj_0771	27	with	IN	(PP*	*	-
wsj_0771	28	President	NNP	(NP*	*	-
wsj_0771	29	Eisenhower	NNP	*))))))))))	(PERSON)	23)
wsj_0771	30	.	.	*))	*	-
wsj_0771	0	“	“	(TOP(S*	*	-
wsj_0771	1	They	PRP	(NP*)	*	(8)
wsj_0771	2	allowed	VBD	(VP*	*	-
wsj_0771	3	this	DT	(S(NP*	*	(6
wsj_0771	4	country	NN	*)	*	6)
wsj_0771	5	to	TO	(VP*	*	-
wsj_0771	6	be	VB	(VP*	*	(16)
wsj_0771	7	credible	JJ	(ADJP*)))))	*	-
wsj_0771	8	.	.	*))	*	-

2 Problem

In this part, documents are allowed. It is worth 137 points.

Named entity linking (NEL) is the process of recognizing names of things, such as persons, locations, organization, in a text and linking them to unique identifiers. In this part, you will analyze and implement a simplified version of a NEL system created by Yang et al. (2017) for an evaluation organized by the U.S. National Institute of Standards and Technology (NIST) (Ji et al., 2017).

The system to analyze is called TAI for Tencent AI lab and the evaluation is called EDL-TAC for entity discovery and linking at the text analysis conference. The second paper by Ji et al. (2017) is only provided for information and will not be used for programming.

Named entity linking usually consists of two steps:

1. The first step recognizes mentions of named entities. This step is similar to chunking that you carried out in the third assignment of the course. In Fig. 2, this corresponds to the detection of *William Shakespeare* and *Stratford-upon-Avon* in the sentence:

   William Shakespeare was born and brought up in Stratford-upon-Avon.

2. The second step identifies the "reals things" behind these mentions. For each real thing, we need a unique identifier stored in a database. In Fig. 2, we could link *William Shakespeare* to a Wikidata identifier: Q692 and *Stratford-upon-Avon* to Q189288. At Lund University, we would use a Swedish personal number so that student Shakespeare, William could register to this examination.

As programming language, you can use Python (strongly preferred), Java, Perl, or Prolog. You will focus on the program structure and not on the syntactic details. You can ignore the Python modules, Java packages or imports for instance.

2.1 Analyzing the Papers

Read the *Abstract*, *Introduction*, and *Overview* sections of Yang et al. (2017) and the *Tri-lingual EDL Task* section of Ji et al. (2017), including Table 2 in their paper.

1. Describe the overall structure of the system by Yang et al. (2017). You can just reformulate their text; 4 points

2. Describe what are the NIL entities; referring to wikipedia, give an example of what could be a NIL entity; 4 points

3. The organizers asked the participants not to make their rankings public and avoid derogatory comparisons. Analyzing Table 6 in Ji et al. (2017) and Table 2 in Yang et al. (2017) as well as the last figure of their abstract, guess the rank of TAI system. You can look at the NERLC column (strong typed all match). There is a possible glitch in the figures for English. 4 points

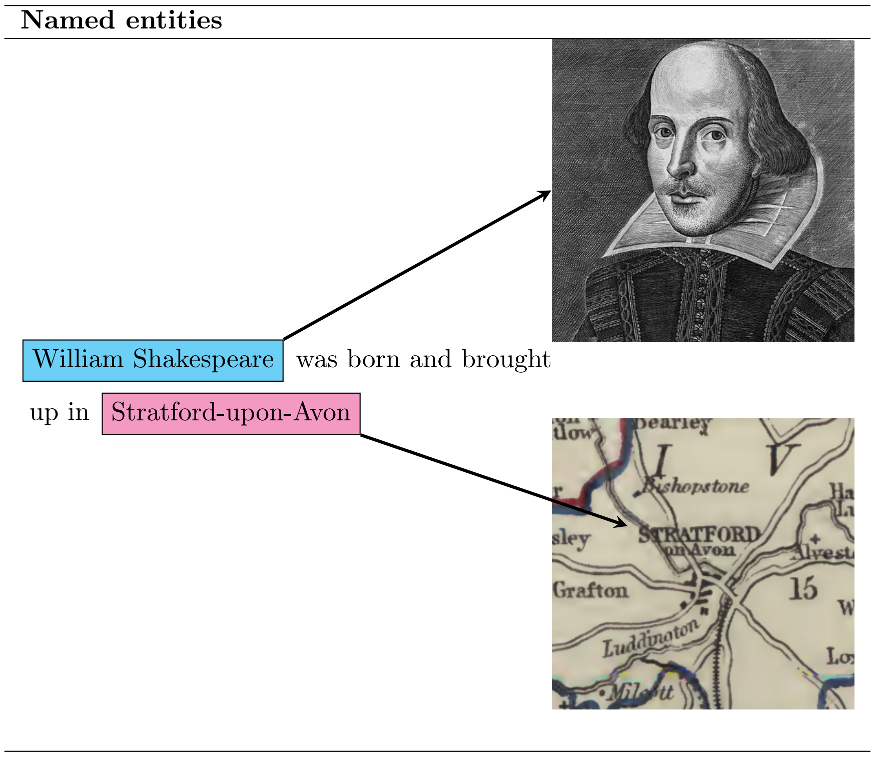


Figure 2: Named entities: entities that we can identify by their names. Portrait: credits Wikipedia. Map: Samuel Lewis, *Atlas to the topographical dictionaries of England and Wales*, 1848, credits: archive.org

2.2 Understanding the Corpus

The NIST provided the corpus in the form of XML documents containing the texts to analyze and one file containing the annotations.

The text below is an excerpt of the EDL-TAC corpus in English and Table 4 shows the annotations, where they marked the named entities, their index in the text, and for each entity, its Freebase identifier and its type. Freebase is a database of entities used by Google and similar to Wikidata.

```
<DOC    id="ENG_NW_001278_20130109_F00011TB4">
<DATE_TIME>2013-01-09T12:19:34</DATE_TIME>
<HEADLINE>
S. Korea's opposition party elects interim leader
</HEADLINE>
<AUTHOR>Kim Junghyun</AUTHOR>
<TEXT>
S. Korea's opposition party elects interim leader

S. Korea's opposition party elects interim leader

SEOUL, Jan. 9 (Xinhua) -- South Korea's main opposition Democratic
```

Table 4: The annotation of the XML text

Entity	Index span	Identifier	Cat	Type
S. Korea	101-108	m.06qd3	GPE	NAM
party	123-127	m.0hzrnfb	ORG	NOM
Kim Junghyun	171-182	NIL01133	PER	NAM
S. Korea	200-207	m.06qd3	GPE	NAM
party	222-226	m.0hzrnfb	ORG	NOM
S. Korea	251-258	m.06qd3	GPE	NAM
party	273-277	m.0hzrnfb	ORG	NOM
SEOUL	302-306	m.0hsqf	GPE	NAM
Xinhua	317-322	m.01n13b	ORG	NAM
South Korea	328-338	m.06qd3	GPE	NAM
Democratic United Party	358-380	m.0hzrnfb	ORG	NAM
leader	417-422	NIL00095	PER	NOM
party	436-440	m.0hzrnfb	ORG	NOM
Moon Hee-sang	516-528	NIL00095	PER	NAM
lawmaker	543-550	NIL00095	PER	NOM
Roh Moo-hyun	599-610	m.08xh6c	PER	NAM
party	649-653	m.0hzrnfb	ORG	NOM
committee	667-675	NIL00767	ORG	NOM
old	691-693	NIL00095	PER	NOM
moderate	698-705	NIL00095	PER	NOM
Lee Hae-chan	769-780	m.0c02l0	PER	NAM
party	892-896	m.0hzrnfb	ORG	NOM
Park Ki-choon	908-920	NIL00012	PER	NAM
leader	933-938	NIL00012	PER	NOM
Park Jie-won	946-957	m.0j_5tn_	PER	NAM

```
United Party on Wednesday elected a new interim leader to head the
party reeling from the narrow defeat in the presidential election
a month ago.

Moon Hee-sang, a five-term lawmaker who served as chief of staff
to late President Roh Moo-hyun, was elected to lead the
center-left party's emergency committee.

The 67-year-old, a moderate without factional affiliations,
is filling the vacuum left by Lee Hae-chan, who was pressured
into resigning by calls for political reform before the
Dec. 19 election.

Last month, the party nominated Park Ki-choon as a floor leader
after Park Jie-won stepped down over the election loss.
</TEXT>
</DOC>
```

1. Name the entity with the NIL00095 identifier. (Use his name); 2 points

Table 5: An excerpt of the TAC corpus in the CoNLL format

Word	Tag IOB	Tag BIEOS
SEOUL	I-GPE-NAM	S-GPE-NAM
Jan.	O	O
9	O	O
Xinhua		
South		
Korea		
's		
main		
opposition		
Democratic		
United		
Party		
on		
Wednesday		
elected		
a		
new		
interim		
leader		

2. Check that the coreference chain of NIL00095 is correct and justify in a few words why the mentions refer to a same person. 2 points

3. The name (NAM) and nominal (NOM) types designate entities that are referred to, respectively, by their name or by a nominal mention. What is the name of entity NIL00012 and his nominal mention? 2 points

4. List all the nominal mentions of NIL00095? 2 points

When available, the entity is labeled with an identifier from Freebase, for instance, `m.06qd3` is for South Korea and would correspond to Q884 in Wikidata.

2.3 Mention Detection

2.3.1 From TAC to CoNLL

In the assignments, you used the CoNLL format. You will manually annotate the named entities of a fragment of the TAC corpus with this format. You will first use the IOB scheme as in Table 2, where B mean between, and then a second scheme called BIEOS. Table 5 shows this fragment, where you will use the categories and type as tag suffix, for instance I-GPE-NAM or I-PER-NOM.

Understanding the Annotation.

1. Using the entities marked in Table 4, complete the IOB annotation of the fragment in Table 5 (You can use the copy of the table in the Appendix).

For the annotation of *South*, you will only consider the CoNLL table, and ignore the dashes in the XML text. 4 points

2. The tagging performance depends on the annotation. Results are different when using IOB and IOB2, for instance. Yang et al. (2017) used the BIEOS scheme[2] that proved superior, where:

   - S marks a mention that consists of a single word;
   - B stands for the beginning of a mention when this mention has two words or more;
   - I, for inside, when this mention has three words or more;
   - E, for end, the last word of a mention, when this mention has two words or more;
   - O for outside.

   Complete the BIEOS annotation of the fragment in Table 5, third column (You can use the copy of the table in the Appendix). 6 points

3. Try to justify why BIEOS gives better results than IOB or IOB2. 3 points

Converting the Annotation. In this part, you will loop over all the sentences and all the words in a sentence and apply rules to convert the IOB annotation into BIEOS. You rules will consider the context of the current word: The word before and the word after.

1. Table 6 shows the tag conversion. It can be written using six rules.

   (a) Convert from I to B: If ...

   (b) Convert from I to I: If ...

   (c) Convert from I to E: If ...

   (d) etc.

   Justify why `O` is always converted to `O`; 2 points

2. Justify why there is no `B-X` → `I-X` and `B-X` → `E-X` conversions (why these conversions are impossible); 4 points

3. Let us consider the conversion of the `I-X` tags into `B-X`. The last word of a sentence, if annotated with `I-X`, will never be converted into `B-X`. Why?

4. Write the rule converting the `I-X` tags into `B-X` using your own words. This rule can be expressed using conditions based on the tags before and after the current tag. You can break the rule considering two cases: This is the first word of the sentence or a word inside the sentence. Think in terms of tag suffixes. 6 points

[2]This scheme is also called BILOU, for Begin (B), Inside (I), Last (E), Outside (O), and Unique (S).

Table 6: Conversion table

→	B	I	E	O	S
I				X	
O	X	X	X		X
B		X	X	X	

2.3.2 Programming

In this section, you will program a partial converter from IOB to BIEOS. The complete top-level converter, `iob2bieos()`, considers all the cases:

```
# We call the converters: I -> S, I -> B, I -> E, and B -> S
def iob2bieos(sentence_iob):
    sentence_bieos = copy.deepcopy(sentence_iob)
    for tag in ['chunk', 'ner']:
        sentence_bieos = I2S(sentence_iob, sentence_bieos, tag)
        sentence_bieos = I2B(sentence_iob, sentence_bieos, tag)
        sentence_bieos = I2E(sentence_iob, sentence_bieos, tag)
        sentence_bieos = B2S(sentence_iob, sentence_bieos, tag)
    return sentence_bieos
```

You will restrict yourself to the `I-X` into `B-X` case. You will suppose that your corpus is available as a list of lists, where each list corresponds to a sentence and each word in this list is a dictionary with the CoNLL columns. The sentence in Table 2, first part of the exam, will be stored as:

```
[{'form': 'South', 'pos': 'JJ', 'chunk': 'I-NP', 'ner': 'I-MISC'},
{'form': 'African', 'pos': 'JJ', 'chunk': 'I-NP', 'ner': 'I-MISC'},
{'form': 'provincial', 'pos': 'JJ', 'chunk': 'I-NP', 'ner': 'O'},
{'form': 'side', 'pos': 'NN', 'chunk': 'I-NP', 'ner': 'O'},
{'form': 'Boland', 'pos': 'NNP', 'chunk': 'B-NP', 'ner': 'I-ORG'},
...
]
```

in a variable called `train_dict`. You will loop over the corpus with:

```
# The loop over the corpus
train_dict_bieos = []
    for sentence in train_dict:
        new_sentence = iob2bieos(sentence)
        train_dict_bieos.append(new_sentence)
```

1. Write the function that converts the IOB tags of a sentence in BIEOS tags for both the chunks and the named entity tags restricted to the `I-X` → `B-X` case: *i.e.* you will only write an `I2B()` function.

   This function consists of a loop over the words and one or more `if` statements with `and` and `or` connectors. Replace the `<FILL IN CODE>` placeholder with your code in the function below. 10 points

```
def I2B(sentence_iob, sentence_bieos, tag='ner'):
    if len(sentence_iob) == 1:
        return sentence_bieos

  <FILL IN CODE>
  return sentence_bieos
```

2.3.3 Understanding the Architecture of the Named Entity Recognition Module

Yang et al. (2017) used a neural network architecture to recognize named entities that you will replicate with a simpler logistic regression classifier.

1. Read Sect. 3 and tell the name of the classifier the authors used? 2 points
2. In the third assignment, you applied logistic regression from left to right. Is it the case for their classifier? Describe its structure. 4 points
3. As features Yang et al. (2017) used the words, the characters, the parts of speech, and NER tags obtained from another tagger (coreNLP). Instead of using the words as is, the authors applied a dimension reduction, similar to a principal component analysis, to the one-hot encoded words so that one-hot vectors are reduced to a dimension of 100. Why do you think this dimension reduction is interesting? 3 points
4. On top of this classifier, the authors used conditional random fields that take into account the tag transitions and try to optimize them. Why do you think this is interesting? 3 points

2.3.4 Programming

In this section, you will program a named entity recognizer using a stacked logistic regression. You will train two classifiers: The first one will proceed from left to right and the second one, from right to left. You will use a window of three words centered on the current word. Your features will be the word forms and the previously assigned tag:

$$\begin{aligned}\mathbf{x} &= (w_{-1}, w_0, w_1, ner_{-1})\\ y &= ner_0\end{aligned}$$

You will use padding symbols for the line before the first word and the line after the last word.

1. Write the three first lines of the $\mathbf{X_{lr}}$ matrix and $\mathbf{y}_{lr}$ vector using the data in Table 5, when proceeding from **left to right**; 3 points
2. Write the three first lines of the $\mathbf{X_{rl}}$ matrix and $\mathbf{y}_{rl}$ vector using the data in Table 5, when proceeding from **right to left**; 3 points
3. You will suppose the corpus with the BIOES annotation is stored in a variable called: `train_dict_bieos`.

   Write an `extract_features()` function to create the $\mathbf{X}$ matrices and $\mathbf{y}$ vectors from `train_dict_bieos`. To build $\mathbf{X_{rl}}$ and $\mathbf{y}_{rl}$, you just need to reverse the sentences. Your program should produce two matrices and two vectors. 9 points

4. Write the code to train your two models. 4 points

5. Write a `predict()` function to carry out the prediction. The input will be a sentence represented as a list of dictionaries. Each dictionary will contain a word, like:

   ```
   {'form': 'provincial', 'pos': 'JJ', 'chunk': 'I-NP', 'ner': 'O'},
   ```

   In addition to predict the NER tag, you will extract the probability of this prediction. This is done by calling `classifier.predict_proba()` instead of `classifier.predict()`. The `classifier.predict_proba()` method returns the prediction and it probability in a tuple[3]. 9 points

   For each word, your `predict()` function will add the predicted tag, `ner_pred`, and its probability, `proba`, as for instance:

   ```
   {'form': 'provincial', 'pos': 'JJ', 'chunk': 'I-NP', 'ner': 'O',
     'ner_pred': 'O', 'proba': 0.986},
   ```

   The `predict()` function will return the modified sentence.

6. Applying your classifiers will result into two predictions for each word: The first one from the left-to-right direction and the second one from the right-to-left. How will you merge them? 3 points

2.4 Entity Linking

You will now link the named entities to unique identifiers. Let us suppose that you NER has recognized the mention EU as a named entity, the candidates can be universities, at least 5:

1. Edinburgh University, Scotland
2. Ehime University, Matsuyama, Ehime, Japan
3. Eastern University (United States), Pennsylvania, United States
4. Elon University, North Carolina, United States
5. Emory University, Atlanta, United States

as well as European Union. In this section, you will generate the candidates and use a baseline technique to rank them.

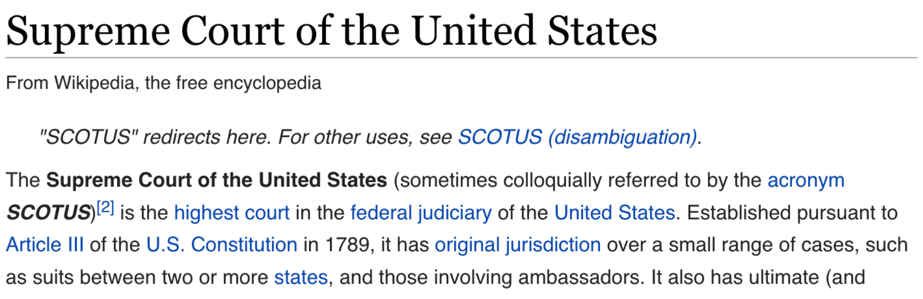
2.4.1 Candidate Generation

Understanding the Generation

1. Read the introduction to Sect. 4 and Sect. 4.1 of Yang et al. (2017). What resources are they using to find the possible names of an entity? 2 points

2. You will extract the entity candidates from Wikipedia using the wikilinks. The wikilinks are the clickable words in a Wikipedia article. When you click on the phrase *United States* in Fig. 3, for instance, you will move to the target of this link: The article on the United States. We will consider that each article in Wikipedia corresponds to an entity, for instance the article *United States* is equivalent to the entity United States.

[3]In reality, this procedure is more complex in sklearn. We simplify it for the examination.


Supreme Court of the United States

From Wikipedia, the free encyclopedia

"SCOTUS" redirects here. For other uses, see SCOTUS (disambiguation).

The **Supreme Court of the United States** (sometimes colloquially referred to by the acronym ***SCOTUS***)[2] is the highest court in the federal judiciary of the United States. Established pursuant to Article III of the U.S. Constitution in 1789, it has original jurisdiction over a small range of cases, such as suits between two or more states, and those involving ambassadors. It also has ultimate (and

Figure 3: Supreme Court of the United States

In the source text of wikipedia, the wikilinks are delimited by square brackets as in this (simplified) example below from the article on the Supreme Court of the United States:

```
The '''Supreme Court of the United States''' is the [[Supreme
court|highest court]] in the [[Federal judiciary of the
United States|federal judiciary]] of the [[United States]].
Established pursuant to [[Article Three of the United
States Constitution|Article III]] of the [[United States
Constitution|U.S. Constitution]] in 1789, it has [[original
jurisdiction]] over a small range of cases, such as suits
between two or more [[U.S. state|states]], and those involving
ambassadors.
```

Sometimes the link and the clickable words are the same as in `[[United States]]`. Some links consist of two parts as:

```
[[Federal judiciary of the United States|federal judiciary]]
```

where the first part is the link, a Wikipedia page whose title is *Federal judiciary of the United States* and the second part is the text that shows on the page, *federal judiciary*, see Fig. 3.
In a construct like `[[link|label]]`, what part corresponds to the mention and what part to the entity? 2 points

3. In a construct like `[[link]]`, how would you extract the mention and the entity? 2 points

Programming. You will now extract the mention-entity pairs using regular expressions. You will suppose the page is available in a variable call `document` that only contains the text and the wikilink markup.

1. Write a regular expression that matches the `[[link]]` patterns and stores the values of the links; 5 points

   Applying the code:

```
link_re1 = ...
entities = [match.group(1) for match
            in re.finditer(link_re1, document)]
```

to the excerpt above should result in

```
['United States', 'original jurisdiction']
```

2. Write a regular expression that matches `[[link|label]]` patterns and stores the values of the link and of the label; Applying the code: 8 points

```
link_re2 =...
entity_mentions = [(match.group(1), match.group(2)) for match
                   in re.finditer(link_re2, document)]
```

to the excerpt above should result in

```
[('Supreme court', 'highest court'),
('Federal judiciary of the United States', 'federal judiciary'),
('Article Three of the United States Constitution', 'Article III'),
('United States Constitution', 'U.S. Constitution'),
('U.S. state', 'states')]
```

3. Write the code to convert the result of the first regular expression so that it is identical to the second one. Concatenate the two lists. 2 points

2.4.2 Candidate Ranking

Given the mention of a named entity in a text, you will rank the entities using the list of mention-entity pairs you have collected from the corpus. As baseline, we will simply associate the entity with the highest count to the mention.

1. This simple ranking method is equivalent to a feature used by Yang et al. (2017). Which one? (Give the number); 3 points
2. Write a function that takes the variable `entity_mentions` as input and outputs a dictionary, where the keys are the mentions and the values, dictionaries, where the keys are the entities and the values their counts. 10 points
3. Finally, having a text where your NER system has recognized all mentions of named entities, how would you link them to entity identifiers? We will assume that the identifiers are the Wikipedia page titles. 5 points

References

Ji, H., Pan, X., Zhang, B., Nothman, J., Mayfield, J., McNamee, P., and Costello, C. (2017). Overview of TAC-KBP2017 13 languages entity discovery and linking. In *Proceedings of the Tenth Text Analysis Conference (TAC 2017)*, Gaithersburg, Maryland.

Pradhan, S., Ramshaw, L., Marcus, M., Palmer, M., Weischedel, R., and Xue, N. (2011). CoNLL-2011 shared task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon. Association for Computational Linguistics.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147, Edmonton.

Yang, T., Du, D., and Zhang, F. (2017). The TAI system for trilingual entity discovery and linking track in TAC KBP 2017. In *Proceedings of the Tenth Text Analysis Conference (TAC 2017)*, Gaithersburg, Maryland.

3 Appendix

You can use the tables below in your exam papers. Just detach them and fill them in. Insert them with your sheets. Do not forget to write your exam code on the sheets.

s	6	7	6			
e	5	6	5			
l	4	5	4	3		
b	3	4	3	2	3	
a	2	3	2	3	4	5
c	1	2	3	4	5	6
start	0	1	2	3	4	5
	start	t	a	b	l	e

Word	**Tag IOB**	**Tag BIEOS**
SEOUL	I-GPE-NAM	S-GPE-NAM
Jan.	O	O
9	O	O
Xinhua		
South		
Korea		
’s		
main		
opposition		
Democratic		
United		
Party		
on		
Wednesday		
elected		
a		
new		
interim		
leader		