Annotation for Machine Learning *

Assignment 1 (50 + 10*)

You are provided with a Google Colab notebook. Make a copy of the notebook, personalize it (put your names), fill the code and text cells with required code and text and submit the copy of the notebook (`.ipynb` file) via Nestor.[1]
The star points are bonus points, and they help to reach the max points (50) if needed.
NB: Ignoring instructions about formatting will be penalized.

1 Read and analyze a corpus (4 + 2 + 3)

You are provided with a portion of the Parallel Meaning Bank (PMB) in `pmb.zip`. Extract the data from the archive (it might take some time) and browse it to understand the content of files. A PMB document contains several translations of the same text/sentence.

A Read the entire pmb data in a dictionary `pmb` with keys `part/doc` numbers and find out how many documents are there in total and how are they distributed across the two subcorpora: RTE and Tatoeba.

```
pmb['01/0777']
{'DE': 'Unsere Mannschaft hat alle Spiele verloren.\n',
 'EN': 'Our team lost all its games.\n',
 'IT': 'La nostra squadra ha perso tutte le partite.\n',
 'NL': 'Ons team heeft alle wedstrijden verloren.\n',
 'met': 'Tatoeba'}
```

B Walking through all PMB directories and reading files is time consuming. Write all the PMB data in the `pmb.txt` file. Don't alter the raw text! We need the original data unchanged. You are provided with the file for reference.

C Using nltk's default tokenizer, tokenize the pmb data, lowercase every token, and for each language provide top 10 most frequent words (structured in a table with counts). Point out major similarities and differences across the lists.[2]

2 Stand-off tokenization (3 + 7 + 5*)

You are going to tokenize the English translations of the PMB with two different tokenizers and save the tokenization as a stand-off annotation (without the original data). You are already pro

*In case the assignment file is updated with an important information (excl. typos and style), the changes will be highlighted as follows: edit1, edit2, edit3 edit4, edit5. Hopefully there won't be many edits necessary :).
[1]For the markdown syntax you might want to consult this.
[2]Use wiktionary if you do not know a meaning of a word in some language

A Read the `pmb` dictionary from `pmb.txt` with the provided `read_pmb_from_file` function. Tokenize the English texts with the NLTK and SpaCy tokenizers. To make this modular, write a function `pmb_en_seg`, which works as shown below.

```
nltk_sen_tok = pmb_en_seg(pmb, 'nltk')
spcy_sen_tok = pmb_en_seg(pmb, spacy.load("en_core_web_sm"))
nltk_sen_tok['01/0777']
[['Our', 'team', 'lost', 'all', 'its', 'games', '.']]
spcy_sen_tok['01/0777']
[['Our', 'team', 'lost', 'all', 'its', 'games', '.']]
```

The tokenization must capture info about sentence boundaries and word token boundaries. That's why the tokenization info is organized as a list of sentences, where each sentence is a list of tokens. To detect sentence boundaries, use `nltk.tokenize.sent_tokenize`, and then use `nltk.tokenize.word_tokenize` to detect tokens inside the sentences.
How many sentences and tokens did each tokenization produce (separately)?

NB: The SpaCy tokenizer is bad with white spaces. Do additional processing to make sure that newlines `\n` are not identified as tokens or as peripheral parts of tokens.

B Write down segmentation info (word tokenization + sentence boundary detection) in `en.nltk.seg` and `en.spcy.seg`[3] with a function that works as follows:

```
nltk_seg = sen_tok_to_off_seg(pmb, nltk_sen_tok, 'en.nltk.seg')
spcy_seg = sen_tok_to_off_seg(pmb, spcy_sen_tok, 'en.spcy.seg')
pmb['01/0777']['EN']
'Our team lost all its games.\n'
nltk_seg['01/0777']
'$--_^---_^---_^--_^--_^----^_'
```

You can see how $, ^, _, and - symbols (i.e., labels in this case) mark segmentation info for each character position.[4] The stand-off segmentation files are provided as a reference. These files show how to keep segmentation annotation of the corpus without the actual corpus.

***C** The NLTK's `word_tokenize` alters the original tokens for one common punctuation mark. This causes mismatch between the positions of STOI-labels and the original character positions because new tokens are of different length than the original, replaced ones. Find out this punctuation mark and fix this issue.

3 Part-of-speech tagging $(7 + 3 + 5^*)$

You have access to the stand-off segmentation and the raw PMB text. Now we need to label each token with part-of-speech (POS) tag (only for the English data).

A Write a function `apply_off_seg` that reads the stand-off segmentation file `en.*.seg` and the pmb raw data `pmb.txt` and returns the dictionary similar to `nltk_sen_tok` and `spcy_sen_tok` from **A**.

```
nltk_sen_tok = apply_off_seg('pmb.txt', 'en.nltk.seg')
spcy_sen_tok = apply_off_seg('pmb.txt', 'en.spcy.seg')
```

[3] For you convenience, I also provide you with `*.raw.seg` file, containing the raw docs and segmentation annotations together. Note that these files are not stand-off annotations anymore.

[4] $, ^, _, and - symbols represent **S**, **T**, **O**, **I** labels for characters: **S** is for **S**entence start, **T** for **T**oken start, **O** for **O**utside of a token, **I** for **I**nside a token.

B After we have a list of sentences that are list of tokens, we can label each token with POS tags by feeding POS taggers with a list of tokenized sentences. You are provided with five functions that tag a list of tokenized sentences. Write a function `pos_tag3` that tags tokens with the three taggers and works as:

```
tagged3_nltk = pos_tag5(nltk_sen_tok)
tagged3_spcy = pos_tag5(spcy_sen_tok)
tagged3_nltk['01/0777']
{'hun': [['PRP$', 'NN', 'VBD', 'PDT', 'PRP$', 'NNS', '.']]
 'nltk': [['PRP$', 'NN', 'VBD', 'DT', 'PRP$', 'NNS', '.']]
 'spacy': [['PRP$', 'NN', 'VBD', 'PDT', 'PRP$', 'NNS', '.']]}
```

***C** Senna and Stanford taggers are slower. You need to find a smart way to use them to tag all the EN sentences in a feasible time.

4 Compare annotations $(5 + 3 + 5 + 5 + 7)$

After automatically obtaining stand-off annotations for segmentation and POS tagging, we can compare them. We don't need the original raw texts to detect differences in annotations, but we need access to it to understand and judge the differences.

A Write a function `contrast_seg` that takes two stand-off segmentation files and `pmb.txt` and writes an HTML file that displays only those documents for which the segmentations differ.[5]

```
contrast_seg('pmb.txt', 'en.nltk.seg', 'en.spcy.seg', 'seg.html')
```

B Pick 3 different types of tokenization differences and discuss each of them. What do you think which one is correct (if there is one) or what tokenization would be correct? Write a code that reports the total number of segmentation labels and the number (and %) of the cases NLTK and SpaCy segmentations differ for the entire pmb date.

C Write a function `contrast_pos` that takes an output of `pos_tag5` and `pmb.txt` and writes an HTML file that displays only those documents for which the POS tagging differ. Highlight the difference with colors.[6]

```
# pos5_nltk shows that nltk segmentation was used to obtain tokens for POS tagging
contrast_pos(tagged5_nltk, 'pmb.txt', 'pos5_nltk.html')
```

D Pick one of the two segmentations (NLTK or SpaCy) and produce POS tags for the tokens with all the five POS taggers. Provide a code that prints the total number of tokens, and counts (and %) for each voting scenarios: unanimous (all taggers agree), 4 vs 1, 3 vs 2, 3 vs 1 vs 1, etc. If you are using only three taggers then you will have a less number of scenarios.

E Pick 10 different types of POS tagging differences and discuss each of them. Make sure that these 10 cases are picked as evenly as possible from different voting scenarios. For each chosen token and non-unanimous five POS tags, detect a gold (i.e. correct) tag based on the Penn TreeBank Project guidelines for part-of-speech tagging. The guidelines are available here.[7]

[5] A sample HTML file `seg.html` is provided.
[6] A sample HTML file `pos5_nltk.html` is provided.
[7] Examples of arguing for gold POS tags based on the guidelines are provided in the Colab notebook.