Intro to NLP 2022: Assignment 1

In this assignment, we work with a dataset that contains sentences from news articles. It has been collected for a shared task at SemEval 2018 for *Complex Word Identification*.

Task Description: https://sites.google.com/view/cwisharedtask2018/
Code for the assignment: *intro2nlp_assignment1_code.zip*

You submit a **pdf** of this document, the format should not be changed.
All floating point numbers should be rounded to **two decimals**.

Your analyses should be conducted using **python 3.8**.
You submit a **zip**-file containing all your code.
You are allowed to use Python packages (e.g. pandas, sklearn).

Each team member needs to be able to explain the details of the submission. By default, all team members will receive the same grade. If this seems unjust to you, provide an extra statement indicating the workload of each team member.

Total points: 20
Structure:
- Part A: Linguistic analysis of the dataset using spacy, 6 points
- Part B: Understanding the task of complex word identification, 7 points
- Part C: Modeling the task with an LSTM, 7 points
- Bonus tasks: options for obtaining a grade >8

Fill in your details below:
Group number: 69

Student 1
Name: Don Mani
Student id: 2693434

Student 2
Name: Azhar Shaikh
Student id: 2701842

Student 3
Name: Caroline Hallman
Student id: 2640914

PART A: Linguistic analysis using spaCy

In the first part of the assignment, we focus on an analysis of the sentences in the training data.
File: *data/preprocessed/train/sentences.txt*
Implement your analyses in *TODO_analyses.py.*

Note that we are using the most recent spaCy version (3.2) and the model *en_core_web_sm*. Results might vary for other versions. If you cannot use 3.2, clearly explain this to your TA and specify on your submission which version you are using instead.

1. **Tokenization** (1 point)
   Process the dataset using the spaCy package and extract the following information:
   Number of tokens: **15477**
   Number of types: **3721**
   Number of words: **13242**
   Average number of words per sentence: **18.89**
   Average word length: **6.54**
   Provide the definition that you used to determine words:

   Words are a list of all tokens that are not punctuations. For example, ‘cat’ is a word, while ‘!’ is not a word. Removal of newline character (\n) is the only preprocessing that is done for the entire assignment.

2. **Word Classes** (1.5 points)
   Run the default part-of-speech tagger on the dataset and identify the ten most frequent POS tags. Complete the table below for these ten tags (the tagger in the model *en_core_web_sm* is trained on the PENN Treebank tagset).

Finegrained POS-tag	Universal POS-Tag	Occurrences	Relative Tag Frequency (%)	3 most frequent tokens with this tag	Example for an infrequent token with this tag
NN	**NOUN**	**2102**	**13.58**	**\\, year, report**	**project**
NNP	**PROPN**	**2020**	**13.05**	**\\, US, President**	**Navy**
IN	**ADP**	**1744**	**11.27**	**of, in, to**	**By**
DT	**DET**	**1379**	**8.90**	**the, a, The**	**Each**
JJ	**ADJ**	**869**	**5.61**	**other, Russian, presidential**	**Sebastian**
NNS	**NOUN**	**779**	**5.03**	**ants, troops, people**	**areas**
,	**PUNCT**	**699**	**4.52**	**,, ;, …**	**…**

VBD	**VERB**	**660**	**4.26**	**was, were, said**	**suggested**
.	**PUNCT**	**655**	**4.23**	**., ?, !**	**!**
VBN	**VERB**	**500**	**3.23**	**been, accused, reported**	**acquitted**

3. **N-Grams** (1.5 points)
   Calculate the distribution of n-grams and provide the 3 most frequent
   Token bigrams:

Most frequent bigrams	Count
\\ “	**240**
. The	**104**
of the	**82**

Token trigrams:

Most frequent trigrams	Count
, \\ “	**40**
. \\ “	**37**
\\ “ .	**28**

POS bigrams:

Most frequent fine grained POS bigrams	Count
DT NN	**671**
NNP NNP	**608**
IN DT	**586**

POS trigrams:

Most frequent fine grained POS trigrams	Count

IN DT NN	**292**
NNP NNP NNP	**200**
DT NN IN	**195**

4. **Lemmatization** (1 point)
   Provide an example for a lemma that occurs in more than two inflections in the dataset.

   Lemma: **write**
   Inflected Forms: **['wrote', 'write', 'wrote', 'written', 'write']**

   Example sentences for each form:

Virginia Álvarez , who <u>wrote</u> the report , noted , \" instead of listening to their demands , instead of starting a dialogue , authorities are doing everything they can to impede people from protesting \" .,

Austrian police find dozens dead inside lorry To <u>write</u> , edit , start or view other articles on Austria , see the Austria Portal Flag of Austria.svg The turning for Parndorf off the A4 , from file .,

In his later years he became an executive and consultant , <u>wrote</u> an autobiography , and advocated for returning to the Moon along with Neil Armstrong , the first human to walk on the Moon , who died in 2012 .,

The law was <u>written</u> in response to violent protests .,

Four men go before Hungarian court after 71 found dead in lorry To <u>write</u> , edit , start or view other articles on Austria , see the Austria Portal Four foreign men have today gone before a court in Hungary on suspicion of people smuggling .

5. **Named Entity Recognition** (1 point)

   Number of named entities: **893**
   Number of different entity labels: **17**

   Analyze the named entities in the first five sentences. Are they identified correctly? If not, explain your answer and propose a better decision.

Sentence 1
children are thought to be aged three , eight DATE , and ten years DATE , alongside an eighteen-month-old DATE baby .

Sentence 2
We mixed different concentrations of ROS GPE with the spores , plated them out on petridishes with an agar-solution where fungus can grow on .

Sentence 3
They feel they are under-represented in higher education and are suffering in a regional economic downturn .

Sentence 4
Especially as it concerns a third ORDINAL party building up its military presence near our borders .

Sentence 5
Police said three CARDINAL children were hospitalised for \ ORG " severe dehydration \" .

In sentence 1, ages of children are misclassified as DATE when it should have been classified as CARDINAL.
In sentence 2, ROS was misclassified as GPE. A better classification would be PRODUCT.
Sentence 3 does not have any entities.
In sentence 4, third is correctly classified as an ORDINAL.
In sentence 5, \ should not have been classified as ORG.

PART B: Understanding the task of complex word identification

6. **Explore the dataset** (1.5 points)
   Read the documentation (https://sites.google.com/view/cwisharedtask2018/datasets) of the dataset and provide an answer to the following questions:

   a) What do the start and offset values refer to? Provide an example.
   Start offset (third column) represents the beginning index of the target word and end offset (fourth column) represents the end index of the target word in the actual sentence (column two).

   Sample from the dataset:
   Both China and the Philippines flexed their muscles on Wednesday. 31 37 flexed 10 10 2 6 1 0.4

   b) What does it mean if a target word has a probabilistic label of 0.4?
   It means that 40% of the total annotators found the target word difficult.

   c) The dataset was annotated by native and non-native speakers. How do the binary and the probabilistic complexity label account for this distinction?

   Binary labels do not account for any distinction between native and non-native speakers as a target word is marked difficult if at least one annotator (either native or non-native) finds it difficult.

   Similarly the probabilistic label also does not account for any distinction as it is calculated as no of annotators found target word difficult(native or non-native)/ total number of annotators.

7. **Extract basic statistics** (0.5 point)
   Let's have a closer look at the labels for this task.
   Use the file *data/original/english/WikiNews_Train.tsv* and extract the following columns:
   Target word, binary label, probabilistic label

Target word	**Binary label**	**Probabilistic label**
Guatemalan	1	0.05
approves	1	0.05

Supreme	0	0.00
Court	0	0.00
approves impeachment	1	0.05

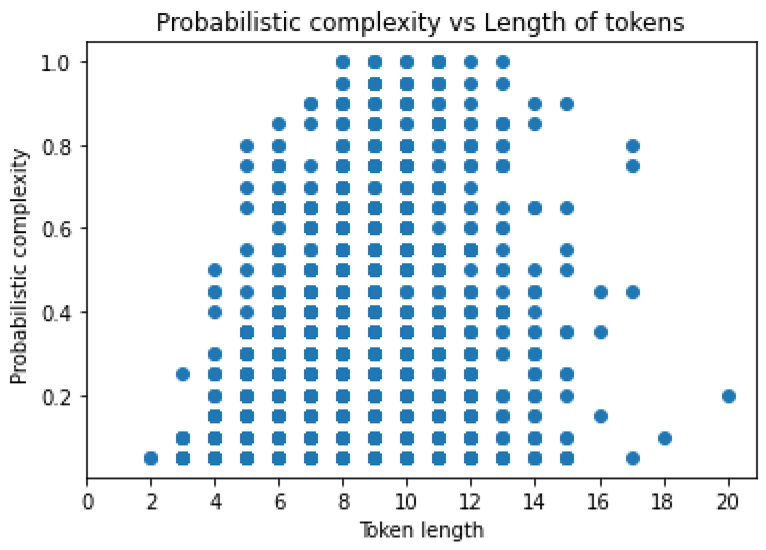
Provide the following information:
Number of instances labeled with 0: **4530**
Number of instances labeled with 1: **3216**
Min, max, median, mean, and stdev of the probabilistic label:
Min: 0.00
Max: 1.00
Median: 0.00
Mean: 0.08
Stdev: 0.17
Number of instances consisting of more than one token: **966**
Maximum number of tokens for an instance: **8**

8. **Explore linguistic characteristics** (2 points)
For simplicity, we will focus on the instances which consist only of a single token and have been labeled as complex by at least one annotator.
Calculate the length of the tokens as the number of characters.
Calculate the frequency of the tokens using the wordfreq package (https://pypi.org/project/wordfreq/).
Provide the Pearson correlation of length and frequency with the probabilistic complexity label:

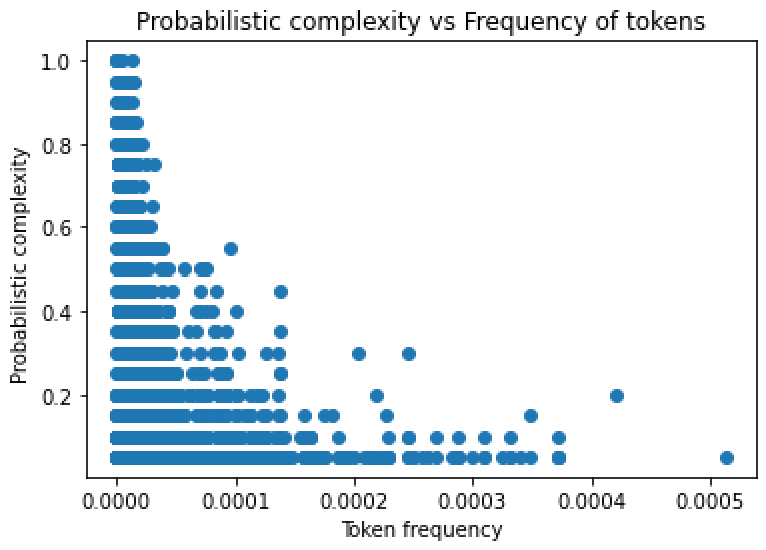
Pearson correlation length and complexity: **0.28**
Pearson correlation frequency and complexity: **-0.30**

Provide 3 scatter plots with the probabilistic complexity on the y-axis.
X-axis: 1) Length 2) Frequency 3) POS tag
Set the ranges of the x and y axes meaningfully.

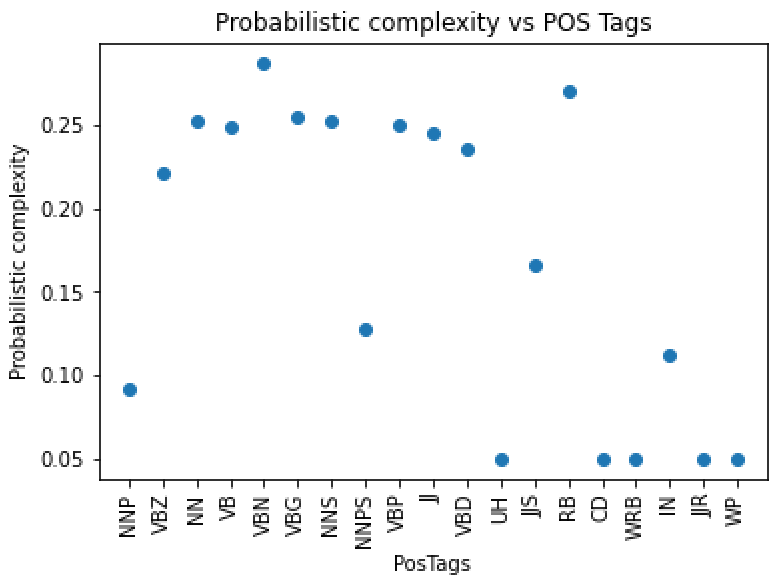
Plot 1:


Probabilistic complexity vs Length of tokens
Probabilistic complexity
Token length
0
2
4
6
8
10
12
14
16
18
20
0.2
0.4
0.6
0.8
1.0

Plot 2:


Probabilistic complexity vs Frequency of tokens
Probabilistic complexity
Token frequency
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
0.2
0.4
0.6
0.8
1.0

Plot 3:



Interpret the results in 3-5 sentences:

We do not observe any pattern in plot 1(probabilistic complexity vs token length) indicating there is no linear relation between probabilistic complexity and token length which is in accordance with the 0.28 pearson correlation we obtained for probabilistic complexity and token length. Therefore token length does not seem to influence token complexity.

From plot 2 (probabilistic complexity vs token frequency) we observe that more frequently occurring tokens have low complexity. Less frequent tokens on the other hand can have varying degrees of complexity.

From plot 3 (probabilistic complexity vs fine grained pos-tags) we can infer that tokens that are nouns and verbs are the most complex on average.

9. **Reflection (1 Point)**
Can you think of another linguistic characteristic that might have an influence on the perceived complexity of a word? Propose at least one and explain your choice in 2-4 sentences.

A linguistic characteristic that might have an influence on the perceived complexity of a word could be the number of inflected forms of a word. We hypothesize that a word with more inflected forms is less complex than a word with relatively lesser inflected forms.

10. **Baselines** (2 Points)
Implement four baselines for the task in *TODO_baselines.py*.
Majority baseline: always assigns the majority class
Random baseline: randomly assigns one of the classes
Length baseline: determines the class based on a length threshold
Frequency baseline: determines the class based on a frequency threshold

Test different thresholds and choose the one which yields the highest accuracy on the dev_data:
Length threshold: **8**
Frequency threshold: **1**

Fill in the table below (round to two decimals!):

Baseline	Accuracy on dev	Accuracy on test
Majority	**0.81**	**0.75**
Random	**0.47**	**0.48**
Length	**0.84**	**0.81**
Frequency	**0.77**	**0.75**

Interpret the results in 2-3 sentences.

We find that the majority of tokens in the dataset are non-complex and since we predict all the tokens as non-complex in the majority baseline, the accuracy value represents the percentage of non-complex tokens in the dataset.

For the random baseline since we predict two classes randomly we expect to see around 50% accuracy which is what we observe too.

For the frequency baseline we observe predicting tokens with very low frequency as complex gives better accuracy.

For the Length baseline we observe that if the length of a token is greater than 8 then predicting it as a complex token gives us good accuracy.

Store the predictions in a way that allows you to calculate precision, recall, and F-measure and fill the table in exercise 12.

PART C: Modeling the task

For part C, we use an implementation for a vanilla LSTM which was originally developed for a named entity recognition project for a Stanford course. You can find more documentation here: https://github.com/cs230-stanford/cs230-code-examples/tree/master/pytorch/nlp

11. **Understanding the code** (1.5 Points)
    Familiarize yourself with our version of the code and try to understand what is going on. Answer in your own words (1-3 sentences per question)
    Run the file *build_vocab.py*. What does this script do?
    The script analyzes the sentences and tokens and parses it according to its corresponding directories which are 1)train 2)val and 3)test. From those 3 directories, the script generates 1)word.txt 2)tags.txt and3)dataset_params.json. Word.txt includes vocabulary of unique words in the 3 parsed sentences dataset and tags.txt distinct tages in the 3 parsed labels datasets. Dataset_params.json transforms the 3 datasets in a json format. Tokenization and frequency order of vocabulary stay the same and have the default arguments of 1.

    Inspect the file *model/net.py.* Which layers are being used and what is their function?

Layer	**Functionality**
LSTM	**When having sequential input, it generates output of each token in a sentence.**
Embedding layer	**Maps a token's index to its embedding vector**
Fully connected layer	**Converts output of LSTM for each token to distribution over the output classes (our case 2 classes N or C)**

How could you change the loss function of the model?

We could use the nn.CrossEntropy function builtin pytorch instead of the loss function defined in the code.

12. **Detailed evaluation** (2.5 points)
    Train the model on the data in *preprocessed/train* and *preprocessed/dev* by running the code in *train.py*.
    Evaluate the model on the data in *preprocessed/test* by running *evaluate.py*.
    The original code only outputs the accuracy and the loss of the model. I adapted the code, so that it writes the predictions to *experiments/base_model/model_output.tsv.*
    Implement calculations for precision, recall, and F1 for each class in *TODO_detailed_evaluation.py*. You can use existing functions but make sure that you understand how they work.
    Provide the results for the baselines and the LSTM in the table below.

Model	Class N			Class C			Weighted Average	Macro Average
	Precision	Recall	F1	Precision	Recall	F1	F1	F1
Random	**0.81**	**0.50**	**0.61**	**0.21**	**0.53**	**0.30**	**0.55**	**0.46**
Majority	**0.80**	**1.00**	**0.89**	**0.00**	**0.00**	**0.00**	**0.71**	**0.44**
Length	**0.85**	**0.99**	**0.91**	**0.85**	**0.31**	**0.46**	**0.82**	**0.69**
Frequency	**0.89**	**0.84**	**0.87**	**0.49**	**0.60**	**0.54**	**0.80**	**0.70**
LSTM	**0.88**	**0.94**	**0.91**	**0.68**	**0.48**	**0.57**	**0.84**	**0.74**

13. **Interpretation** (1.5 Points)
    Compare the performance to the results in the shared task (https://aclanthology.org/W18-0507.pdf) and interpret the results in 3-5 sentences. Don't forget to check the number of instances in the training and test data and integrate this into your reflection.

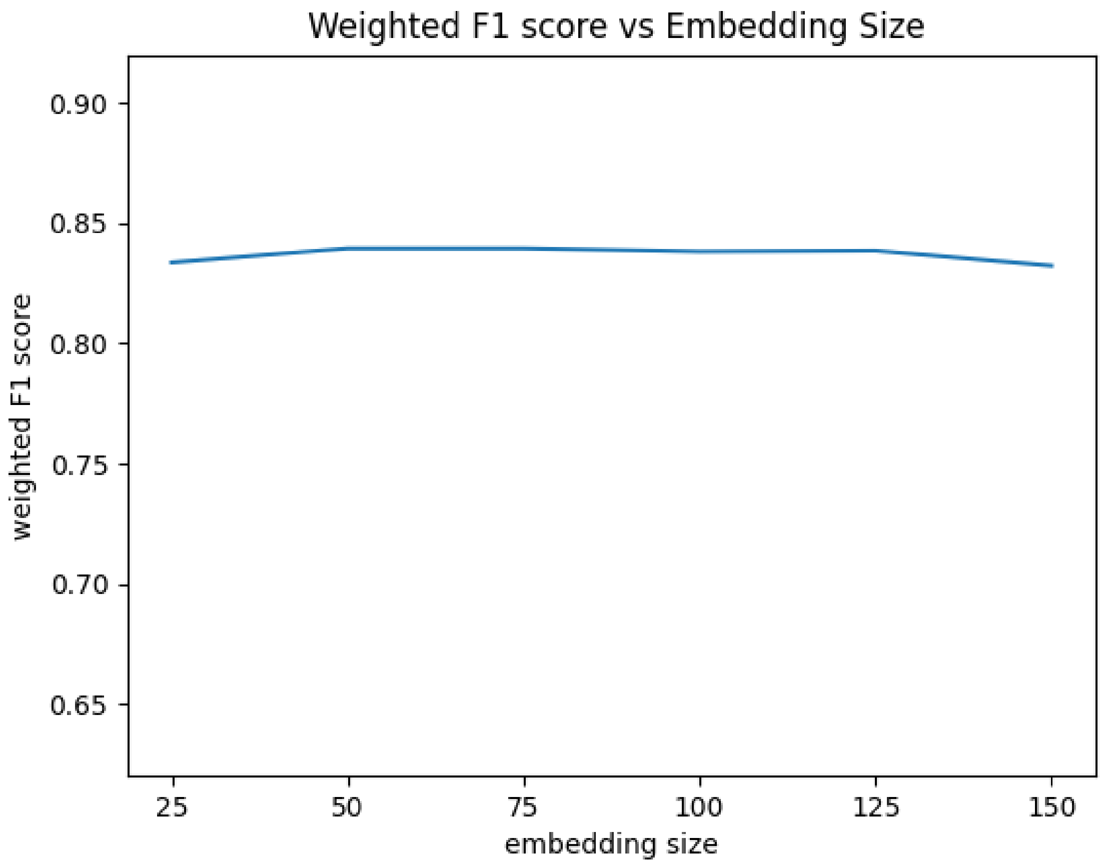
    Number of instances:
    - **Training : 7746**
    - **Test data: 1287**

We observe the LSTM model performs the best compared to the implemented baselines. The LSTM model would be placed 23rd on the shared task leaderboard and is 10 points less than the best performing model (CAMB). However it should be noted that Camb was trained on a bigger corpus of training data (27,299 instances) with complex domain specific feature engineering while the LSTM model was trained on a much smaller dataset(7746) with no hyper-parameter tuning. Training on a bigger corpus of data could improve performance but for the CWI task better feature engineering methods seems to outweigh purely data-intensive approaches like deep learning methods. This is

evidenced by the fact that the LSTM trained using Transfer learning by NILC could not beat the feature engineering method by CAMB. Furthermore, frequency and length based feature engineering approaches seem to do better. This is in line with the high scores we observed for the naive frequency and length based baselines that have been implemented.

14. **Experiments** (2 points)
    Vary a hyperparameter of your choice and plot the F1-results (weighted average) for at least 5 different values. Examples for hyperparameters are embedding size, learning rate, number of epochs, random seed,

    Hyperparameter: **embedding size: [25, 50, 75, 100, 125, 150]**
    Plot:



Interpret the result (2-4 sentences):
As can be seen from the above plot the embedding size doesn't seem to have any major effect on the weighted F1-score values. For the task of complex word identification, embedding size of the model does not seem to matter much.

Provide 3 examples for which the label changes when the hyperparameter changes:

1. **forensic, N at embedding_dim = 50, C at embedding_dim=100**
2. **coordinator, N at embedding_dim = 100, C at embedding_dim=125**
3. **ensued, N at embedding_dim = 50, C at embedding_dim=25**

Bonus Tasks

The maximum grade you can get for the assignment is an 8. If you want to obtain a better grade, you need to individually send results for one of the bonus tasks to intro2nlp@googlegroups.com. If the group project grade is less than an 8, we do not check the bonus task submission. If the group project grade is an 8 and you submitted an answer for a bonus task, you might still only receive an 8, if the quality of the bonus task submission is not sufficient.

Task options:

- Provide answers for exercises 8 and 12-14 for at least one of the other languages of the CWI task.
- Improve the model by making a substantial change. Varying a hyperparameter or simply adding another layer **is not** a substantial change. Motivate your modification and interpret the findings.
- Identifying complex words is only the first step for lexical simplification. Read up on related work and explain potential architectures for contextualized lexical simplification in detail.