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: 40

Student 1
Name: Andreea Hazu
Student id: 2645225

Student 2
Name: Giulia Bössenecker
Student id: 2624238

Student 3
Name: Maike Nützel
Student id: 2580833

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: 16130
   Number of types: 3746
   Number of words: 13895
   Average number of words per sentence: 19.35
   Average word length: 4.72

   Provide the definition that you used to determine words: tokens after removing punctuations.

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	2074	0.13	\, year, report	project
NNP	PROPN	2063	0.13	\,US, President	Navy
IN	ADP	1745	0.11	of,in,to	By
DT	DET	1378	0.09	the,a,the	Each
JJ	ADJ	868	0.05	other,Russian,presidential	Sebastian
NNS	NOUN	774	0.05	ants,troops,people	areas
,	PUNCT	699	0.04	, ; …	…
VBD	VERB	660	0.04	was,were,said	suggested
.	PUNCT	655	0.04	. ? !	!
VBN	VERB	500	0.03	been, accused, reported	acquitted

3. **N-Grams** (1.5 points)
   Calculate the distribution of n-grams and provide the 3 most frequent
   Note: n-grams are separated by spaces.

   Token bigrams: [('**\\ "**', 240), ('**. The**', 104), ('**of the**', 82)]
   Token trigrams: [('**, \\ "**', 40), ('**. \\ "**', 37), ('**\\ " .**', 28)]

   POS bigrams: [('DT NN', 671), ('NNP NNP', 611), ('IN DT', 587)]
   POS trigrams: [('IN DT NN', 293), ('NNP NNP NNP', 201), ('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: be
   Inflected Forms: ‘are’, ‘were’, ‘was’
   Example sentences for each form:

   ‘Children are thought to be aged three, eight, and ten years , alongside an eighteen-month-old baby’

   ‘Police said three children were hospitalised for "severe dehydration"’

   ‘That decision was the one challenged unsuccessfully in the High Court’

5. **Named Entity Recognition** (1 point)
   Number of named entities: 1627
   Number of unique named entities: 888
   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.

   All 5 sentences have named entities identified correctly.

   children are thought to be aged three , eight , and ten years , alongside an eighteen-month-old baby .
   Named entities: ['three , eight', 'ten years', 'eighteen-month-old']

   We mixed different concentrations of ROS with the spores, plated them out on petridishes with an agar-solution where fungus can grow on .
   Named entities: ['ROS']

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

Named entities: []

Especially as it concerns a third party building up its military presence near our borders.
Named entities: ['third']

Police said three children were hospitalised for \" severe dehydration \" .
Named entities: ['three', '\\']

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.

   The start and offset values refer to the start and end index of the target word within the sentence. To provide an example, in the training data:
   - Looking at the first row, 31 is the start index in the sentence where the target word “flexed their muscles” is. 51 is the index where the target word ends (or start index + the length of the start index)
   - The logic is the same when looking at the second row, 31 is the start index of “flexed” in the sentence, 37 is the index where the target word ends.

   b) What does it mean if a target word has a probabilistic label of 0.4?

   It means that 40% of the total number of annotators who saw the sentence marked the word as difficult. In the example displayed on second row, 8 annotators out of 20 marked the word “flexed” as 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?

   The binary and the probabilistic complexity label do not account for any distinction between annotations made by native or non-native speakers.

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

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: 0.00, 1.00, 0.00, 0.08, 0.17
Number of instances consisting of more than one token: 1086
Maximum number of tokens for an instance: 10

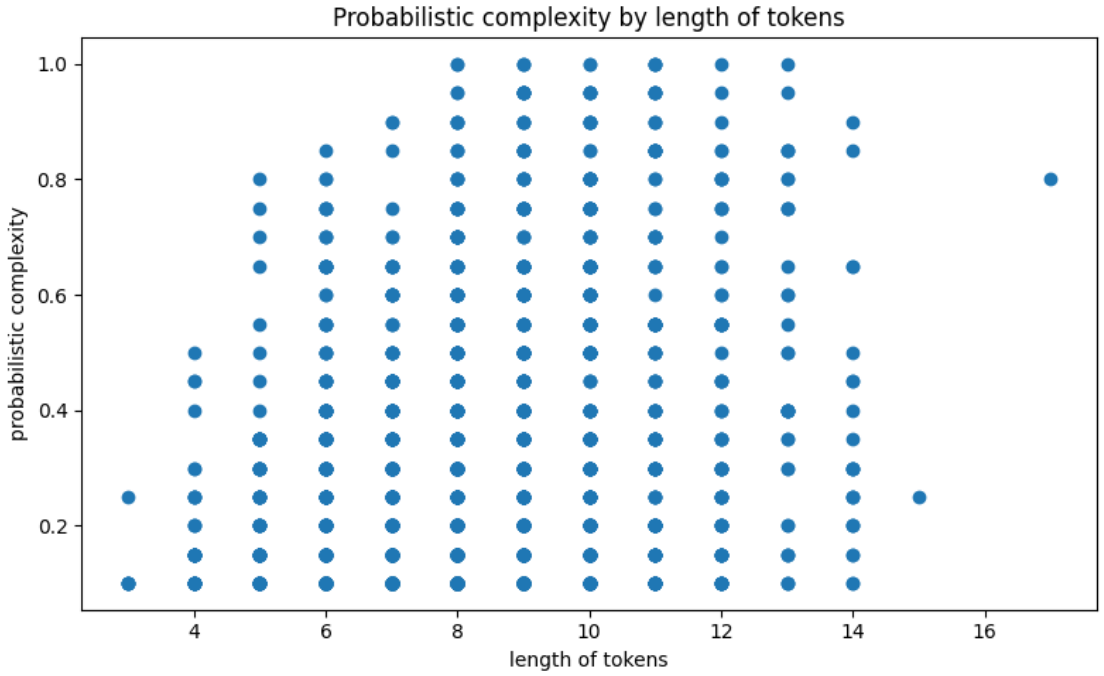
8. **Explore linguistic characteristics** (2 points)
For simplicity, we will focus on the instances which consist only of a single token and have been labeled as complex by at least one annotator.
Calculate the length of the tokens as the number of characters.
Calculate the frequency of the tokens using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

Provide the Pearson correlation of length and frequency with the probabilistic complexity label:

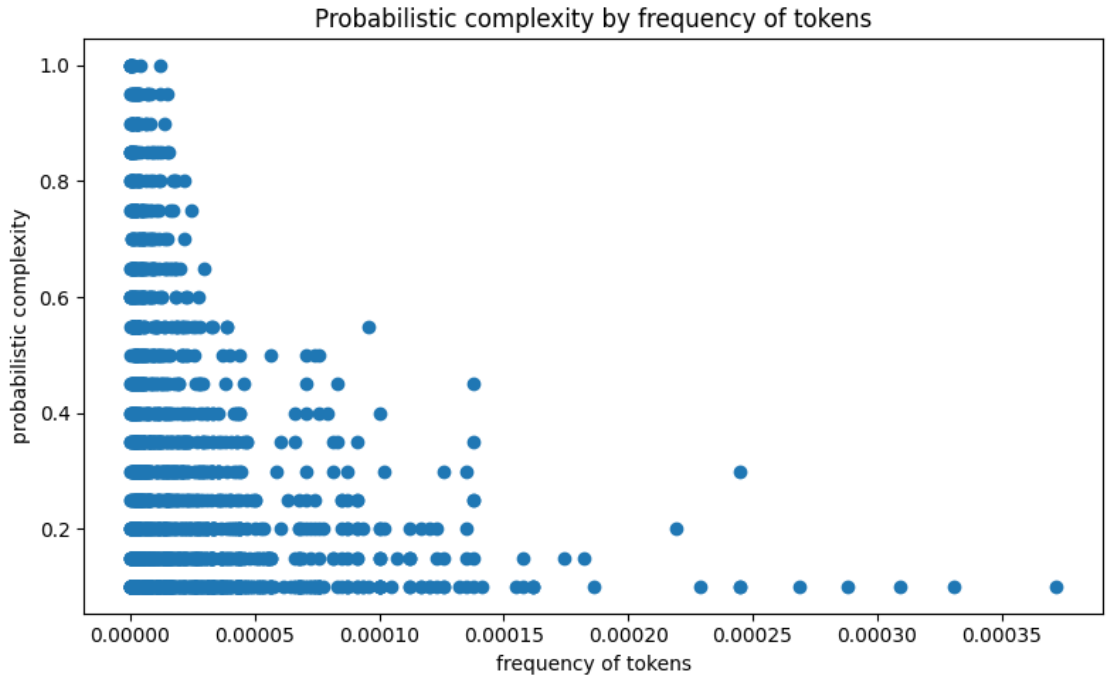
Pearson correlation length and complexity: 0.27
Pearson correlation frequency and complexity: -0.32

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.

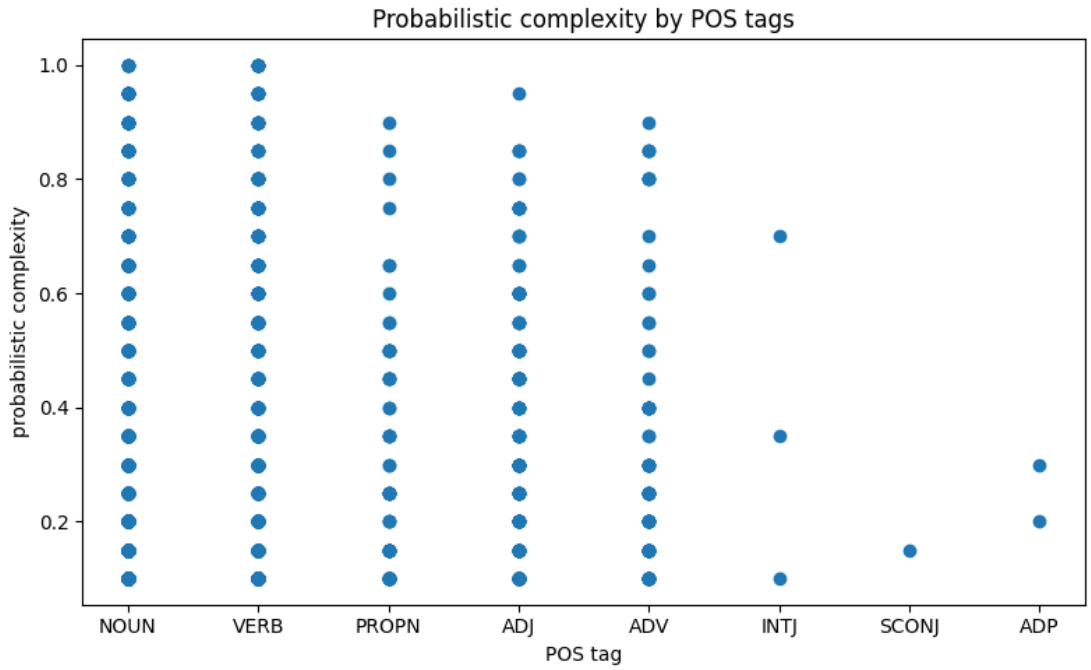
Plot 1:



Plot 2:



Plot 3:



Interpret the results in 3-5 sentences:

There is no statistically significant correlation between length of token and probabilistic complexity, though we can see that the tokens with lengths higher than 7 characters have a high probabilistic complexity, Again, no statistically significant correlation between frequency of token and probabilistic complexity, though there is a clear relationship between the two, high frequency tokens having low probabilistic complexity.

The plot for probabilistic complexity by frequency of tokens shows no correlation, but there is a clear relationship between the two, high complexity being displayed for low frequency tokens (as it would make sense – words that are not used very often have higher change of being marked as complex words).

The scatterplot for probabilistic complexity by POS tags is not very informative, except that subordinating conjunctions (SCONJ) as well as adpositions (ADP) have low probabilistic complexities; Interjections (INTJ) cluster for low to medium probabilistic complexities, while other present POS tags vary across the probabilistic complexity board.

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.

   The number of spoken syllables in a word is an intuitive characteristic that should correlate positively with the probabilistic complexity of a word. The more syllables, the more difficult a word is to pronounce/articulate and to remember/learn. Thus the complexity correlation. The consonant-to-vowel ratio based on the idea that consonants are considered more complex than vowels(Shankweiler and Harris (1966)), being harder to master from the early stages of learning.

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: 0.055

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

Baseline	Accuracy on dev	Accuracy on test
Majority	0.85	0.80
Random	0.50	0.50
Length	0.88	0.85
Frequency	0.85	0.80

Interpret the results in 2-3 sentences

We ended up with 8 for the length threshold and 0.055 for the frequency threshold after running the threshold experiments. These thresholds yielded the highest accuracies. As we can see in the table above, accuracy is lowest for random on both dev and test (0.50). This makes sense, since we would expect random to perform at 50% chance level. Best performing results were obtained for length baseline, followed closely by majority and frequency baselines. Even so, in the next analyses we will see that accuracy is doubtful for showing correct performance results for the random and frequency models.

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

For each baseline a file named model_output.tsv will be stored in directory experiments/model_name. This output file will be used later on in exercise 12 in order to compute more advanced metrics.

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 parses the sentences and corresponding labels from directories train, val and test and based on these 3 separate files creates:
    - words.txt that will contain a vocabulary of unique words found in the 3 parsed sentences datasets

- tags.txt that will contain a vocabulary of unique tags or classes found in the 3 parsed labels datasets
- dataset_params.json will contain the 3 datasets properties in a JSON format

The possibility to keep in the vocabularies just the most frequent words or tokens exists, but given the default arguments of 1, all are kept. PAD and UNK words are appended to the word's vocabulary.

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

The following layers are being used:

- Embedding layer*:* maps a sequence of word indices (each token) to embedding vectors (a params.embedding_dim vector). The purpose of this layer is to learn the word embedding during training.
- LSTM: applies a multi-layer long short-term memory (LSTM) RNN to an input sequence. For each element in the input sequence, each layer computes a function output that is the hidden state at time t. Because this is a multilayer LSTM, the input of the current layer is the hidden state of the previous layer multiplied by a dropout variable.
- Fully connected layer: helps convert the LSTM output for each token to a distribution over NER tags. This applies a linear transformation to the incoming tokens from the LTSM by multiplying them with a weight matrix and applying a bias. The result is an output tensor; in our case the NER tags.

How could you change the loss function of the model?

Because we are dealing with a binary classification task, our output layer can be the standard sigmoid (where the output represents the probability of a word to be considered normal 'N'). The loss we could use would be binary cross-entropy that computes the following average:

$$Loss = -\frac{1}{output\ size} \sum_{i=1}^{output\ size} y_i * \log \hat{y}_i + (1 - y_i) * \log (1 - \hat{y}_i)$$

where $\hat{y}_i$ is the ii-th scalar value in the model output, y_i is the corresponding target value, and output size is the number of scalar values in the model output.

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	0.79	0.46	0.58	0.20	0.52	0.28	0.52	0.43
Majority	0.80	1.00	0.89	0.00	0.00	0.00	0.71	0.44
Length	0.85	0.99	0.91	0.84	0.31	0.45	0.82	0.68
Frequency	0.80	1.00	0.89	0.00	0.00	0.00	0.71	0.44
LSTM	0.88	0.94	0.46	0.68	0.49	0.28	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.

The performance results in the shared task from the report are discussed based on macro average F1 score; thus, the same measure was computed for the models in exercise 12. The LSTM model is by far the best performer, with an F1 score of 0.74, comparable to the results obtained by the hu-berlin systems based on multinomial Naïve Bayes classifiers, placing it somewhere around rank 23 of 33 (or better than ~30% of the evaluated models in the shared task).
Looking at the available instances in the sampled training (7746 instances) and test sets (1287 instances), these are roughly 3.5 times less than in the shared task reported, the power of the results is though lower. With larger datasets, the results could possibly look much better. The length model has a performance of 0.68, lower than the 0.71 F1 score of the shared task baseline. Random, majority and frequency models all have bad performance, with F1 score around 0.44.

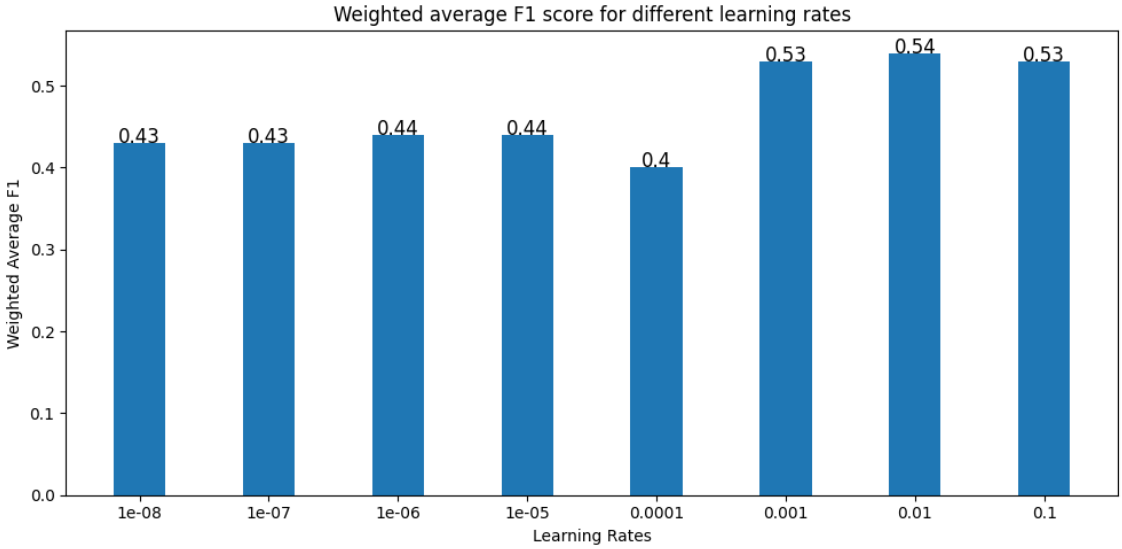
14. **Experiments** (2 points)

For this exercise, we (group 40) used the experiments.py file. Please run this file when checking.

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: Learning Rate

Plot:



Interpret the result (2-4 sentences):

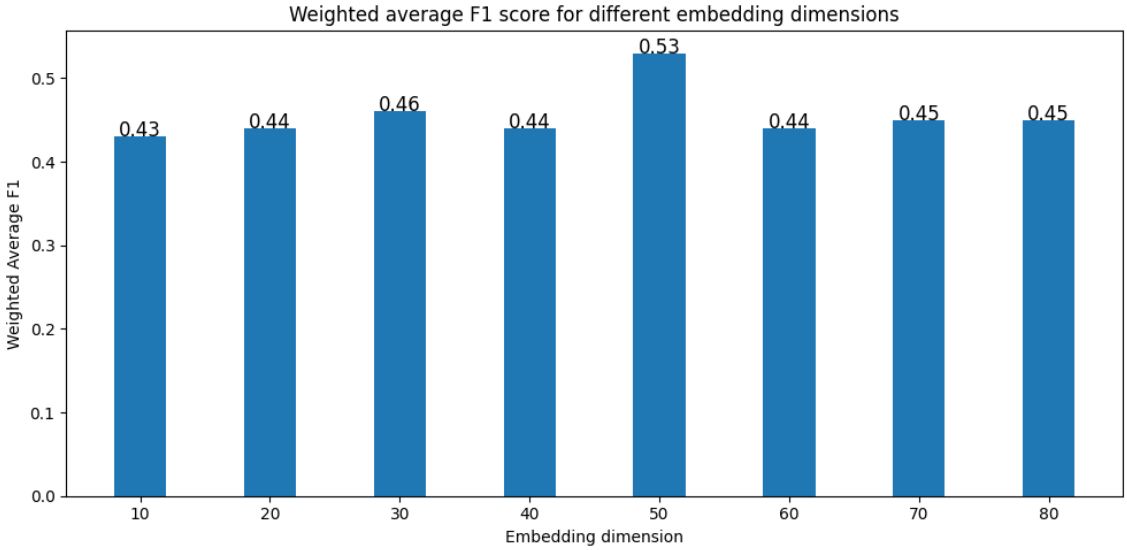
Learning rates between 0.1 and 1e-08 were explored. Best results were obtained for a learning rate of **0.01** when using training dataset and dev dataset for testing. The value is very close to its neighbors, 0.001 and 0.1. As a learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, and a learning rate that is too small can cause the process to get stuck, choosing a learning rate of 0.001 seems like a good decision, even if this is not the best value found in our experiment.

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

learning_rate - 0.0001 vs 0.01

1. house, N, C
2. met, N, C
3. gunfire, N, C

Hyperparameter: Embedding dimension
Plot:



Interpret the result (2-4 sentences):

Contrasting the result with the one obtained for searching for the optimal learning rate, in this experiment the differences between the 8 plotted embedding dimensions clearly point towards a winner (50 embedding dimensions with 0.53 F1). All the other values tested range between 0.43 and 0.46 weighted average F1, significantly lower than the winning value.

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

1. happen, N, C
2. repeatedly, N, C
3. Reportedly, N, C

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.