IN4080 – 2023
Mandatory assignment 1

Thursday 7 September 2023

Submission deadline: Monday 25 September 2023 23:59 CEST

General requirements

We assume that you have read and are familiar with IFI's requirements and guidelines for mandatory assignments:
https://www.uio.no/english/studies/examinations/compulsory-activities/mn-ifi-mandatory.html
https://www.uio.no/english/studies/examinations/compulsory-activities/mn-ifi-guidelines.html

Note that any use of pretrained language models (such as ChatGPT) is forbidden for this assignment.

This is an individual assignment. You should not deliver joint submissions.

You may upload several submissions to Devilry, but only the last submission will be evaluated. Therefore, make sure to include all files in the last submission. If you submit more than one file, put them into a zip-archive and name your submission as follows: **<username>_in4080_mandatory_1**

Your submission should contain:

- One or several Jupyter notebooks containing your code, and where you answer the text questions in markup.
- A PDF version of the notebook(s) where all the results of the runs are included.

The assignment consists of three parts with a total of 100 points. You are required to get at least 60 points to pass. It is more important that you try to answer all questions rather than that you get everything correct.

Part 1 – Conditional frequency distributions (35 pts)

The NLTK book, chapter 2, shows how the frequency of modal verbs differs across different genres:
https://www.nltk.org/book/ch02.html#brown-corpus

Here, we will conduct a similar experiment, focusing on the distribution of masculine pronouns (*he, him*) and feminine pronouns (*she, her*) across genres.

a. Conduct a similar experiment as the one mentioned above with the genres: *news, religion, government, fiction, romance* as conditions, and occurrences of the words: *he, she, her, him*, as

events. Make a table of the conditional frequencies and deliver code and table.
Note: you may use the tools from NLTK or create a Pandas dataframe, whichever you prefer.

b. Describe your results. How does gender vary across genres?

Maybe not so surprisingly, the masculine forms are more frequent than the feminine forms across all genres. However, we also observe another pattern. The relative frequency of *her* compared to *she* seems higher than the relative frequency of *him* compared to *he*. We want to explore this further and formulate a hypothesis, which we can test.

Hypothesis: The relative frequency of the object form, *her*, of the feminine personal pronoun (*she* or *her*) is higher than the relative frequency of the object form, *him*, of the masculine personal pronoun, (*he* or *him*).

c. First, consider the **complete Brown corpus**. Construct a conditional frequency distribution, which uses gender as condition, and for each gender counts the occurrences of nominative forms (*he, she*) and object forms (*him, her*). Report the results in a two-by-two table. Then calculate the relative frequency of *her* from *she* or *her* and compare to the relative frequency of *him* from *he* or *him*. Report the numbers. Submit table, numbers and code you used.

It is tempting to conclude from this that the object form of the feminine pronoun is relatively more frequent than the object form of the masculine pronoun. Beware, however, that *her* is not only the feminine equivalent of *him*, but also of *his.* So what can we do?

Luckily, the Brown corpus is annotated with part-of-speech tags, and we can use these tags to distinguish the two functions of her. Chapter 5 of the NLTK book shows how to access the tags of the Brown corpus:
https://www.nltk.org/book/ch05.html#reading-tagged-corpora
The list of part-of-speech tags can be consulted here:
http://korpus.uib.no/icame/manuals/BROWN/INDEX.HTM#bc6

d. Use the tagged Brown corpus to count the occurrences of *she, he, her, him* as personal pronouns and *her, his, hers* as possessive pronouns. Report the results in a two-ways table.

e. We can now correct the numbers from point (b) above. What percentage of the feminine personal pronoun occurs in nominative form and in object form? What are the respective percentages for the masculine personal pronoun?

f. Illustrate the numbers from (d) with a bar chart.

g. Write a short text (200-300 words) and discuss the consequences of your findings. Why do you think the masculine pronoun is more frequent than the feminine pronoun? If you find that the there is a different distribution between nominative and object forms for the masculine and the feminine pronouns, why do you think that is the case? Do you see any consequences for the

development of language technology in general, and for language technology derived from example texts in particular? Make use of what you know about the Brown corpus.

Part 2 – Zipf's law of abbreviation (30 pts)

In an earlier exercise set, we have considered Zipf's law of frequency, which states that the frequency of a word is roughly proportional to its rank in the frequency list. Here, we look at Zipf's law of abbreviation, which is another of Zipf's law. It states that the frequency of a word is negatively correlated with its length in characters, i.e. short words tend to be frequent and long words tend to be rare. It may be useful to start by reading the Wikipedia page of this law: https://en.wikipedia.org/wiki/Brevity_law

We will use the text *Tom Sawyer* from Project Gutenberg. Download the raw text from this page: https://www.gutenberg.org/ebooks/74

a. You will see that the downloaded text contains a preamble and a long appendix about the Gutenberg project and copyrights. Load the file with Python and remove these parts so that you only use the book text.

b. Tokenize the text, remove the punctuation marks, and produce a frequency distribution of the word lengths. Format this distribution as a Pandas dataframe.

c. What are the five most frequent word lengths? How long are the longest words of the text?

d. Order the table by word length and produce a plot that shows the frequencies.

e. Also produce a visualization with the data ordered by decreasing frequency.

f. When dealing with word frequency data, it is often recommended to plot frequencies on a logarithmic scale. How does this change the plots?

g. How well does this dataset match Zipf's law of abbreviation? In your opinion, which plot is most suitable to prove or disprove the law of abbreviation? Answer in text.

h. Select one word length (maybe there are some outliers in the plots that you want to know more about?) and investigate in detail which words of this word length occur in the text. Are these words specific to the Tom Sawyer text, or would you expect them to occur similarly frequently in other English texts? Or is there any evidence of preprocessing (e.g. tokenization) errors?

Part 3 – Identifying dialogue act types in chat messages (35 pts)

In this exercise, we will use the *NPS Chat Corpus* (which comes with NLTK) to experiment with different text classification methods such as Naïve Bayes and Logistic Regression.

The following code loads the chat data into a Pandas dataframe:

```
import nltk
nltk.download("nps_chat")
from nltk.corpus import nps_chat
import pandas as pd

data = []
for f in nps_chat.fileids():
    posts = nps_chat.xml_posts(f)
    for p in posts:
        data.append((p.get('class'), p.text))

df = pd.DataFrame(data, columns=['label', 'text'])
print(df.head(20))
```

Let us get an idea of the dataset and its annotation.

a. Answer the following questions:
   - How many distinct labels are there, and how many instances per label?
   - Try to understand what the labels mean, looking at some examples if necessary.
   - What is the average message length (in characters)?
   - How do these numbers compare to the Movie Reviews corpus used in the last exercise set?
   - Explain in a few sentences how these differences may impact the text classification performance. Which text classification methods or parameter settings do you expect to be better adapted to the NPS Chat problem?

Replicate the multinomial Naïve Bayes system from the last exercise set. Concretely:

- Split the data into training and test data. To simplify things, we will just use a two-way split of 90% training data and 10% test data in this first experiment.
- Instantiate a CountVectorizer with default parameters and fit it to the training data.
- Instantiate a MultinomialNB model and fit it to the training data.
- Predict the test set labels using the trained model and display the classification report.

b. Write a few lines about your observations: Which classes are the easiest / most difficult to predict? Are there classes where precision differs drastically from recall, and if so, what does this mean?

For the following experiments, we are going to leave aside the test set, and we are going to use five-fold cross-validation on the training data to compare the different models. Train the same model again and provide the cross-validation scores. Cross-validation is handled semi-automatically by Scikit-Learn. Initialize the CountVectorizer and the classifier model as before and use the following code to produce the cross-validation scores (cv determines the number of folds):

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(classifier, x_train_bow, y_train, cv=5)
print("{:.3f} accuracy with a standard deviation of
{:.3f}".format(scores.mean(), scores.std()))
```

Note that this code does not use the test set at all. We will return to it at the end. You are now asked to change one parameter at a time, run the cross-validation again, report the scores, and decide whether to keep the change or not for the following model. (If you do not manage to implement one option, just skip it, and continue with whatever best model you have.)

c. Produce the cross-validation scores with the initial Naïve Bayes model.

d. By default, the CountVectorizer lowercases all input and uses a simple whitespace-based tokenizer. Check if other settings provide better results. Note that NLTK provides a TweetTokenizer which might work best for user-generated content:
https://www.nltk.org/api/nltk.tokenize.casual.html
Feel free to experiment with other parameters of the CountVectorizer.

e. By default (and as its name implies), the CountVectorizer produces frequency counts. Evaluate the impact of binary feature values (presence or absence of a word in an instance). There is a second way to include binary features, namely by using the BernoulliNB model instead of the MultinomialNB one. What is the difference between the binary multinomial model and the Bernoulli model (both in terms of scores and theoretically)?

f. We know that Logistic Regression may produce better results than Naive Bayes. We will see what happens if we use Logistic Regression instead of Naive Bayes on this task. Keep the same CountVectorizer as before and replace the Naïve Bayes classifier by a Logistic Regression one with default parameters:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

You may have to revisit the count/binary features distinction and the case folding parameter as this is a different type of classifier.

g. Evaluate different regularization settings (L2, L1, different values of C). The supported types of regularization depend on the Solver used for gradient descent. The documentation provides more details about the different available solvers:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Note that the solvers provided with Scikit-Learn determine the learning rate on their own, so there is not parameter to set a specific learning rate.

You should now have identified a model setup that produces the best cross-validation accuracy.

h. Retrain the best model on the entire training set and evaluate it on the test set. Report the detailed evaluation scores and compare them with the initial Naïve Bayes scores. On which classes did your model improve the most? Which classes are still difficult to predict?