Natural Language Processing Fundamentals

Elisa Schaeffer

Fall 2022

1 Course content

Session	**Topic**	*Tool(s)*
1	Fundamental Concepts	Python & R
2	Topic Detection	R
3	Sentiment Analysis	R
4	Tagging	Python & R
5	Word Networks	Python
6	Correction & Prediction	Python
7	Stemming	Python
8	Vectorization	R
9	Chatbots	Python
10	Speech	Python

Official course information website (sorry for the URL): `https://continuingstudies.mcgill.ca/search/publicCourseSearchDetails.do?method=load&courseId=9607046`

GitHub repository of example codes and book links: `https://github.com/satuelisa/NLPF`

2 Evaluation

Item	% of total grade	*Questions / points*
Participation	10	10×1
Weekly reflection 1	9	9×1
Weekly reflection 2	9	9×1
Weekly reflection 3	9	9×1
Weekly reflection 4	9	3×3
Weekly reflection 5	9	3×3
Weekly reflection 6	9	$2 \times 4 + 1$
Weekly reflection 7	9	9×1
Weekly reflection 8	9	$3 \times 2 + 1 \times 3$
Weekly reflection 9	9	$1 \times 3 + 3 \times 2$
Final reflection	9	9×1
TOTAL	100	

There is no mandatory minimum attendance required to pass the course. There are no examinations, neither midterm nor final.

The **weekly journal of self-assessed participation** in each session is uploaded as a 10-page PDF file in which each page corresponds to the free-form notes indicating how the student participated in that session. They can include a scan or a photograph of hand-written notes as well and accommodate many pages of notes on a single page of the PDF with a multi-page layout if they so wish. A blank page is used to indicate those weeks on which they did not participate. The participation journal is submitted before the end of the work week of the last session.

In the **ten weekly reflections**, the participants carry out tasks of increasing complexity, using the computational tools employed in the examples of that week's session, and answer a quiz on *myCourses* based on the results they obtain. The tasks are posed in two variants: **experiential** (the participant applies the new concepts in their work and discusses the outcomes) and **academic** (the participant extends the in-class examples as instructed and reports the findings). The reflections for weeks 1–9 are submitted by midnight three working days after the next session. For each one, the number of questions and the amounts of points per question is summarized above.

3 Weekly reflection questions

McGill University values academic integrity. Therefore, all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the McGill Code of Student Conduct and Disciplinary Procedures.

See `https://www.mcgill.ca/students/srr/honest/` for more information.

The responses are submitted as a quiz on *myCourses*. Please prepare all the responses before starting the quiz. The optional readings as indicated on *myCourses* may prove helpful in answering some of the questions, as will attendance to the class sessions.

You are always allowed to "upscale" the assignments, if you happen to have computing resources at hand (such as a powerful GPU) and applicable former knowledge (of ML techniques, for example) that you can incorporate in carrying out the exercises by elevating, expanding, or generalizing the objective. When you do so, however, please include an *introductory paragraph* in your response that explains **how** you have revamped the question and **why** you chose to do so.

3.1 Week 1 reflection

Working with the two example codes, either with example data such as the book downloaded from Gutenberg in the examples or with text data of particular interest to you, please answer the following questions briefly.

1. What *data* did you use? If it is a book from Gutenberg, which one?
2. Can you estimate how many *sentences* it contains? What can you do to compute this (conceptually, in Python and/or in R)?
3. How about *paragraphs*? Can you estimate how many there are? What can you do to compute this (conceptually, in Python and/or in R)?
4. How many *names* (of places and people) are mentioned in the text? What did you do to compute this (conceptually, in Python and/or in R)?
5. What are the ten most frequent words in the text that you consider to clearly be **stop words**?
6. What are the ten most frequent words in the text that you consider to **not** be stop words?
7. How would you go about *automatically* identifying **potential stop words** for documents written in a language you **do** not speak?
8. Looking at a *histogram* of the word frequencies (like the horizontal bar chart in the R example code), what can be said about the shape of the **distribution**?
9. Please show an example of a **word cloud** you created from your text along with a code snippet of how you did it.

   You may censure any potentially confidential terms that appear in your data if you are using workplace experiential learning and the data set you used is not in open data.

   Also feel free to comment on any *insights* you have gained of the data while carrying out these exercises as well as your initial feeling on the usefulness of Python and R in NLP.

3.2 Week 2 reflection

1. We saw that the frequencies of word appearances follow a Zipf's law. How about the distribution of the *lengths* of the words (that is, the number of characters per word)?

   How did you obtain that information (in R or in Python)? What can you say about the *shape* of the distribution?

   Please include a *visualization* of the distribution and a **code snippet** in your response.

2. How about the distribution of the tf-idf **weights** themselves? Does that follow Zipf's law?

   Please include in your response all of the following:

   (1) a *code snippet* to obtain and visualize that distribution

   (2) the resulting *image*

   (3) your *interpretation* of what the distribution is like

3. How could you compute a *similarity measure* for **documents** based on the `tf-idf` matrix?

   Please show a code snippet and discuss your proposal.

4. How could you compute a similarity measure for **terms** based on the `tf-idf` matrix?

   Please show again a code snippet and discuss your proposal.

5. Please investigate state-of-the-art *alternatives* for LDA and provide here the DOI (if applicable, URL if there is no related DOI) and a very brief critique of at least one alternative that you consider promising.

6. Please carry out LDA or another topic-detection method of your choice on your data.

   Include in your response the **code snippet** and a listing of *how many tokens were associated to each topic*.

7. Propose (either based on creativity alone or looking up alternatives) and implement an alternative way to **visualize** the topics (other than the comparison cloud created in the example code).

   If you base your creation on some consulted sources, please clearly cite these sources in your response. Also, as before, include a code snippet and an image of the resulting visualization.

8. How would you go about using `tf-idf` to *build a list of stop words* from scratch when none is available? Are there any *language-specific* concerns in the approach you are envisioning?

   Your response may either be fully conceptual (pseudo-code is encouraged) or include a code snippet in R or Python. Again, please clearly cite sources if you consulted any.

9. Based on both your own experience thus far and on what you have encountered until know or can scrape up online just to answer this question, what do you perceive to be the **strengths** and **weaknesses** of Python and R as NLP tools in particular?

   Are you aware of any *other* tools that would be competitive for NLP tasks at present?

3.3 Week 3 reflection

1. How would you go about **combining** the three lexicons with different ways to *quantify* sentiments into a single one, conceptually? (You can implement it if you wish, but this is not required.) What would you do with words that appear in two or more lexicons?

2. Pick a text repository of your choice (you can use the three set of reviews of the example code to take the short way out or pick some other open data or process something from your work).

   What *percentage* of the words (after cleaning out stop words) of the repository are **present** in each of the three sentiment lexicons?

3. **Aggregating** over the three lexicons in some way you see fit, what are the top-ten most *positive* words and the top-ten most *negative* words in your input data?

   Please include a **code snippet** to show how you arrived at this result.

4. Think of a way for quantifying the *range* of sentiments expressed in each *document* of a multi-document corpus in terms of at least one the three sentiment lexicons used in the example code.

   Describe your approach conceptually, accompanied by a **code snippet** that implements this idea.

5. What *percentage* of the documents (whatever sub-elements of the text repository you choose to consider as documents in a sense) could be considered either **neutral** or **mixed** (in the sense that they are neither clearly positive nor clearly negative)?

   Please include a **code snippet** that determines this.

6. How would you *visualize* the results of such a sentiment *quantification* over a corpus? Again, first discuss and then, if you wish, attach a code snippet.

   If you do not write code, you can draw a conceptual visualization by hand to communicate your approach.

7. Do you expect a sentiment quantification of *product reviews* to somehow **(cor)relate** to the associated star/numerical *ratings*?

   You may either try this out in code and interpret the results or speculate conceptually on what you would expect to find.

8. Search on Google Scholar for recent *contributions to the state of the art* on sentiment analysis within the field of NLP.

   Please provide the DOI for three that you find interesting and briefly describe what the authors propose.

9. Now, search online for existing **applications** of NLP sentiment analysis.

   Please provide the URL and a brief description of each, indicating who applies it (company, university, government) and to what exactly (the inputs, the output, and the context in which it is applied).

3.4 Week 4 reflection

If you find the questions challenging, taking a look at the optional readings may prove helpful. Please again cite all sources you consult, even when you do not directly copy anything from the source, and include code snippets in your answers where applicable as well as images to visualize your results.

This time each question is worth **three points** as the tasks are of a wider scope now.

1. How does the **precision** of an n-gram tagger behave as you *increase* the value of n from one to k where $k > 3$ is the value of your choice (depending on the computing resources you have at hand).

   You are free to choose your own corpus (it does not have to be brown like in the examples).

2. What is the effect on that precision when **sentence breaks** are taken into account versus when they are ignored?

   (See the section *Tagging Across Sentence Boundaries* in the Python textbook, Chapter 5).

3. Compare the **accuracy** and the **training time** of a *non-transformer tagger* to a *transformer tagger* (each one of your own choice) on at least **three** different corpora (again or your choice).

3.5 Week 5 reflection

In this reflection, we ponder upon the potential of WordNet-like information in NLP. With the input data and tool set of your choice within the Python/R ecosystem, we will implement a couple of ways to transform and analyze text.

Instead of limiting to English and/or French, you can choose to carry this exercise out in using text data in *any* language that Google Translate or some similar free-to-use service is roughly able to translate. Simply accompany your example texts with an automated translation.

For the following languages, it is *not necessary* to include the automated translations in your responses as the instructor is able to read them: English, French, German, Spanish, Portuguese, Italian, Swedish, Norwegian, Danish, Finnish.

Again, each question is worth **three points**.

3.5.1 Question 1

Based on a word network, write a program that takes a sentence and exchanges each word to another one with a similar meaning, if one exists. You are free to exclude stop words and names from this transformation.

Please include a code snippet and at least three example inputs with their respective outputs in your response.

Discuss how legible (in the sense of easy to understand) you find the transformed texts in comparison to the originals.

3.5.2 Question 2

Now, based on the response to the previous question: design a program that assigns a numerical similarity score to two input sentences in terms of how similar they are with respect to where the words are within the conceptual hierarchies in `WordNet`.

Instead of using just unigram-level similarity, try to incorporate n-gram aspects of assigning a higher similarity to texts that contain sequences of words that are all similar to one another, lowering the similarity whenever this breaks.

For example *a small dog that is hungry* is very similar to *a petite canine who runs* in the first four words but then differs at the end.

Again, please provide a code snippet and examples (you can reuse the inputs and the outputs of the previous question as examples in this question).

3.5.3 Question 3

Using the *Open Multilingual Wordnet* at `http://compling.hss.ntu.edu.sg/omw/`, write a program that takes as input a sentence along with information about what language this sentence is written in and what language to translate it to, and then, using `WordNet` to map concepts, write a very rough automated translator.

Provide a **code snippet**, **input-output examples**, and a *discussion* on the aspects of language translations that are hard or impossible to capture just using a `WordNet` as a knowledge base.

3.6 Week 6 reflection

The first two questions are worth four points each and the last one is worth one point.

3.6.1 Question 1

Build a **auto-corrector** based on a (small) vocabulary of "known words" extracted from a text repository of your choice that takes as input a sentence (with possible misspelled words) and *replaces* each of the words with one from the vocabulary that *minimizes* the **edit distance**.

Please cite your sources, show your code, and include some input-output examples.

Discuss what other techniques (that we have discussed in previous sessions or that you can think of otherwise) could be used to make this simple auto-correct *perform better* in terms of inferring the intended *meaning* of the word or to take into account similarities in *pronunciation* between differently-spelled words.

3.6.2 Question 2

Using either some n-gram based or another type of approach (remember to cite any sources you consult) and a text repository of your choice, implement a simple **auto-complete** that suggests possible options for what the *next* word could be, given a start of a sentence as input.

Please include code snippets and examples, as usual.

Discuss how the value for n affects the *quality* you observe (subjective or measured). Would you actually need a **range** of values for n instead of a single value for this to work well?

3.6.3 Question 3

Modify your auto-complete so that it *never* suggests **names** of people or places.

Include a code snippet and examples in your response.

3.7 Week 7 reflection

Remember to cite all sources and show your code. Images that visualize your results are also welcome. We have nine questions, one point each.

1. How does the `tf-idf` **weight distribution** (computed for Question 2 on Week 2) differ between non-stemmed and stemmed versions of one same set of documents? Is there a visually identifiable difference (in their histograms)? Can a *statistical test* tell them apart?
2. What is the effect of the presence/absence of stemming in the proposed term/document **similarity measures** (Questions 2 & 3 of Week 2)? You can try it out and/or speculate about it on a conceptual level.
3. What happens to the proportion of words found in the *sentiment* lexicons (used on Week 3) when stemming or lemmatization is carried out versus when it is not? Do you expect this to affect NLP systems that seek to identify or quantify feelings?
4. What is the effect of using or not using stemming or lemmatization in the precision of n-gram taggers (Question 1 of Week 1)?
5. How are the transformer-based taggers affected by stemming or lemmatization (Question 3 of Week 4)?
6. In order to use a similarity quantifier (like the one of Question 2 of Week 5) to detect suspected *plagiarism*, would you expect this to be more useful for this purpose or less so if lemmatization or stemming were used *explicitly* before querying for `WordNet` similarities?
7. Would *text prediction* (like the auto-complete of Question 2 of Week 6) be easier in some way if stemming or lemmatization were included in some part of the process?
8. Are there languages that are harder or easier to stem than English? What linguistic structures affect this?
9. Are there languages in which even *segmentation* (splitting a paragraph into sentences and sentences into words) is challenging?

3.8 Week 8 reflection

Remember to cite all sources and show your code in your responses. Images that visualize your results are also welcome.

The first three questions are two points each, the last one is worth three points.

1. How would you incorporate other vectorizations than `tf-idf` into your proposed **document similarity measure** (Question 3 of Week 2)?
2. How could one make use these other vectorizations to improve *sentiment analysis* by considering longer sequences within the text just than single words?
3. Would *text prediction* (like the auto-completion of Question 2 of Week 6) be easier in some way if these other vectorizations are taken into consideration? How and why?
4. Compute and compare (k-means or other) clusterings (for a corpus of your choice) based on our three vectorizations — `tf-idf`, PPMI, and skipgram embeddings — in terms of some consistency measure such as the *Rand* or the *Jaccard index*. Discuss the differences you observe. Does the spacial grouping, in any of the three cases, appear to capture some sort of perceivable semantic similarity?

3.9 Week 9 reflection

The first question is worth three points and the other three are worth two points each.

Remember to cite your sources — whether they be code examples or readings of another sort. Looking up scientific literature through Google Scholar or similar may be fruitful for the last two questions.

1. Combine a rule-based and a machine-learning chatbot (like those of the example code), using rules and training data of your choice (you may retain the example setup if you wish) to implement a bot that answers clear-cut situations based on rules and defaults to the responses from a fitted neural network when no rule matches. Interact with it and discuss the strengths and the weaknesses of your bot. Please include code snippets and an example dialogue in your response.

2. Does it sound like a good idea to *continue* training a chatbot based on the **reactions** that its responses elicit from the users in a live environment? What would you gain by doing this? What could go wrong? In what context could it make sense to do this?

3. How about training a chatbot in a completely *non-supervised* manner (meaning that no labeled data is ever used)? Does this sound feasible? What could cause difficulties with such an approach?

4. Find some chatbots (at least three) in the customer service options of the websites or social media of some companies or organizations that interest you. Test them out for a few minutes and write a brief review of each, speculating on how you imagine them to be implemented and if you can think of ways to improve them.

3.10 Final reflection

After the final session, the participants reflect, in writing, upon the *applicability* of the concepts and tools covered throughout the course for their present and potential future professions as well as provide feedback on the overall structure and contents of the course. Remember to cite your sources, if any.

The final reflection is submitted **before the end of the work week of the last session**.

1. What kind of **contextual** reasoning do you expect to be needed to choose the correct *pronunciation* to a word that has several options depending on the intended meaning (like research, where the verb and the noun are pronounced differently) when converting text to speech?

2. When converting speech to text, one needs to map the sound to a word, which becomes problematic when two words that are *spelled* differently *sound* (exactly or almost) the **same**. Is this essentially much like the task that we addressed in the first question or do you perceive relevant differences between the two scenarios? Please discuss.

3. If you were to convert a *text-based* chatbot into a *speech-based* one, do would you expect to be able to simply plug it into text-to-speech and speech-to-text modules and be done with it or do you foresee some other complexities in achieving that? Please discuss.

4. Based on what you know now about NLP, what do you expect to be the difficulties involved in automated *translation*? Do the difficulties differ between translating **spoken** and **written** language? (See Chapter 10 of SLP for context.)

5. Did you use **data sets** that differ from the ones employed in the example in any of the assignments during this course? Would you have liked to do that more? Would you make any different choices if you were to start over with the course work?

6. How and when do you expect to **apply** NLP in your work? Do you expect to be able to apply any of the coursework in the future? If so, which parts and how?

7. Do you expect the exposure to the technical fundamentals of NLP covered in this course to be useful to you if you choose to at some later point dive into the more *advanced* linguistic, mathematical, and machine-learning aspects involved in NLP?

8. What aspects of NLP do you feel like you still need to learn (more) about? How and when do you hope to do that?

9. How did the contents of the course, in retrospect, match your initial expectations? Do you feel that something should be added or removed in future iterations of the course? Would you alter the order in which the topics are discussed or the level of detail?