LING/COMP 445, LING 645
Problem Set 2

Jan Tiegges: , **261180937**:
Cerys Jenkins:

Due before 4:35 PM on Wednesday, September 27, 2023

Please enter your name and McGill ID above. There are several types of questions below.

- For questions involving answers in English or mathematics or a combination of the two, put your answers to the question in an answer box like in the example below.
- For programming questions, please put your answers into a file called `ps2-lastname-firstname.clj`. Be careful to follow the instructions exactly and be sure that all of your function definitions use the precise names, number of inputs and input types, and output types as requested in each question.

  For the code portion of the assignment, **it is crucial to submit a standalone file that runs**. Before you submit `ps2-lastname-firstname.clj`, make sure that your code executes correctly without any errors when run at the command line by typing `clojure ps2-lastname-firtname.clj` at a terminal prompt. We cannot grade any code that does not run correctly as a standalone file, and if the preceding command produces an error, the code portion of the assignment will receive a 0.

  To do the computational problems, we recommend that you install Clojure on your local machine and write and debug the answers to each problem in a local copy of `ps2-lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

  Note there is a built-in function called `reverse`. **Do not use the built-in function `reverse` in this problem set!** We remove `reverse` from the namespace when we grade, so using it anywhere will lead to an error and a result in a 0.

Once you have entered your answers, please compile your copy of this LaTeX file[1] into a PDF and submit

(i) the compiled PDF renamed to `ps2-lastname-firstname.pdf`
(ii) the raw LaTeX file renamed to `ps2-lastname-firstname.tex` and
(iii) your `ps2-lastname-firstname.clj`

to the Problem Set 2 folder under 'Assignments' on MyCourses.

Example Problem: This is an example question using some fake math like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

Example Answer: Put your answer in the box provided, like this:

$\boxed{\text{Example answer is } L = \sum_0^\infty \mathcal{G}\delta_x.}$

[1] To compile a file `file.tex` to `file.pdf`, you can use the command `pdflatex file.tex` at the command line, or make use of an online service such as `https://overleaf.com`. You can find more information about LaTeX here `https://www.latex-project.org/`.

MAIN PROBLEM SET

Problem 1: Write a single-argument function called `absval` that, when passed a number, computes its absolute value. It should do this by finding the square root of the square of the argument. (Note: you should use the `Math/sqrt` function built in to Clojure (from Java), which returns the square root of a number.)

Answer 1: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 2: In both of the following definitions, there are one or more errors of some kind. In each case, explain what's wrong and why, and fix it:

```
(defn take-square
  (* x x))

(defn sum-of-squares [(take-square x) (take-square y)]
  (+ (take-square x) (take-square y)))
```

Answer 2: Please put the fixed functions in `ps2-lastname-firstname.clj` and **describe what is wrong and why in the box below**.

The first function *take-square* is missing the argument x, which is required for functions. The second function *sum-of-squares* is missing the argument x and y, as it is calling the function *take-square* with the arguments x and y in the input brackets, resulting in an error. These kind of operations can only be made inside the function itself, but not in the input brackets.

Problem 3: The expression `(+ 11 2)` evaluates to `13`. Write four other different Clojure expressions which also evaluate to the number `13` (either the integer `13` or the float `13.0`). Using `def`, assign these expressions to the symbols `exp-13-1`, `exp-13-2`, `exp-13-3`, and `exp-13-4`.

In each def statement, be sure to quote the expression (as below for our example), so it is not evaluated before being assigned to the symbol.

```
(def exp-13-0 '(+ 11 2))
```

Answer 3: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 4: Define a function called `third`, that selects the third element of a list. For example, given the list `'(4 5 6)` as its argument, `third` should return the number `6`.

Answer 4: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 5: Define a function called `compose`, that takes two one-place functions `f` and `g` as arguments. It should return a new function, the composition of its input functions, which computes `f` of `g` of `x` when passed the argument `x`. For example, the function `Math/sqrt` (built in to Clojure from Java) takes the square root of a number, and the function `Math/abs` (likewise) takes the absolute value of a number. If we use `defn` to define functions `sqrt` and `abs` as

```
(defn sqrt [x] (Math/sqrt x))
(defn abs [x] (Math/abs x))
```

then `((compose sqrt abs) -36)` should return `6`, since the square root of the absolute value of `-36` equals `6`.

Answer 5: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 6: Define a function called `first-two` that takes a list as its sole argument, and returns a two-element list containing the first two elements of the argument. For example, given the list `'(4 5 6)`, `first-two` should return the list `'(4 5)`.

You may assume that the list passed in has at least two elements.

Answer 6: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 7: Define a function called `remove-second` that takes a list, and returns a new list that is the same as the input list, but with the second value removed. For example, given `'(3 1 4)`, `remove-second` should return the list `'(3 4)`.

Answer 7: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 8: Define a function called `add-to-end` that takes in two arguments: a list `lst` and a value `x`. It should return a new list which is the same as `lst`, except that it has `x` appended as its final element. For example, `(add-to-end (list 5 6 4) 0)` should return the list `'(5 6 4 0)`.

Answer 8: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 9: Define a function called `reverse-list`, that takes in a list, and returns the reverse of the list. For example, if it takes in the list `'(a b c)`, it will output the list `'(c b a)`.

Do not use the built-in function `reverse` (in this problem, nor anywhere in this problem set).

Answer 9: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 10: Define a function called `count-to-1`, that takes a positive integer `n`, and returns a list of the integers counting down from `n` to `1`. For example, given input `3`, it will return the list `(list 3 2 1)`.

Answer 10: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 11: Define a function called `count-to-n`, that takes a positive integer `n`, and returns a list of the integers from `1` to `n`. For example, given input `3`, it will return the value of `(list 1 2 3)`.

Hint: Use the procedures `reverse-list` and `count-to-1` that you wrote in the previous problems.

Answer 11: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 12: Define a function called `get-max`, that takes a list of numbers, and returns the maximum value. So, `(get-max '(2 3 3))` should return `3`.

Don't use the built-in `max` function.

Answer 12: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 13: Define a function called `greater-than-five?`, that takes a list of numbers, and returns a list of equal length to the input list, but where each number is replaced with `true` if the number is greater than `5`, and `false` otherwise. For example, given input `(list 5 4 7)`, it will return the list `'(false false true)`.

Hint: Use the built in function `map`.

Answer 13: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 14: Define a function called `concat-three`, that takes three sequences (represented as lists), `x`, `y`, and `z`, and returns the concatenation of the three sequences. For example, given the arguments `(list 'a 'b)`, `(list 'b 'c)`, and `(list 'd 'e)`, the procedure should return the value of `(list 'a 'b 'b 'c 'd 'e)`.

Don't use the built-in `concat` function.

Answer 14: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 15: Define a function called `sequence-to-power`, that takes a sequence (represented as a list) `x`, and a nonnegative integer `n`, and returns the sequence $\mathtt{x}^n$. For example, given the sequence `(list 'a 'b)` as the first argument and the number `3` as the second, the procedure should return the value of `(list 'a 'b 'a 'b 'a 'b)`.

You may use the built-in `concat` function for this problem.

Answer 15: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 16: Define L as a language containing a single sequence, $L = \{a\}$.

In Clojure, we can represent the sequence a as the list `'(a)`.

Define a function called `in-L-star?` that takes a sequence (represented as a list), and returns true if and only if the sequence is a member of the language L^*. For example, given a sequence such as `'(a b)`, the procedure should return `false`, because ab is not a member of L^*.

Answer 16: Please put your answer in `ps2-lastname-firstname.clj`.

Problem 17: Let A and B be two distinct formal languages. We'll use $(A \cdot B)$ to denote the concatenation of A and B, in that order. Find an example of languages A and B such that $(A \cdot B) = (B \cdot A)$.

Answer 17: Please put your answer in the box below.

Let $A = \{a\}$ and $B = \{aa\}$. Then $(A \cdot B) = \{aaa\}$ and $(B \cdot A) = \{aaa\}$. Therefore, $(A \cdot B) = (B \cdot A)$.

Problem 18: Let A and B be languages. Find an example of languages A and B such that $(A \cdot B)$ does not equal $(B \cdot A)$

Answer 18: Please put your answer in the box below.

Let $A = \{a\}$ and $B = \{b\}$. Then $(A \cdot B) = \{ab\}$ and $(B \cdot A) = \{ba\}$. Therefore, $(A \cdot B) \neq (B \cdot A)$.

Problem 19: Find an example of a language L such that $L = L^2$, i.e. $L = (L \cdot L)$.

Answer 19: Please put your answer in the box below.

Lets take the 2-alternating language $L = \{a, b\}^*$. Then $L^2 = \{a, b\}^* \cdot \{a, b\}^* = \{a, b\}^*$. Therefore, $L = L^2$.

Problem 20: Argue that the intersection of any two languages L and L' is always contained in L.

Answer 20: Please put your answer in the box below.

This is because $L \cap L'$ is the set of all strings that are in both L and L', i.e. $L \cap L' = \{x | x \in L \wedge x \in L'\}$. Therefore, all strings in $L \cap L'$ are also in L, i.e. $L \cap L' \subseteq L$.

Problem 21: Let L_1, L_2, L_3, and L_4 be languages. Argue that the union of Cartesian products $(L_1 \times L_3) \cup (L_2 \times L_4)$ is always contained in the Cartesian product of unions $(L_1 \cup L_2) \times (L_3 \cup L_4)$.

Answer 21: Please put your answer in the box below.

$(L_1 \times L_3) \cup (L_2 \times L_4)$ results in tuples where the first string is from L_1 and the second string is from L_3, or the first string is from L_2 and the second string is from L_4, or more formally $(L_1 \times L_3) \cup (L_2 \times L_4) = \{(x, y) | (x \in L_1 \wedge y \in L_3) \vee (x \in L_2 \wedge y \in L_4)\}$. $(L_1 \cap L_2) \times (L_3 \cap L_4)$ results in tuples where the first string is from either L_1 or L_2 and the second string from either L_3 or L_4, or more formally $(L_1 \cap L_2) \times (L_3 \cap L_4) = \{(x, y) | x \in L_1 \cup L_2 \wedge y \in L_3 \cup L_4\}$. Since x can be in either L_1 or L_2 and y in either L_3 or L_4, it includes all possibilities for x and y and therefore, $(L_1 \times L_3) \cup (L_2 \times L_4) \subseteq (L_1 \cup L_2) \times (L_3 \cup L_4)$.

Problem 22: Let L and L' be finite languages. Show that the number of elements in the Cartesian product $L \times L'$ is always equal to the number of elements in $L' \times L$.

Answer 22: Please put your answer in the box below.

$|L \times L'| = |L| \cdot |L'|$ and $|L' \times L| = |L'| \cdot |L| = |L| \cdot |L'|$. Therefore, $|L \times L'| = |L' \times L|$.

Problem 23: Suppose L is a language, and that concatenation of L with itself is equal to itself: $(L \cdot L) = L$. Show that L is either the empty set, the set $\{\epsilon\}$, or an infinite language.

Answer 23: Please put your answer in the box below.

If L is the empty set, then $(L \cdot L) = \emptyset \cdot \emptyset = \emptyset$. If L is the set $\{\epsilon\}$, then $(L \cdot L) = (\{\epsilon\} \cdot \{\epsilon\}) = \{\epsilon\}$. If L is an infinite language, then $(L \cdot L) = L^* \cdot L^* = L^*$, as all strings are just concatenations of strings from L. Lets assume now L is a finite language with the string x. Then $L \cdot L$ will contain the string xx, which is not in L. This is because in a finite language, concatenating the language with itself a finite number of times would still result in a finite language, not equal to itself. Therefore, L is either the empty set, the set $\{\epsilon\}$, or an infinite language.

LONG FORM READING QUESTION:

(This section is optional for students in LING/COMP 445, but must be completed if taking LING 645.)

You must answer this question on your own.

Andreas et al. (2022) lay out a framework for how to think about data collection and ML model design when trying to study language representation. Though they use sperm whale communication as their driving example, much of the points they make are applicable to any communication system, including humans. What is the difference between supervised learning and self-supervised learning in ML models? What are some of the benefits and downfalls of using self-supervision to train a model of language representation? (give at least 2 benefits and 2 downfalls - your answer should be 1/2 a page to a page in length.)

Answer: Please put your answer in the box below.

Supervised learning is a type of machine learning in which models are trained using labeled data. This means that input data is associated with the corresponding known classification outputs in advance. The model then uses this labeled data to make predictions, with the overall goal being to minimize the error between the predictions and the actual output. In contrast, self-supervised learning is a different approach to machine learning that works with unlabeled data where the model does not know the correct outputs. In natural language processing, self-supervised models are often trained to predict certain aspects of the input data, such as the next word in a sentence or a missing word in a sentence. The goal is to develop a representation, usually a high-dimensional feature vector, that captures the underlying structure of the data.

One of the advantages of self-supervised learning is its ability to learn data representations without requiring labeled data, which can be a costly and time-consuming effort. This is especially important in cases where the true meaning of the data is difficult to assess, such as in the study of sperm whale communication, where humans cannot directly interpret the data. Moreover, self-supervised learning can develop more generalizable data representations that can be applied to a wide range of downstream tasks.

Nevertheless, self-supervised learning also has its drawbacks. A major challenge lies in evaluating the quality of the learned representations, as there is no clearly defined objective measure for assessing the model performance. In addition, self-supervised learning can require significant computational effort as the need to be trained on large amounts of unlabeled data to produce valuable representations.

In summary, the authors emphasize the potential of self-supervised learning as it learns more robust representations than supervised learning and can help to better understand the mysteries of the oceans. However, there are also drawbacks mentioned above, to which can be added that besides the cost and environmental impact of training these self-supervised models, a large data set must be available in the first place. The authors emphasize the importance of open source datasets and cross-cutting initiatives in science to really exploit the potential of this technique.