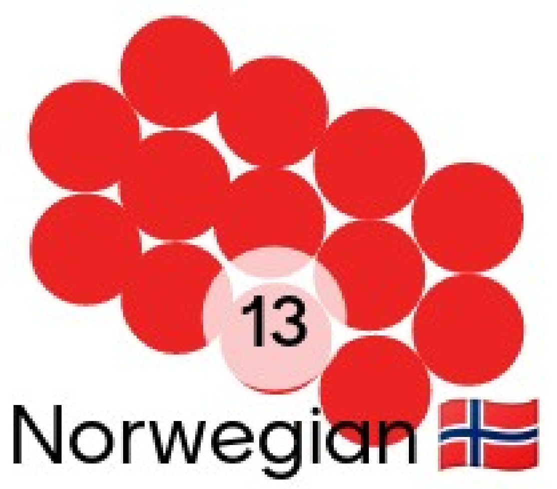
IN[34]120 - Search technology 💻📚🔍

2024-10-15 14:15 @ Chill 🎃🎃

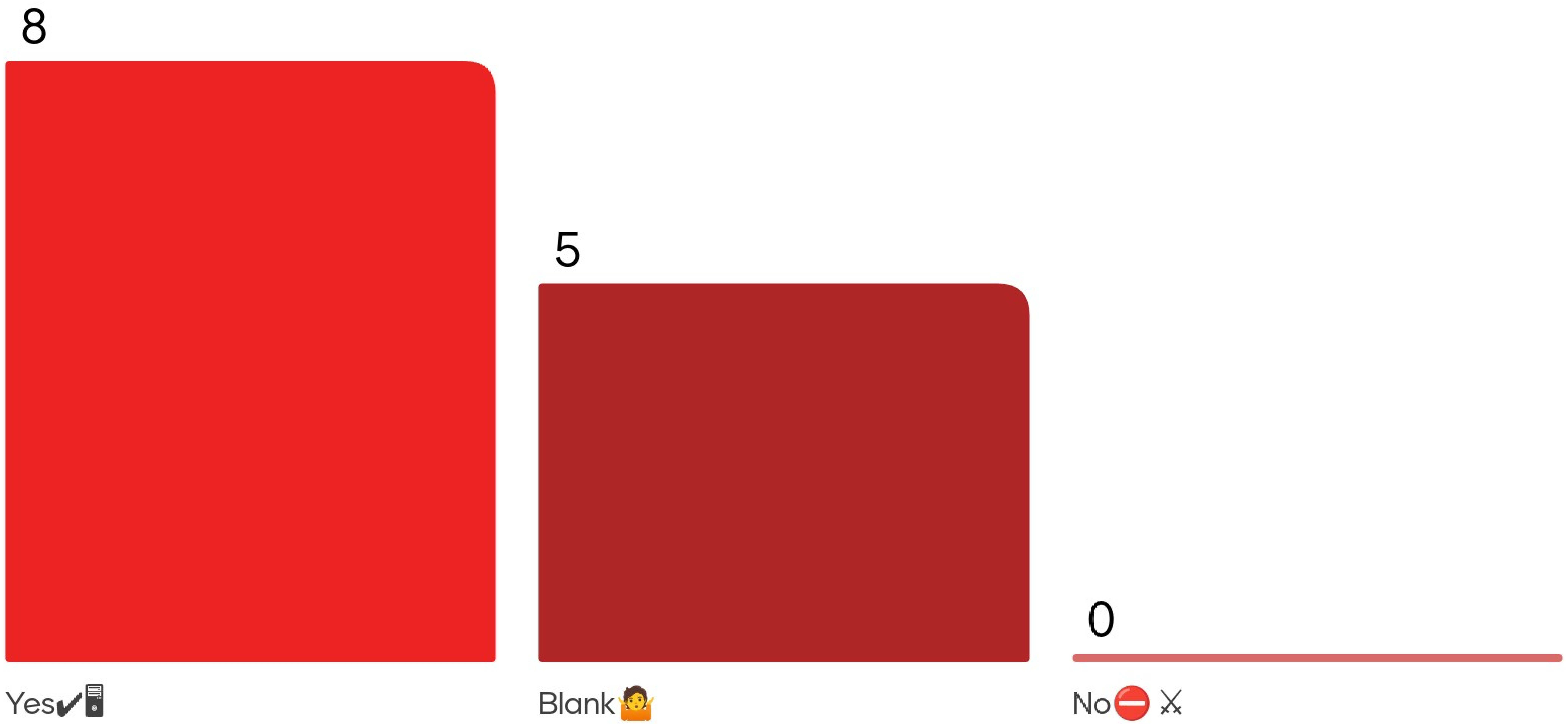
Agenda:
- Introduce assignment D
- Live-program assignment B-1
- Assignment aid

What language do you prefer for the seminars?



1
13

Should we live-program B-1 today?



2 👍 13 👤

IN4120: Science fair group deadline

- Remember the deadlines:
- Group self-assignment: 2024-10-21
- Topic selection: 2024-11-04
- Contact prof. Øhrn about this

Assignment B: Solution sketch available

- → Was pushed early friday 😎👍
- → Is on Github
- → No future assignment depends on B

Assignment C

- Was due fredag
- More doable than B? ☀️
- Correction is underway

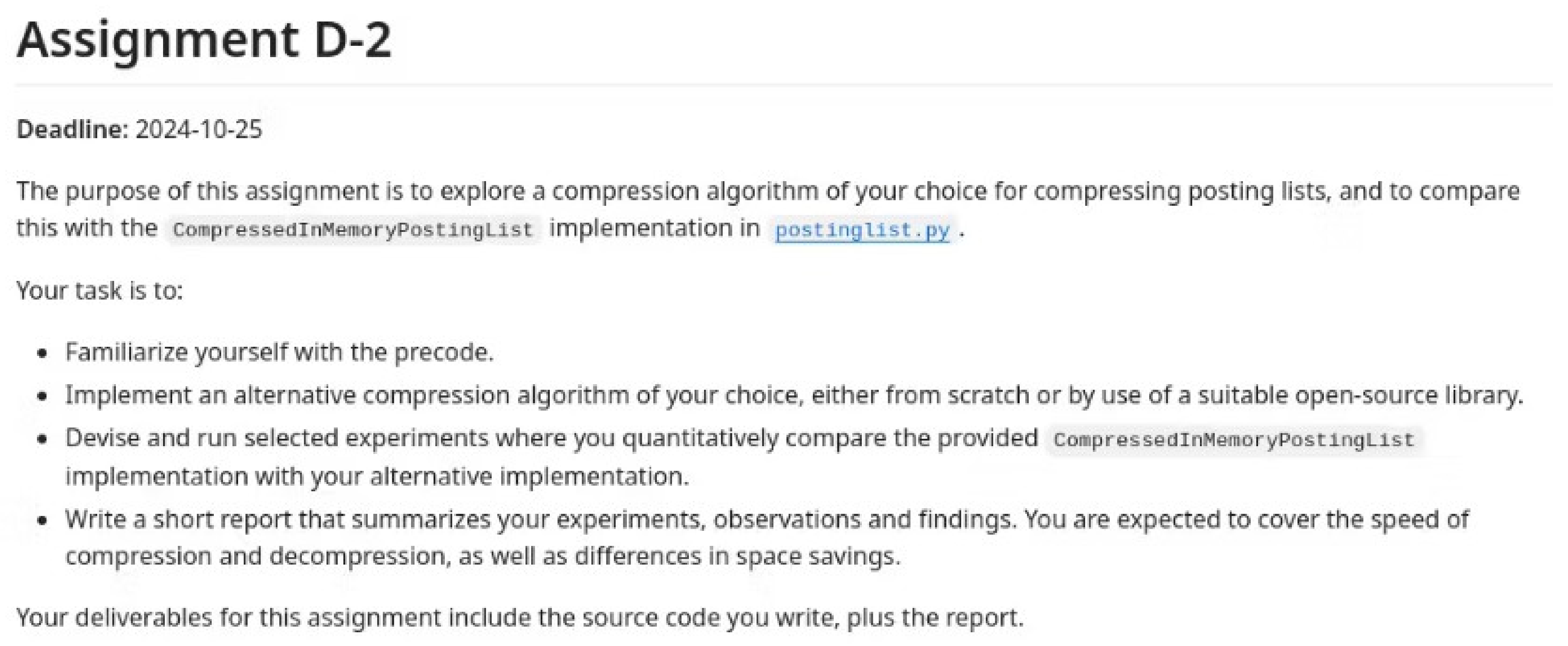
Winter is coming 🙀⛰️

Assignment 4/5: D

Due: 2024-10-25

Assignment D-2

- → New assignment type!
- → Topic: Compression
- → Write a text (and code)
- → At least 600 words



Assignment D-2

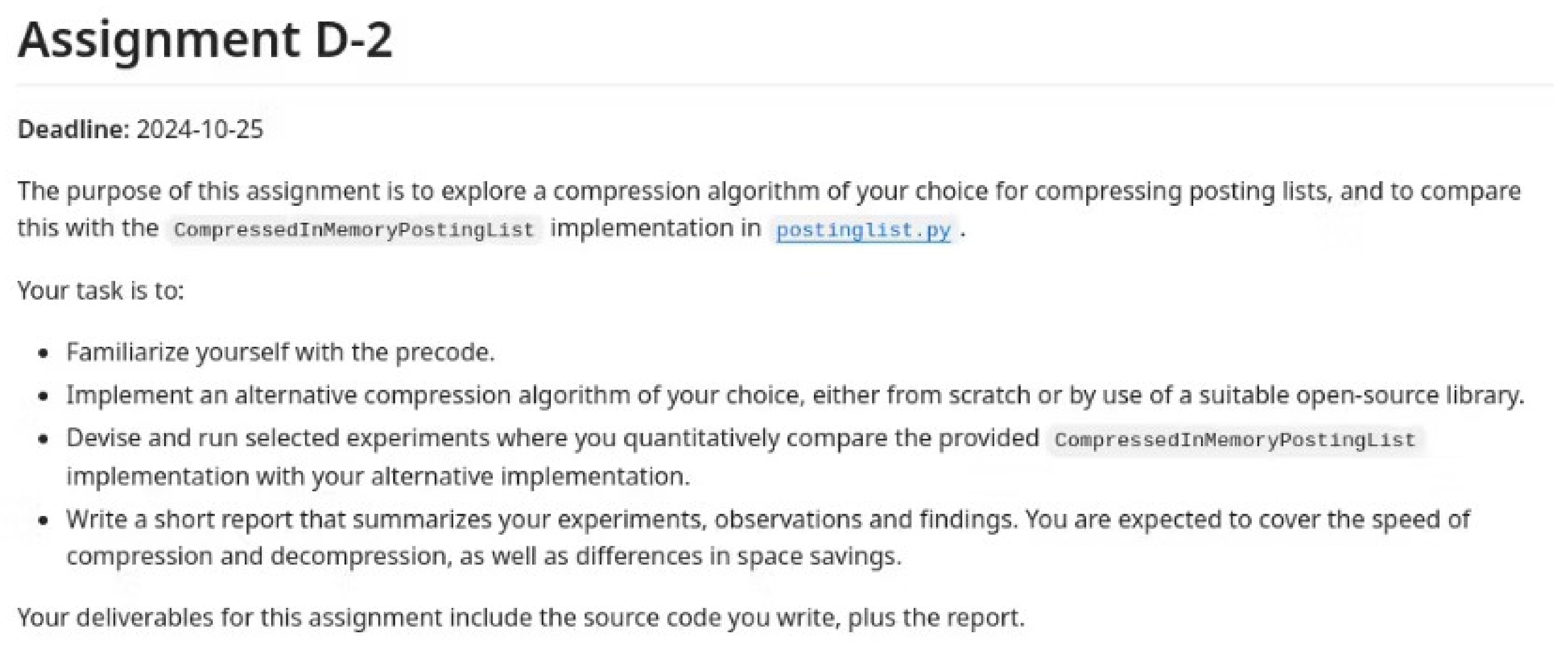
Deadline: 2024-10-25

The purpose of this assignment is to explore a compression algorithm of your choice for compressing posting lists, and to compare this with the `CompressedInMemoryPostingList` implementation in `postinglist.py`.

Your task is to:

- Familiarize yourself with the precode.
- Implement an alternative compression algorithm of your choice, either from scratch or by use of a suitable open-source library.
- Devise and run selected experiments where you quantitatively compare the provided `CompressedInMemoryPostingList` implementation with your alternative implementation.
- Write a short report that summarizes your experiments, observations and findings. You are expected to cover the speed of compression and decompression, as well as differences in space savings.

Your deliverables for this assignment include the source code you write, plus the report.



Assignment D-2

Deadline: 2024-10-25

The purpose of this assignment is to explore a compression algorithm of your choice for compressing posting lists, and to compare this with the `CompressedInMemoryPostingList` implementation in `postinglist.py`.

Your task is to:

- Familiarize yourself with the precode.
- Implement an alternative compression algorithm of your choice, either from scratch or by use of a suitable open-source library.
- Devise and run selected experiments where you quantitatively compare the provided `CompressedInMemoryPostingList` implementation with your alternative implementation.
- Write a short report that summarizes your experiments, observations and findings. You are expected to cover the speed of compression and decompression, as well as differences in space savings.

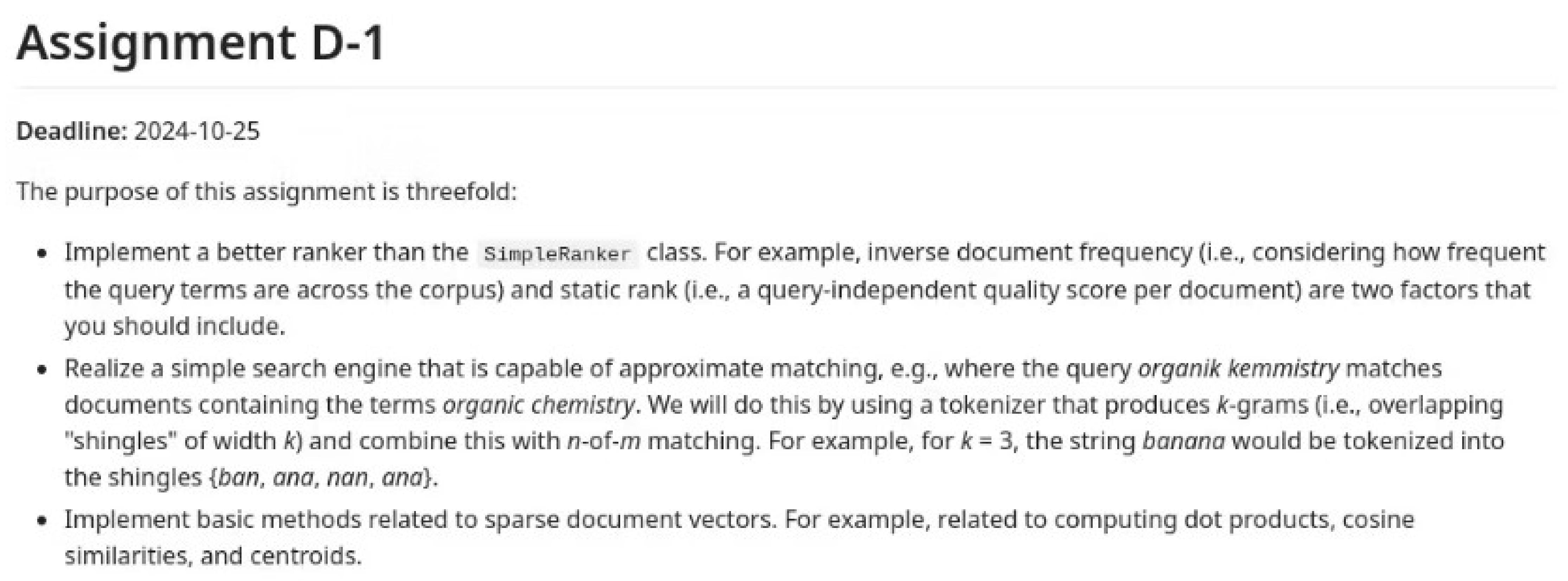
Your deliverables for this assignment include the source code you write, plus the report.

Assignment D-2: Tips

→ Get an overview early

→ Present the work professionally

→ Pull InvertedIndex if your A lacks compression



Assignment D-1

Deadline: 2024-10-25

The purpose of this assignment is threefold:

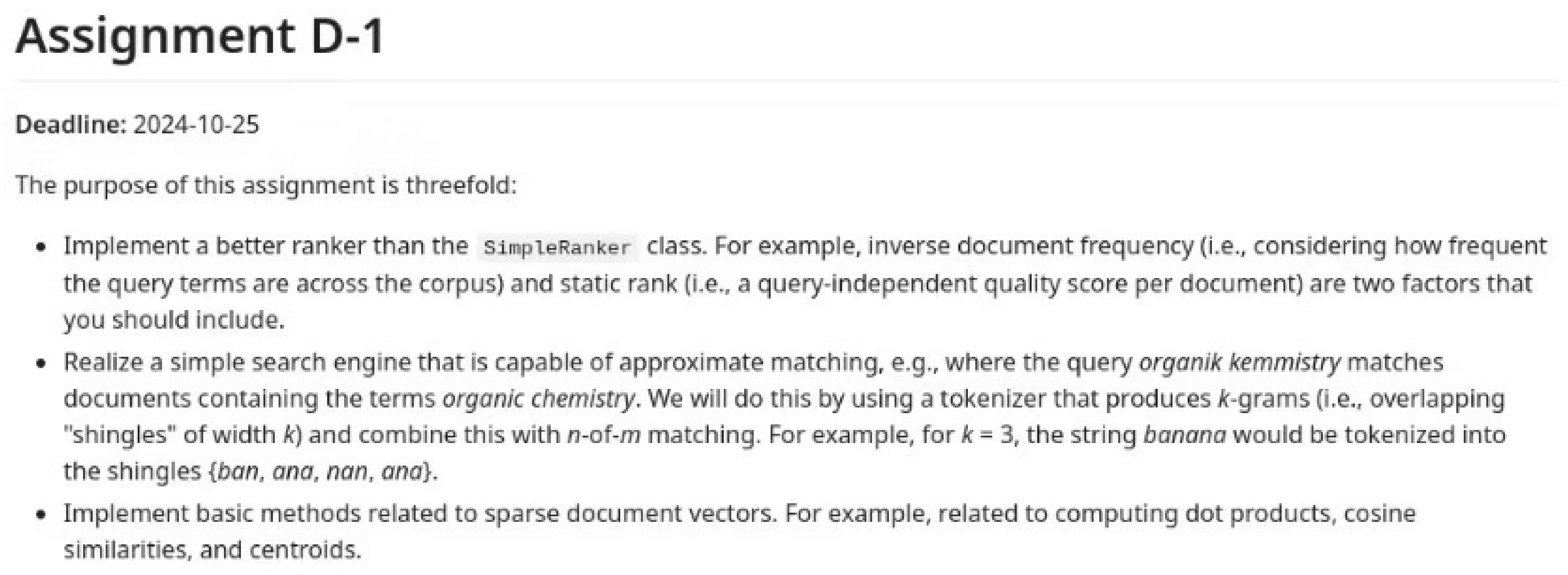
- Implement a better ranker than the `SimpleRanker` class. For example, inverse document frequency (i.e., considering how frequent the query terms are across the corpus) and static rank (i.e., a query-independent quality score per document) are two factors that you should include.
- Realize a simple search engine that is capable of approximate matching, e.g., where the query *organik kemmistry* matches documents containing the terms *organic chemistry*. We will do this by using a tokenizer that produces *k*-grams (i.e., overlapping "shingles" of width *k*) and combine this with *n*-of-*m* matching. For example, for *k* = 3, the string *banana* would be tokenized into the shingles {*ban*, *ana*, *nan*, *ana*}.
- Implement basic methods related to sparse document vectors. For example, related to computing dot products, cosine similarities, and centroids.

Assignment D-1

- → Ranking, TF-IDF
- → Shingles
- → Vectors

Assignment D-1: Hints

- Vector is new -> report bugs
- Don't overcomplicate shingles
- Vector should just be math™



Assignment D-1

Deadline: 2024-10-25

The purpose of this assignment is threefold:

- Implement a better ranker than the `SimpleRanker` class. For example, inverse document frequency (i.e., considering how frequent the query terms are across the corpus) and static rank (i.e., a query-independent quality score per document) are two factors that you should include.
- Realize a simple search engine that is capable of approximate matching, e.g., where the query *organik kemmistry* matches documents containing the terms *organic chemistry*. We will do this by using a tokenizer that produces *k*-grams (i.e., overlapping "shingles" of width *k*) and combine this with *n*-of-*m* matching. For example, for *k* = 3, the string *banana* would be tokenized into the shingles {*ban*, *ana*, *nan*, *ana*}.
- Implement basic methods related to sparse document vectors. For example, related to computing dot products, cosine similarities, and centroids.

Demystify TF-IDF in Indexing and Ranking

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Recall: TF-IDF

Next seminar: classification

- Naïve bayes
- For assignment E
- Base NLP curriculum

Live program B-1?

What could go wrong...

1

Break until 15:15 😄

Assignment workshop. Write something here and we'll discuss it towards the end.