IN[34]120 - Søketeknologi 🖥️📚🔍

2024-10-29 🎃🦅

Tema: Naïve Bayes

- Introdusere oblig E
- Naïve Bayes
- Live-progge C-1?
- Oblighjelp

Assignment D

- Hadde frist fredag ☀️
- Samme nivå som C?
- Rettes fortløpende

Assignment D-1: Gjennomgående feil 👮

- → Ikke kall dunder-metoder direkte
- → Ikke appliser static score flere ganger
- → Husk å bruke vektene!

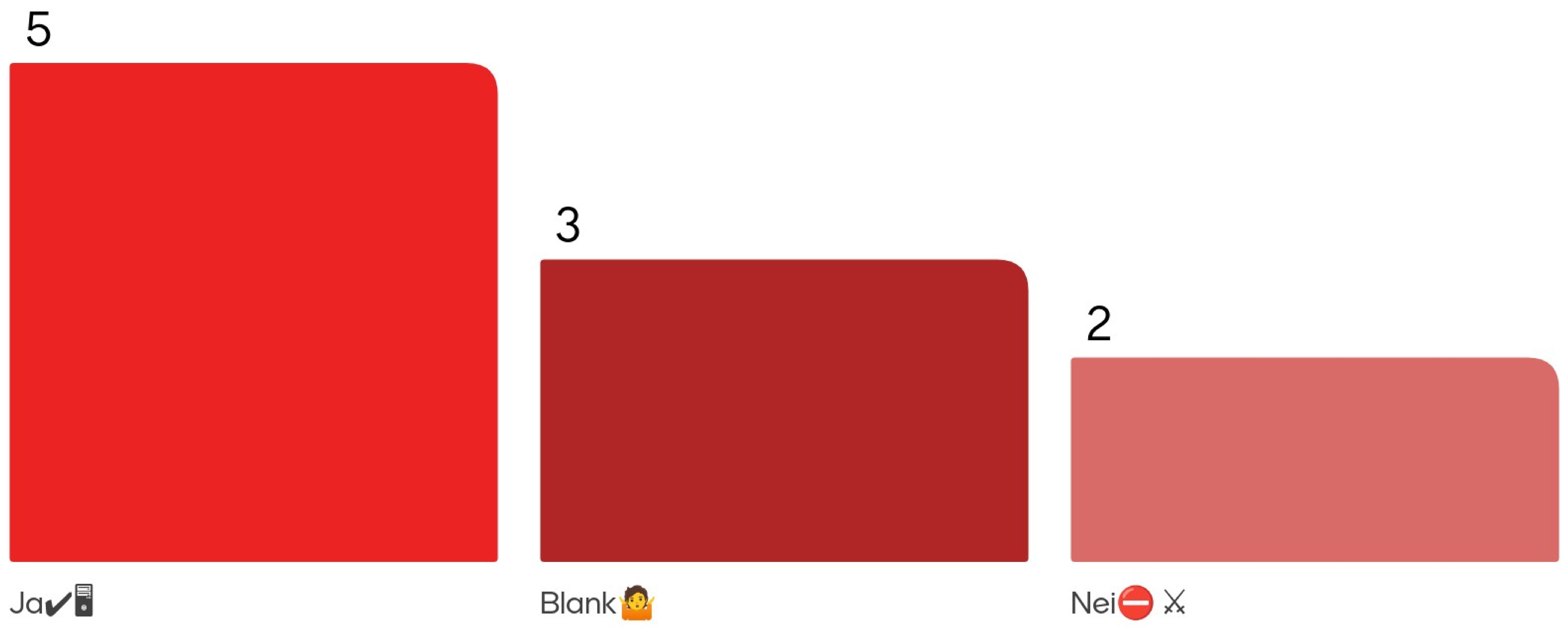
IN4120: Science fair deadlines

- ✔️Group self-assignment: 2024-10-21
- ☞Topic selection: 2024-11-04
- Contact prof. Øhrn about this 🦅

Assignment C: Løsningsforlsag ute

- Kom på dagen på lørdag 😎👍
- Ligger på Github (kan pulles)
- D-1 avhenger av C-1.

Live-progge C-1 etterpå?



3 👍 10 👤

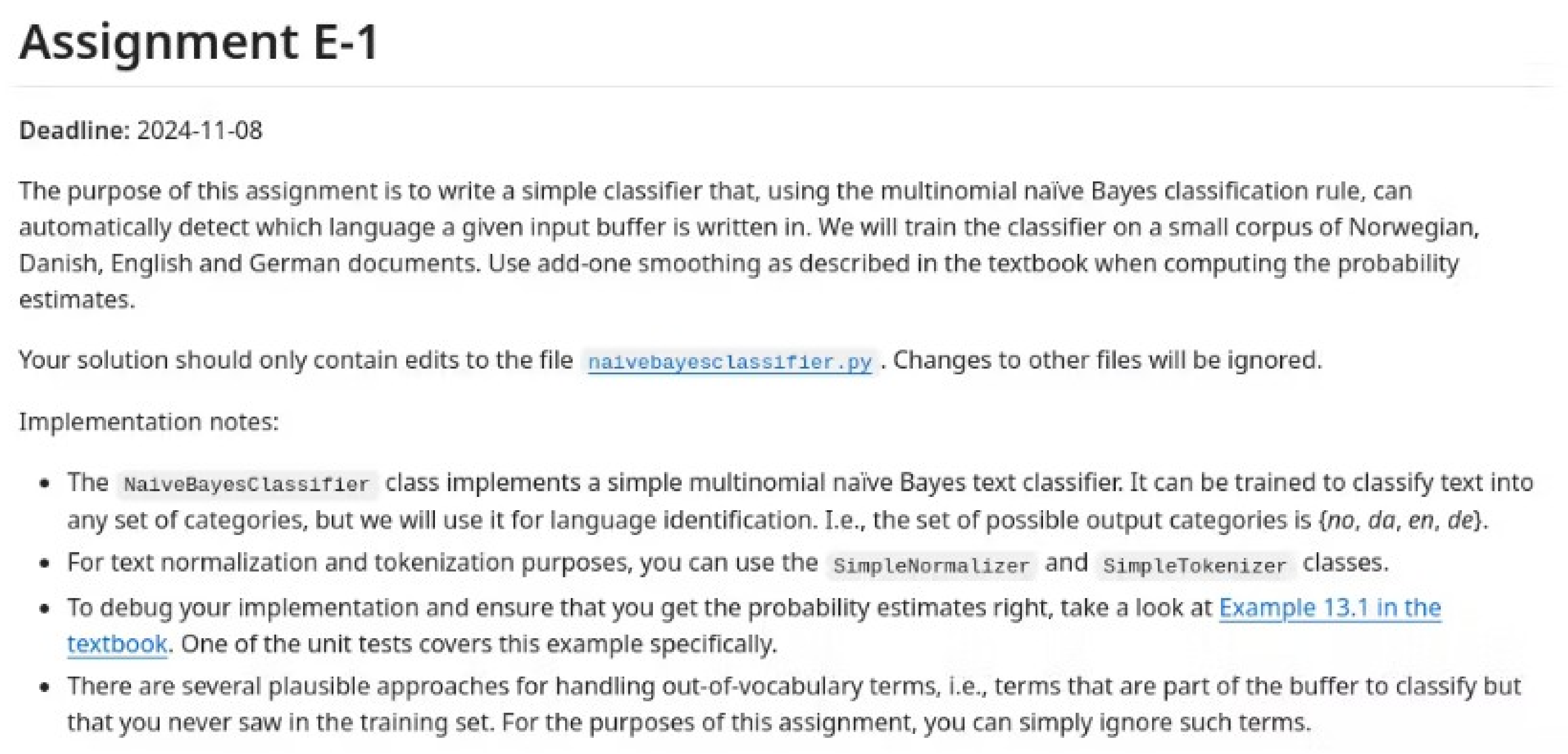
Oblig E

E-1 og E-2

Frist 08.11

Oblig E-1

- Implementere naïve bayes
- *Gitt en tekst, si hvilket språk*
- Tradisjonell: Har tester



Assignment E-1

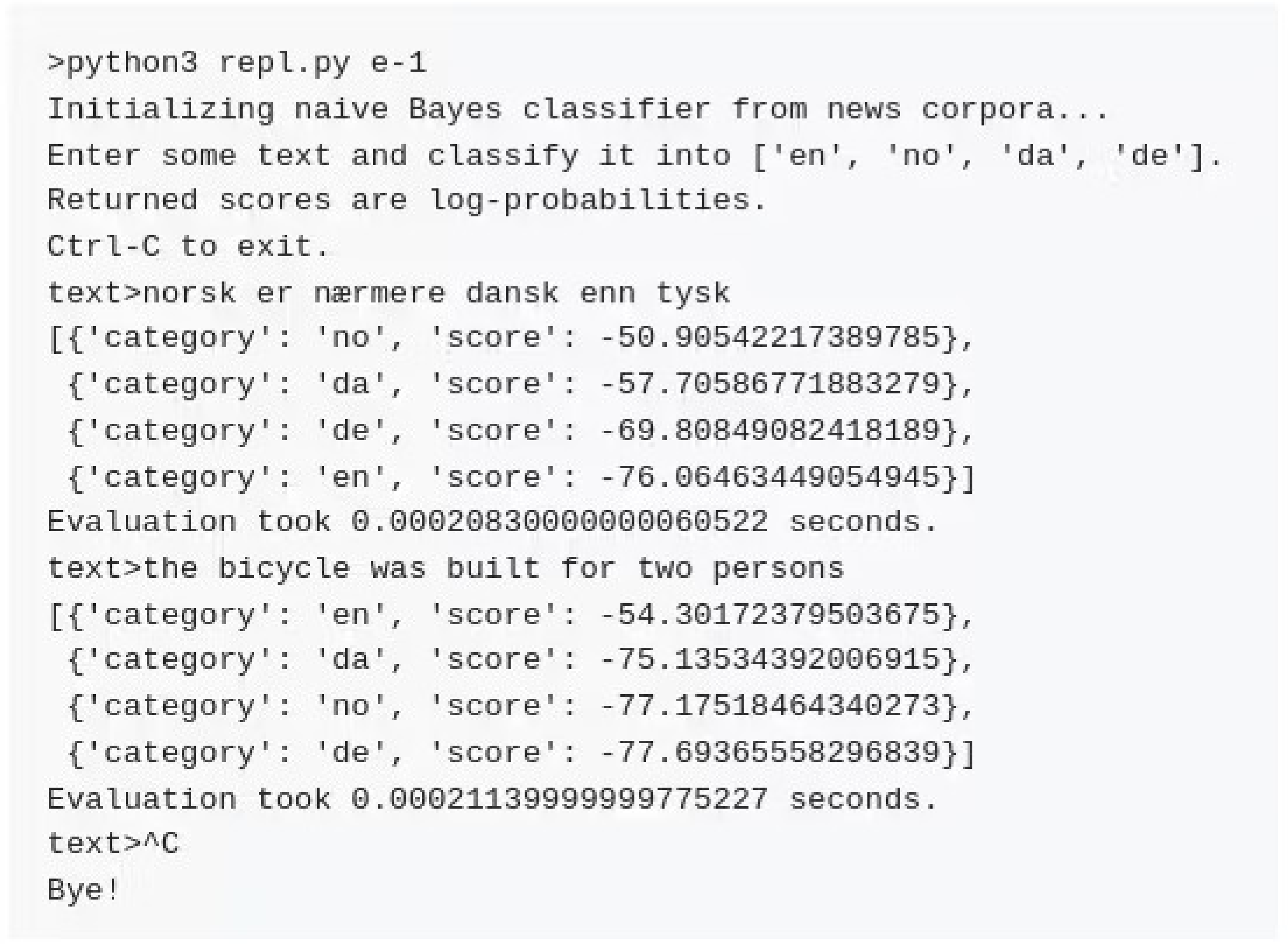
Deadline: 2024-11-08

The purpose of this assignment is to write a simple classifier that, using the multinomial naïve Bayes classification rule, can automatically detect which language a given input buffer is written in. We will train the classifier on a small corpus of Norwegian, Danish, English and German documents. Use add-one smoothing as described in the textbook when computing the probability estimates.

Your solution should only contain edits to the file `naivebayesclassifier.py`. Changes to other files will be ignored.

Implementation notes:

- The `NaiveBayesClassifier` class implements a simple multinomial naïve Bayes text classifier. It can be trained to classify text into any set of categories, but we will use it for language identification. I.e., the set of possible output categories is {*no*, *da*, *en*, *de*}.
- For text normalization and tokenization purposes, you can use the `SimpleNormalizer` and `SimpleTokenizer` classes.
- To debug your implementation and ensure that you get the probability estimates right, take a look at Example 13.1 in the textbook. One of the unit tests covers this example specifically.
- There are several plausible approaches for handling out-of-vocabulary terms, i.e., terms that are part of the buffer to classify but that you never saw in the training set. For the purposes of this assignment, you can simply ignore such terms.



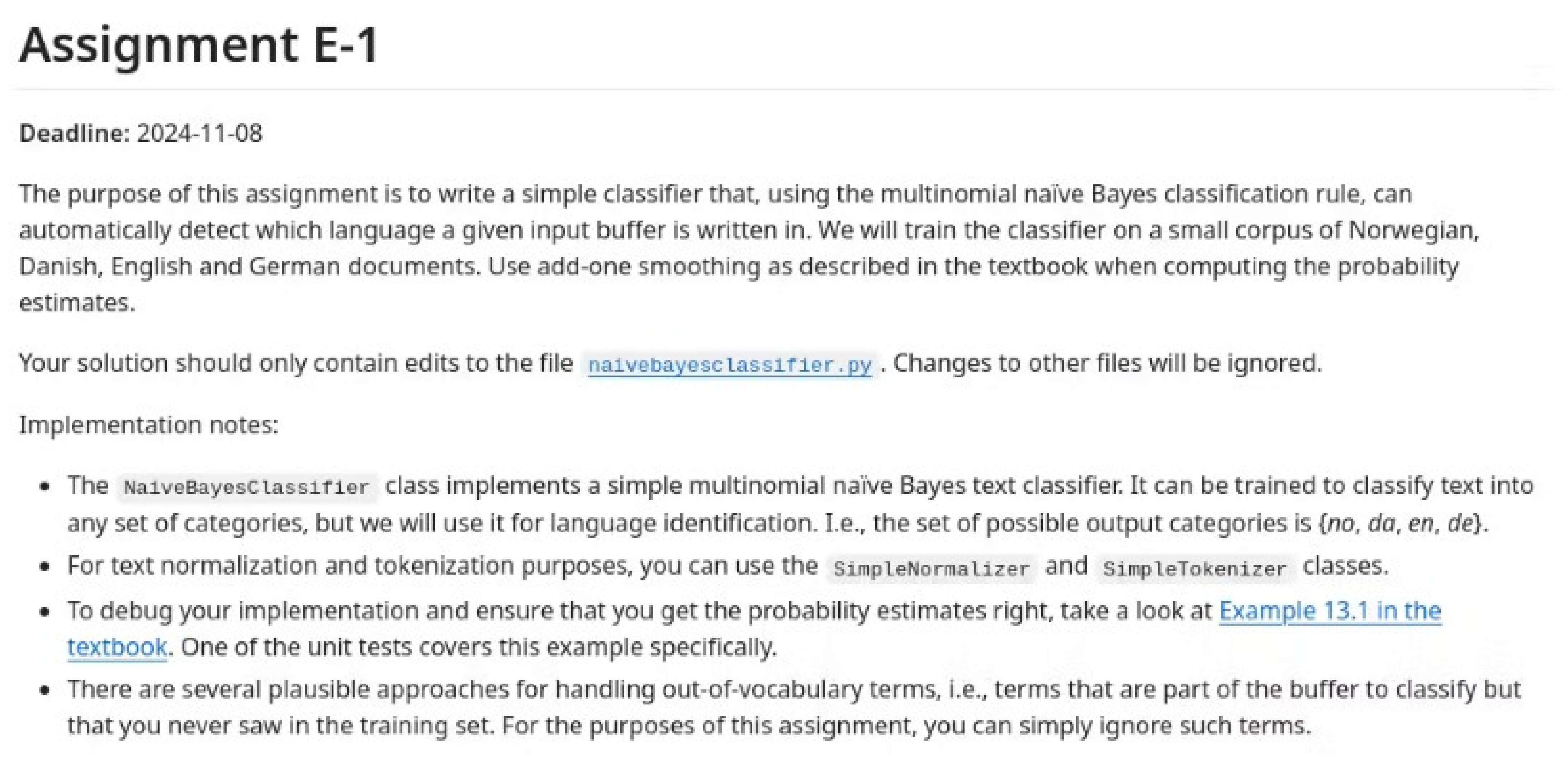
```
>python3 repl.py e-1
Initializing naive Bayes classifier from news corpora...
Enter some text and classify it into ['en', 'no', 'da', 'de'].
Returned scores are log-probabilities.
Ctrl-C to exit.
text>norsk er nærmere dansk enn tysk
[{'category': 'no', 'score': -50.90542217389785},
 {'category': 'da', 'score': -57.70586771883279},
 {'category': 'de', 'score': -69.80849082418189},
 {'category': 'en', 'score': -76.06463449054945}]
Evaluation took 0.00020830000000060522 seconds.
text>the bicycle was built for two persons
[{'category': 'en', 'score': -54.30172379503675},
 {'category': 'da', 'score': -75.13534392006915},
 {'category': 'no', 'score': -77.17518464340273},
 {'category': 'de', 'score': -77.69365558296839}]
Evaluation took 0.00021139999999775227 seconds.
text>^C
Bye!
```

Oblig E-1: Notabene

- Må "lære" språkene vi bruker
- språk $\in \{NO, DA, DE, EN\}$
- Probalistiske resultater

Oblig E-1: Hint

- Sjekk verdiene opp mot China-eksempelet(!!)
- Print verdiene underveis
- Kap #13, side #261 ish
- Fritt å endre struktur på metodene*



Assignment E-1

Deadline: 2024-11-08

The purpose of this assignment is to write a simple classifier that, using the multinomial naïve Bayes classification rule, can automatically detect which language a given input buffer is written in. We will train the classifier on a small corpus of Norwegian, Danish, English and German documents. Use add-one smoothing as described in the textbook when computing the probability estimates.

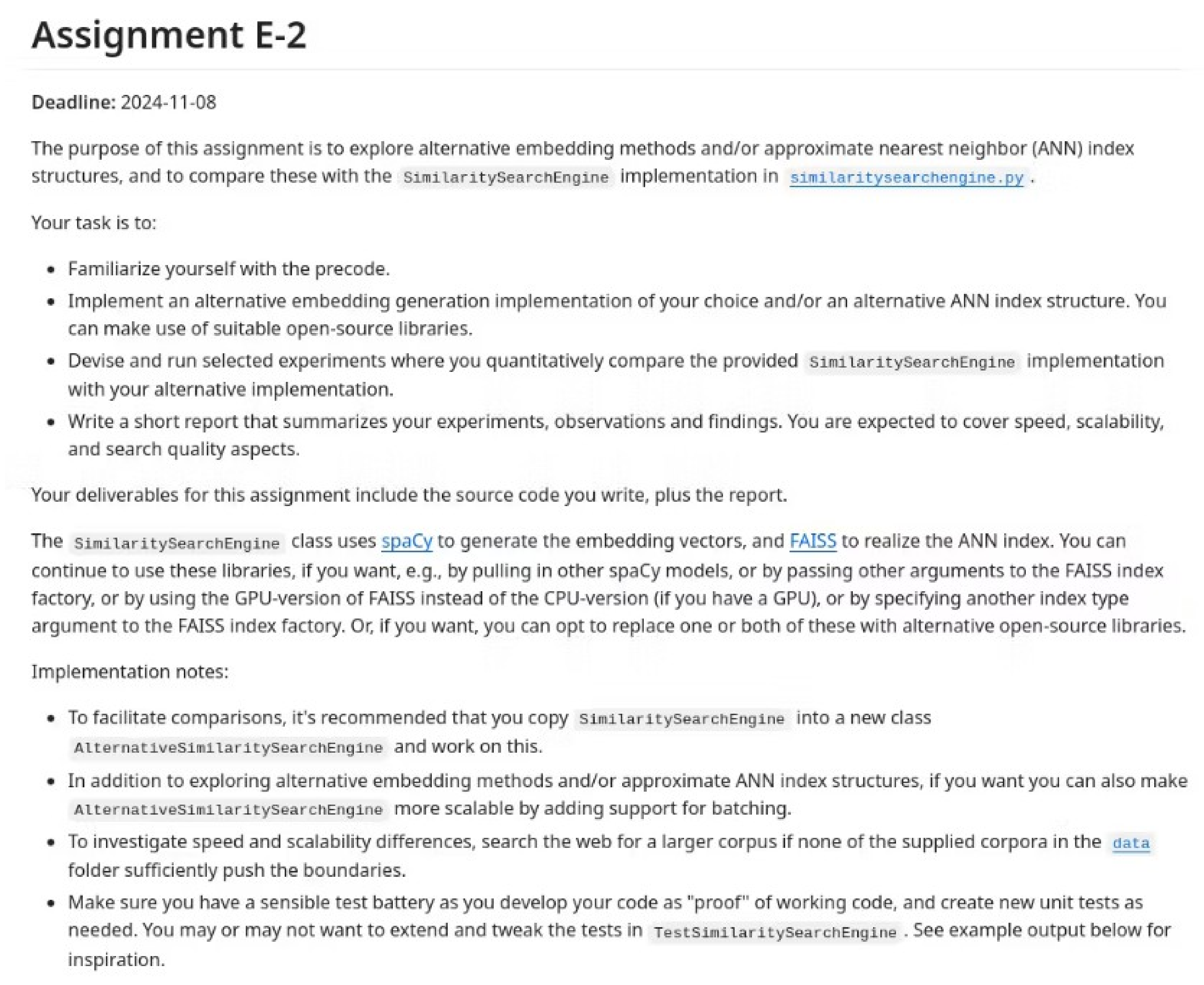
Your solution should only contain edits to the file `naivebayesclassifier.py`. Changes to other files will be ignored.

Implementation notes:

- The `NaiveBayesClassifier` class implements a simple multinomial naïve Bayes text classifier. It can be trained to classify text into any set of categories, but we will use it for language identification. I.e., the set of possible output categories is {*no*, *da*, *en*, *de*}.
- For text normalization and tokenization purposes, you can use the `SimpleNormalizer` and `SimpleTokenizer` classes.
- To debug your implementation and ensure that you get the probability estimates right, take a look at Example 13.1 in the textbook. One of the unit tests covers this example specifically.
- There are several plausible approaches for handling out-of-vocabulary terms, i.e., terms that are part of the buffer to classify but that you never saw in the training set. For the purposes of this assignment, you can simply ignore such terms.

Oblig E-2

- Similiarity search
- Rapport, à la D-2
- Embedding methods
- Approximate nearest neighbors (ANN)



Assignment E-2

Deadline: 2024-11-08

The purpose of this assignment is to explore alternative embedding methods and/or approximate nearest neighbor (ANN) index structures, and to compare these with the `SimilaritySearchEngine` implementation in `similaritysearchengine.py`.

Your task is to:

- Familiarize yourself with the precode.
- Implement an alternative embedding generation implementation of your choice and/or an alternative ANN index structure. You can make use of suitable open-source libraries.
- Devise and run selected experiments where you quantitatively compare the provided `SimilaritySearchEngine` implementation with your alternative implementation.
- Write a short report that summarizes your experiments, observations and findings. You are expected to cover speed, scalability, and search quality aspects.

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

The `SimilaritySearchEngine` class uses spaCy to generate the embedding vectors, and FAISS to realize the ANN index. You can continue to use these libraries, if you want, e.g., by pulling in other spaCy models, or by passing other arguments to the FAISS index factory, or by using the GPU-version of FAISS instead of the CPU-version (if you have a GPU), or by specifying another index type argument to the FAISS index factory. Or, if you want, you can opt to replace one or both of these with alternative open-source libraries.

Implementation notes:

- To facilitate comparisons, it's recommended that you copy `SimilaritySearchEngine` into a new class `AlternativeSimilaritySearchEngine` and work on this.
- In addition to exploring alternative embedding methods and/or approximate ANN index structures, if you want you can also make `AlternativeSimilaritySearchEngine` more scalable by adding support for batching.
- To investigate speed and scalability differences, search the web for a larger corpus if none of the supplied corpora in the `data` folder sufficiently push the boundaries.
- Make sure you have a sensible test battery as you develop your code as "proof" of working code, and create new unit tests as needed. You may or may not want to extend and tweak the tests in `TestSimilaritySearchEngine`. See example output below for inspiration.

Naïve Bayes

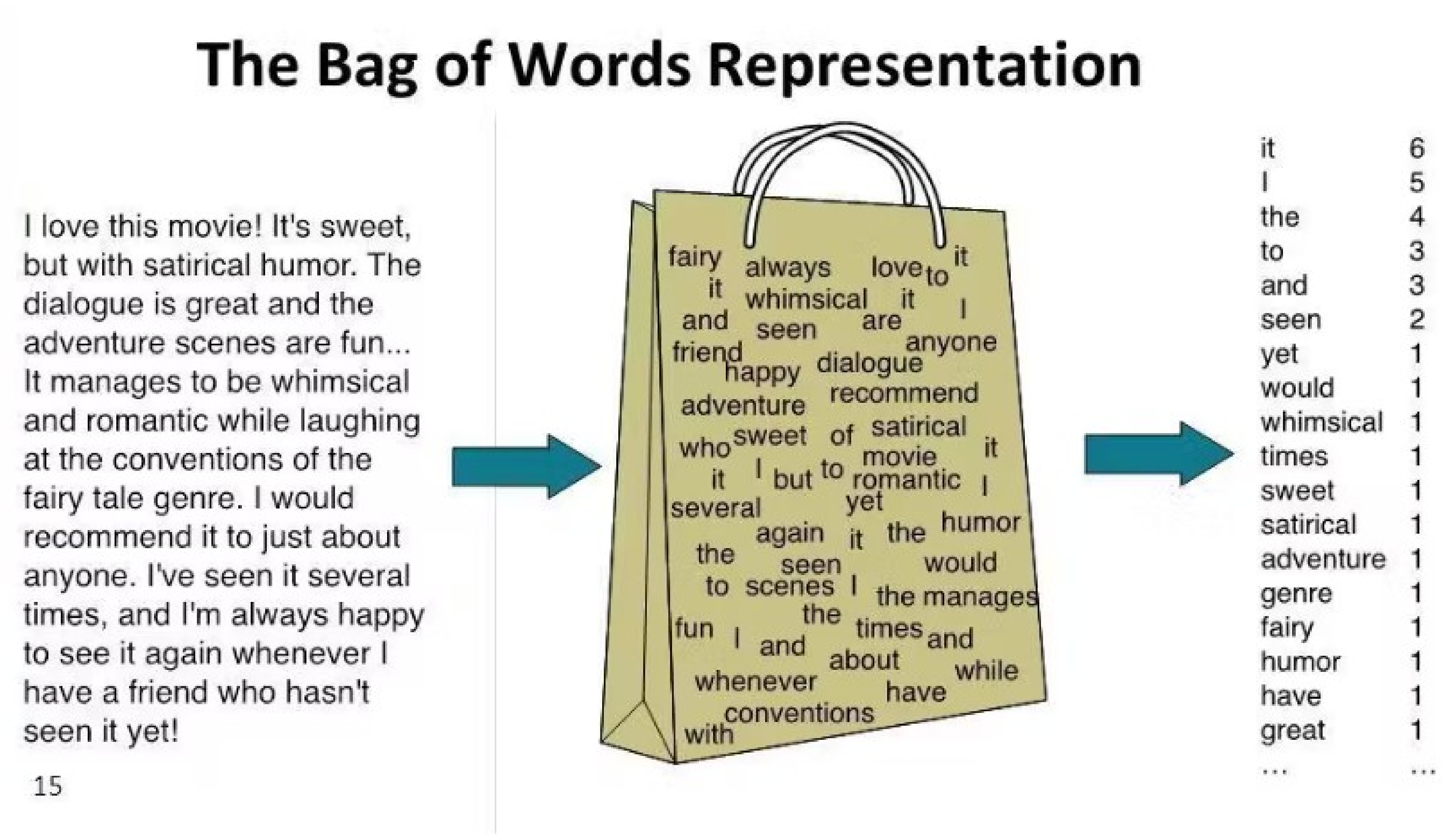
En innføring i språktek for prosa-folk.

Med fokus på oblig E-1.

Fra forrige uke

Den distribusjonelle hypotesen

- → Rekkefølgen på ord er irrelevant
- → Ergo naïv.
- → **Aka: Bag of words (B-O-W)**



The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

15

Bayes' theorem (1/2)

- *Betinget sannsynlighet*
- "Sannsynligheten for språk x gitt ord y"

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Statistikk-notasjon

- P(x): Sannsynligheten for x
- P(y): Sannsynligheten for y
- P(x | y): Sannsynligheten for x gitt y
- Tenk: P() = probability

Bayes' theorem (2/2)

- *Betinget sannsynlighet*
- "Sannsynligheten for språk x gitt ord y"
- Merk: Vi vet P(B | A), men ikke P(A | B)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes: Hva vet vi?

- Mål: "Sannsynligheten for språk x gitt ord y", P(A | B)
- A: Språk
- B: Term
- Vi vet sannsynligheten for et ord gitt et språk, P(B | A)
- Vet vi P(A) og P(B)?

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Notasjon i maskinlæring

- ^: Caret. Indikerer prediksjon
- c: En klasse med navn c
- ĉ: En *predikert* klasse c

Naïve Bayes: Forklart med ord

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c).$$

1. Ta inn en tekst
2. Regn ut sannsynligheten for at teksten er hver klasse vi kjenner
3. Returner sortert resultat

Ting vi ikke rakk:

- Smoothing
- Denominators
- Bruk av logaritmer

Neste gang:

- TDB (kom gjerne med innspill)
- *Vurderer gruppearbeid*
- Bloom filters?
- Live-progge D-1?
- Oblighjelp for E.

Live progge C-1?

Hva kan nå gå galt…

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

Break until 15:15 😄