IN2110 våren 2020 – obligatorisk innlevering 2b

Leveres innen ***tirsdag 12. mai kl. 23.59*** *i Devilry.*

Det er en god idé å lese gjennom hele oppgavesettet før du setter i gang. Har du spørsmål så spør på Piazza (foretrukket) eller send epost til

`in2110-hjelp@ifi.uio.no`.

På grunn av den relativt korte tidsfristen (og den generelle COVID-19 situasjonen) forventer vi ikke at dere skal kunne løse absolutt alle spørsmålene i denne oppgaven. Vi har derfor vært rause med poengfordelingen (i parentes) i denne oppgaven, slik at dere kan velge å hoppe over noen få spørsmål.

Oppsett

For IN2110 har vi laget et utviklingsmiljø for Python som inneholder programvare og data for obligene. For mer informasjon om bruk av miljøet via SSH, eller for installasjon på egen maskin, se;
https://github.uio.no/IN2110/in2110-shared

Merk at miljøet er nylig blitt oppdatert for å kunne kjøre maskinoversettelsesystemet OpenNMT. Om du allerede har installert miljøet på egen maskin må du oppdatere det ved å kjøre `update.sh`:

```
$ ./update.sh
```

I oppgavene under skal dere ta i bruk data som er tilgjengelig under mappen `obliger/2b` i kurs-repoet for gruppetimer og obliger:
https://github.uio.no/IN2110/in2110-lab/

Innleveringsformat

Innleveringen skal bestå av en Python-fil med kode samt en liten rapport.

Del 1: Maskinoversettelse

Vi skal bruke en (nevral) maskinoversettelsemodell til å oversette filmtekstinger fra Ringenes Herre (og Hobbiten) fra tysk til engelsk.

Data

Filmtekstingene ligger i filene lotr.de og lotr.en for henholdsvis de tyske og engelskspråklige filmtekstingene. Disse to filene utgjør et såkalt *parallellkorpus*, altså en tekstsamling hvor hver setning (i språk A) er koblet til en tilsvarende setning i språk B. De 2 filene har samme antall linjer, slik at den tyske setningen på linjen i av lotr.de har en engelsk oversettelse på samme linje av lotr.en. Filmtekstingene er ekstrahert fra korpuset OpenSubtitles-2018.[1]
Her er f.eks. de 10 første linjene i lotr.de og lotr.en:

	Tysk (lotr.de)	Engelsk (lotr.en)
1	Die Welt ist im Wandel .	The world is changed .
2	Ich spüre es im Wasser .	I feel it in the water .
3	Ich spüre es in der Erde .	I feel it in the earth .
4	Ich rieche es in der Luft .	I smell it in the air .
5	Vieles , was einst war , ist verloren , da niemand mehr lebt , der sich erinnert .	Much that once was is lost . For none now live who remember it .
6	Es begann mit dem Schmieden der Großen Ringe .	It began with the forging of the Great Rings .
7	3 wurden den Elben gegeben , den unsterblichen , weisesten und reinsten aller Wesen .	Three were given to the Elves : Immortal , wisest and fairest of all beings .
8	7 den Zwergenherrschern , großen Bergleuten und Handwerkern in ihren Hallen aus Stein .	Seven to the Dwarf-lords : Great miners and craftsmen of the mountain halls .
9	Und 9 ... 9 Ringe wurden den Menschen geschenkt , die vor allem anderen nach Macht streben .	And nine nine rings were gifted to the race of Men who , above all else , desire power .
10	Denn diese Ringe bargen die Kraft und den Willen , jedes Volk zu leiten .	For within these rings was bound the strength and will to govern each race .

Merk at teksten allerede er tokenisert. Noen ganger kan det være store sprik mellom innholdet i filmtekstingene. Det er ikke nødvendigvis en oversettelsefeil – det er bare at filmtekstere kan velge å transkribere hva som skjer i filmen på litt ulike måter.

Maskinoversettelsesystem

Vi skal ta i bruk OpenNMT, en programvare for å trene og kjøre maskinoversettelsemodeller basert på dype nevrale nettverk. Vi skal imidlertid ikke trene en ny modell, da trening av slike modeller krever ganske store regneressurser, blant annet GPUs. Men heldigsvis finnes det allerede pre-trente modeller som kan brukes. Last ned den tysk-engelsk modellen tilgjengelig her:

```
$ wget -O de2en.pt https://s3.amazonaws.com/opennmt-models/iwslt-brnn2.s131_acc_62.71_ppl_7.74_e20.pt
```

Modellen ovenfor er en nevral oversettelsemodell (sequence-to-sequence med 2 LSTM-nivåer) som ble trent på transkripsjoner av TED og TEDx talks. Tre-

[1] For å vite mer om korpuset: Lison, P., Tiedemann, J. & Kouylekov, M. (2018) Statistical rescoring of sentence alignments in large, noisy parallel corpora. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC-2018)*.

ningsdata som ble brukt er ikke særlig stor, og oversettelsemodellen klarer derfor ikke å oversette alt (eller produserer feil oversettelser).

Komme i gang

Viktig: oversettelsemodeller krever at alle ordene er skrevet med små bokstaver. Derfor må vi forvandle alle linjene til lowercase:

```
$ ipython
In [1]: with open("lotr.de") as fd_in, open("lotr.lc.de", "w") as fd_out:
   ...:     for line in fd_in:
   ...:         fd_out.write(line.lower())
```

La oss nå lage en liten testfil med de første 100 linjene av filmtekstingene:

```
    $ head -n 100 lotr.lc.de > lotr.small.lc.de
```

Og kjøre OpenNMT på filen:

```
$ onmt_translate --model de2en.pt --src lotr.small.lc.de --output lotr.small.out.en
```

Resultatet bør se slikt ut (7 første linjer):

```
1 the world is changing .
2 i feel it in the water .
3 i feel it in the world .
4 i smell it in the air .
5 a lot of what once was , is lost because nobody lives more , who remembers .
6 it started with the back of the big rings .
7 three were given the <unk> , the immortal , <unk> and <unk> of all creatures .
```

<unk>-ordene

Merk de spesielle <unk>-ordene i den sisten linjen, som viser at modellen ikke har klart å oversette noen tyske ord. Setningen på tysk side er nemlig:

```
7 3 wurden den Elben gegeben , den unsterblichen , weisesten und reinsten aller
↪  Wesen .
```

Siden modellen ble trent på et datasett av TED og TEDx talks er det ikke overraskende at den ikke vet hvordan et ord som "Elben" (alver) skal oversettes! I et slikt tilfelle velger modellen å generere et spesielt <unk>-token i stedet.

En liten digresjon: Oversettelsemodellen som brukes her baserer seg på en sequence-to-sequence arkitektur som genererer oversettelsen ord etter ord. Når man jobber med et språk som tysk (som har en relativ rik morfologi, med mange samensatte ord) er det ofte bedre å benytte seg av modeller som klarer å håndtere lingvistiske enheter som ligger under ordnivå (som morfemer eller andre

orddeler). Det finnes flere maskinoversettelsemodeller som gjør akkurat det[2], men de er litt mer kompliserte, så dere skal slippe å forholde dere til det.

OpenNMT tilbyr en måte å erstatte ukjente ord med ordene i kildespråket (altså tysk) som har høyest "attention weight" ved generering av hvert <unk>-ord. Mekanismen aktiveres ved å legge til `-replace_unk`:

```
$ onmt_translate --model de2en.pt --src lotr.small.lc.de  --output
↪  lotr.small.out.en --replace_unk
```

Resultatet ser nå slikt ut på linjen 7:

```
7  three were given the elben , the immortal , weisesten and reinsten of all
↪  creatures .
```

Her ser man at <unk>-ordene har blitt erstattet med tyske ord fra kildesetningene. Noen ganger er denne erstatningsmekanismen ganske nyttig, for eksempel for å oversette egennavn (hvis man ikke vet hvordan e.g. "Gandalf" skal oversettes er det beste å gjenbruke samme ordet i målsetningen). Men her får vi en litt komisk oversettelse.

a) Utvikling av en frasetabell (2 poeng)

Vi skal nå sørge for at systemet blir bedre på å oversette ukjente ord som "Elben". OpenNMT tilbyr en postprosesseringsmekanisme for å håndtere ukjente ord med en såkalt *frasetabell*, altså en slags ordbok. Hvis et ord ikke kan oversettes direkte av det nevrale nettverket vil systemet søke ordet i tabellen, og, hvis ordet finnes der vil oversettelsen brukes som erstatning. I OpenNMT er mekanismen begrenset til enkle ord på kildesiden (men kan bestå av flere ord på målsiden, altså engelsk i vårt tilfelle).

Rent praktisk gir man frasetabellen ved å legge til `-phrase_table table_file`. Frasetabellen må ha et ordpar per linje, og skille kilde- og målord med `|||`.

I filen `de-en.txt` kan dere finne et generisk tysk-engelsk ordbok (som jeg lastet ned fra nettsiden www.dict.cc) som kan brukes om eksempel:

```
$ onmt_translate --model de2en.pt --src lotr.small.lc.de --output
↪  lotr.small.out_with_dic.en  --replace_unk --phrase_table de-en.txt
```

Oppgaver:

1. Hva er modellens oversettelse for linjene 10-15?
2. Finn et eksempel hvor bruk av ordboken førte til en dårlig oversettelse.
3. Lage en liste med minst 10 ord som systematisk er feil oversatt (som f.eks. "Beutlin" som bør oversettes til "Baggins") og legge disse i frasetabellen `de-en.txt`.

[2]Se f.eks. Sennrich, R., Haddow, B. og Birch, A. (2016), Neural Machine Translation of Rare Words with Subword Units, *Proceedings of ACL 2016.*

b) Evaluering (6 poeng)

Vi er nå klare til å evaluere kvaliteten på oversettelsene vi har generert. Her skal vi bruke en evalueringsmetode som er veldig populær i maskinoversettelse, nemlig BLEU. BLEU er en automatisert evalueringsmetode som sammenligner oversettelse som systemet har produsert med en eller flere fasiter, altså oversettelser skrevet av menneskelige eksperter. I vårt tilfelle er fasiten de engelskspråklige filmtekstingene i `lotr.en`.

BLEU beregnes ved å se på *overlapp* mellom N-grams fra fasiten(e) og oversettelsene fra systemet. Mer presist ekstraherer vi for hver setning alle N-grams (med N fra 1 til 4) fra både systemet og fasiten, og beregner hva som er precision for $i \in 1, 2, 3, 4$:

$$precision_i = \frac{\text{Antall } i\text{-grams som er felles i både system og fasit (for samme setning)}}{\text{Antall } i\text{-grams i setninger fra systemet}} \tag{1}$$

Deretter slår vi sammen precision-tallene:

$$BLEU = brevity\_penalty * \left(\prod_{i=1}^{4} precision_i \right)^{\frac{1}{4}} \tag{2}$$

hvor "brevity penalty" brukes til å "straffe" modeller som produserer for korte oversettelser:

$$brevity\_penalty = min(1, \frac{\text{Antall ord i systemets setninger}}{\text{Antall ord i fasitens setninger}}) \tag{3}$$

Oppgaver:

1. Fyll ut resten av funksjonen `compute_precision(ref_file, output_file, ngram_order)` som beregner *precision*-verdien (som definert over) for en gitt N-gram ordre.
2. Fyll ut resten av funksjonen `compute_brevity_penalty(ref_file, output_file)` som beregner verdien for *brevity penalty* (som definert over).
3. Kjør OpenNMT på hele filen lotr.de (husk å konvertere den til lowercase først) uten frasetabell og beregne BLEU-scoren ved å kalle funksjonen `compute_bleu(ref_file, output_file)`, som allerede er implementert.
4. Kjør OpenNMT på hele filen lotr.de, denne gang med frasetabellen som du har redigert, og beregne BLEU-scoren.

Tips: BLEU-scoren uten frasetabell bør være ca. 0.229, mens scoren med frasetabell bør være ca. 0.239.

Del 2: Interaktive systemer (8 poeng)

Filmtekstinger kan brukes til andre formål enn å trene og teste maskinoversettelsesystemer – filmtekstinger består også i all hovedsak av *samtaler* og kan derfor også brukes til å bygge datadrevne dialogsystemer!

I denne delen av oppgaven skal dere utvikle en liten *retrieval-based chatbot* basert på korpuset i `lotr.en`. Chatboten vår vil derfor "snakke" som filmkarakterer i Ringene Herre. Hovedidéen er å:

- beregne TF-IDF-vektorene av alle setningene i korpuset vårt og lagre disse. La oss kalle disse vektorene $[t_1, t_2, ...t_{|C|}]$, hvor $|C|$ er antall setninger i korpuset.
- når chatbot mottar en ny inputsetning fra brukeren beregner vi TF-IDF vektoren q av denne setningen.
- deretter leter man etter setningen i korpuset som ligner mest på inputsetningen ved å beregne *cosine similarity* mellom TF-IDF vektoren av inputsetningen og hver TF-IDF-vektor fra korpuset C:

$$i^* = \arg\max_{i=1}^{|C|} \frac{q^T t_i}{||q||\ ||t_i||} \tag{4}$$

- Til slutt tar vi setningen som kommer *rett etter* setningen t_{i^*}, altså t_{i^*+1}.

For å ta et eksempel: la oss si at brukeren skriver:

```
Are you Bilbo Baggins ?
```

Ifølge cosine similarity mellom TF-IDF vektorer er setningen i korpuset som ligner mest på inputsetningen:

```
4907 Bilbo Baggins .
```

Da vil systemet ta setningen på linjen 4908 og svare brukeren:

```
4908 I 'm sorry , do I know you ?
```

Oppgaver:

1. Fyll ut metoden `get_tf_idf(self, utterance)` som beregner TF-IDF vektoren for den gitte setningen.
2. Fyll ut metoden `compute_cosine(self, tf_idf1, tf_idf2)` som beregner verdien for *cosine similarity* mellom to TF-IDF vektorer.
3. Fyll ut metoden `get_response(self, query)` som tar en brukersetning som input, og returnerer svaret som forklart ovenfor. Metoden bør ta i bruk de to metodene `get_tf_idf` og `compute_cosine` som dere nettopp har implementert.