DD1418 Språkteknologi med introduktion till maskininlärning
Johan Boye, Dmytro Kalpakchi
2020-11-20

Uppgift 5

Litteratur: Läs kapitel 6 i Jurafsky-Martin, samt titta på föreläsning 9.

Kod: Skelettkoden kan laddas ned från Canvas eller från

`http://www.csc.kth.se/~jboye/teaching/language_engineering/a05/RandomIndexing.zip`

Unzippa koden i lämplig mapp. Öppna ett kommandofönster, gå till foldern `HMM` och skriv:

```
pip install -r requirements.txt
```

Nu ska allting du behöver för att göra labben vara installerat.

Problem:
Ordvektorer ("*word embeddings*") är ett sätt att representera semantiken hos enskilda ord som vektorer av reella tal. Det finns flera olika metoder att konstruera sådana vektorer, några ganska gamla (> 20 år). Området fick en renässans i och med artikeln om *word2vec* som publicerades 2013.

I den här labben ska vi undersöka en av dessa metoder: *Random Indexing*. Denna metod är enklare och mindre känd än *word2vec*, men ger ändå bra resultat. Din uppgift är att **utöka programmet random_indexing.py så att programmet skapar ordvektorer från de sju Harry-Potter-böckerna.**

Labben har tre delar:

1. **"Tvätta" råtexten**. Harry-Potter-böckerna finns tillgängliga som vanliga textfiler, com inte direkt kan användas för att skapa ordvektorerna. Till exempel, texten kan se ut på följande sätt:

```
1
HARRY POTTER
AND THE CHAMBER OF SECRETS
by
J. K. Rowling
(this is BOOK 2 in the Harry Potter series)
Original Scanned/OCR: Friday, April 07, 2000
v1.0
(edit where needed, change version number by 0.1)
C H A P T E R O N E
THE WORST BIRTHDAY
```

   Om man vill skapa ordvektorer från denna text, finns det flera problem:

   - det finns metatext, som `Original Scanned/OCR`;
   - det finns vissa typografiska egenheter, som ordet `C H A P T E R O N E`, och meningar som är brutna med newline;
   - skiljetecken sitter samman med ord, som i `needed,` (varför är detta ett problem?).

Vi kommer att ignorera de första två problemen i denna labb. Din uppgift är att implementera en funktion `clean_line`, som tar en rad text som indata, och returnerar raden utan skiljetecken och numeriska symboler som resultat.

För att bli godkänd: Kör `check_cleaned_text.sh`, som skriver ut skillnaden mellan din tvättade text och den korrekt tvättade texten. Om allt är korrekt så terminerar programmet utan att skriva ut någonting.

2. **Skapa ordvektorer**. Skriv kod som skapar ordvektorer genom använda Random Indexing. Detta innefattar två steg: att bygga en ordlista av alla ord som förekommer i Harry-Potter-böckerna, och sedan använda Random Indexing för att skapa ordvektorerna för dessa ord. **När du skapar ordvektorerna, antag att vänsterkontexten för första ordet och högerkontexten för sista ordet är tom**.

   För att bli godkänd: Funktionen `get_word_vector` ska returnera ordvektorn för ett givet ord ifall ordet finns i ordlistan, eller returnera `None` i annat fall.

3. **Hitta de närmaste orden**. Skriv kod som hittar de närmaste orden till ett givet ord genom att använda algoritmen *k-nearest-neighbours (kNN)*.. Du behöver inte implementera kNN-algoritmen själv, utan vi föreslår att du använder implementationen som finns tillgänglig i biblioteket `scikit-learn`.

   För att bli godkänd: Funktionen `find_nearest` ska kunna anropas med en lista med ord som indata, och ska då som resultat ge de 5 närmaste grannarna till varje ord, med en likhetssiffra för varje ord. Du ska kunna besvara följande frågor: Vilket likhetsmått kan du använda i din algoritm? Vilket är att föredra? Varför?

   Hitta de närmaste grannarna till följande ord:

   ```
   Harry, Gryffindor, chair, wand, good, enter, on, school
   ```

   Som ett exempel ger vår implementation följande 5 närmaste grannar till ordet `Harry`: `Hagrid, Snape, Dumbledore, Hermione`.

   Experimentera med att ändra några hyperparametrar till Random-Indexing-metoden, till exempel:

   - ändra vektorernas dimensionalitet till 10 med 8 nollsklida element (prova olika dimensionaliteter och antal nollskilda element).
   - ändra fönsterstorleken till värdena 2, 3, 10, och prova att gör vänster- och högerfönstrena antingen symmetriska och assymetriska.

   Vad had dessa ändringar för effekt på resultaten?