DD1418 Språkteknologi med introduktion till maskininlärning
Johan Boye
2020-10-28

Uppgift 2

Litteratur: Läs kapitel 3 i Jurafsky-Martin och titta på föreläsning 4.

Kod: Kodskelettet kan laddas ner från Canvas eller från

`http://www.csc.kth.se/~jboye/teaching/language_engineering/a02/LanguageModels.zip`

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

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

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

Problem:

1. Vi vill ha ett program som beräknar alla bigram-sannolikheter från ett givet träningskorpus, och sedan sparar dessa sannolikheter i en fil. Från till exempel filen `data/small.txt` vill vi producera innehållet i filen `small_model_correct.txt`. Notera att:

   - Den första raden innehåller två tal, separerade med ett mellanslag: Storleken på vokabulären V (=antalet unika löpord, inklusive skiljetecken), och storleken på korpuset N (=totala antalet löpord).
   - Efter det kommer V rader, där varje rad innehåller tre saker: ett identifierande index $(0, 1, \ldots)$, ett löpord, och antalet gånger det löpordet förekommer i korpuset.
   - Därpå följer ett antal rader, en för varje nollskiljd bigram-sannolikhet. Varje rad innehåller tre tal: Indexen till det första och andra löpordet i bigrammet, följt av de 15 första decimalerna av logaritmen av bigram-sannolikheten. Den naturliga logaritmen har använts (som man kan beräkna med hjälp av metoden `math.log`).
   - Filens sista rad är "-1" för att markera att filen är slut.

   `BigramTrainer.py`-programmet innehåller ett programskelett för att läsa ett korpus, räkna ut unigram- och bigram-sannolikheterna, samt skriva ut modellen.

   Er uppgift är att utvidga koden så att programmet fungerar korrekt. (Leta efter kommentaren `YOUR CODE HERE` i programmet.) Använd skripten `run_trainer_small.sh` och `run_trainer_kafka.sh` för att köra programmet på testexemplen small och kafka.

   Ni kan använda `-d` för att spara modellen till en fil:

   ```
   python BigramTrainer.py -f data/kafka.txt -d kafka_model.txt
   ```

   Då sparas modellen i filen `kafka_model.txt`.

   Genom att lägga till `--check` kan ni verifiera att era resultat är korrekta:

   ```
   python BigramTrainer.py -f data/kafka.txt --check
   ```

2. Ni kan nu generera ord enligt sannolikheterna givna av er språkmodell. Till exempel, om det senast genererade ordet var `röd` och modellen ger att `röd` kan följas av antingen `boll` med sannolikheten $p = 0.5$, `leksak` med $p = 0.3$, eller `bil` med $p = 0.2$ så ska nästa genererade ord vara `boll`, `leksak` eller `bil` med respektive sannolikheter 0.5, 0.3 och 0.2.

   `Generator.py` innehåller ett programskelett för att generera ord från en språkmodell på sättet beskrivet ovan. Utvidga koden så att programmet fungerar (ändra vid `YOUR CODE HERE`). Generera sen några ord från de olika modellerna ni tränade i föregående problem. (Om alla bigram-sannolikheter från det senast genererade ordet är noll, välj vilket ord som helst från korpuset med hjälp av en likformig fördelning.)

3. `BigramTester.py` innehåller ett programskelett för att läsa en modell i formatet beskrivet i problem 1, läsa ett testkorpus, och räkna ut entropin för testkorpuset givet modellen. (Cross-entropin av träningsdatan och testdatan)

   (a) Utvidga koden i `BigramTester.py` så att programmet fungerar korrekt (leta efter `YOUR CODE HERE`). Testdatans entropi beräknas som den genomsnittliga log-sannolikheten:

   $$-\frac{1}{N}\sum_{i=1}^{N}\log P(w_{i-1}w_i)$$

   där N är antalet löpord i testkorpuset. För att kunna hantera saknade ord och bigram, använd linjär interpolation:

   $$P(w_{i-1}w_i) = \lambda_1 P(w_i|w_{i-1}) + \lambda_2 P(w_i) + \lambda_3$$

   Värdena på konstanterna λ_1, λ_2 och λ_3 ges i koden för `BigramTester`-programmet. Skriptet `run_tester_small_kafka.sh` testar modellen tränad på `small.txt` genom att använda `kafka.txt` som testkorpus, och skriptet `run_tester_kafka_small.sh` testar modellen tränad på `kafka.txt` på testkorpuset `small.txt`. Jämför era resultat med våra genom att använda `--check` (Resultaten kan skilja sig något; till exempel om ni använder en annan logaritm. Vi använde den naturliga logaritmen.)

   (b) Bygg en modell från filen `data/guardian_training.txt` och en annan modell från filen `data/austen_training.txt`. Beräkna entropin för de två testfilerna `guardian_test.txt` och `austen_test.txt`, för båda modellerna. Rapportera era siffror och slutsatser från dessa experiment!