Sprachtechnologie

6. Übung

Prof. Tobias Scheffer
Uwe Dick

Sommer 2016

Ausgabe am: 31.05.16
Besprechung am: 06.06.16

Aufgabe 1 *Neuronale Netze (word2vec)*

In der Vorlesung haben Sie gelernt, wie ein Sprachmodell trainiert werden kann. Dieses kann genutzt werden, um jedes Wort in einen n-dimensionalen Vektor zu transformieren. Das in der Vorlesung vorgestellte Verfahren ist im Softwarepaket word2vec (https://github.com/dav/word2vec/) implementiert. Laden Sie dieses Toolkit herunter und installieren Sie es. Sie können auch eine Alternative verwenden:

- word2vec für MacOSX Maverics (https://github.com/h10r/word2vec-macosx-maverics),
- Gensim Python Bibliothek (https://radimrehurek.com/gensim/index.html).

1. Mit einem trainierten Sprachmodell (wie in der Vorlesung beschrieben) können Analogie-Anfragen beantwortet werden (z.B. Mann verhält sich zu König wie Frau zu Königin). Das Skript demo-analogy.sh (im scripts-Ordner) kann genutzt werden, um solche Analogie-Anfragen zu stellen. Dafür wird, wenn noch nicht vorhanden, im ersten Schritt ein Modell trainiert und anschließend geladen.
   (a) Stellen Sie die Anfrage: „man king woman“. Welches Wort ist am wahrscheinlichsten?
   (b) Stellen Sie die Anfrage: „germany berlin italy“. Welches Wort ist am wahrscheinlichsten?
   (c) Überlegen Sie sich min. zwei weitere Analogie-Anfragen. Diskutieren Sie mit den anderen Studenten, ob die gefundenen Analogien sinnvoll sind.
2. Mit einem gelernten Sprachmodell können auch semantisch ähnliche Wörter für eine gegebene Suchanfrage gefunden werden. Starten Sie dafür das Skript demo-word.sh. Welche Wörter sind ähnlich zu dem Wörtern:
   - berlin,
   - woman,
   - frog,
   - sex?

Aufgabe 2 *Sprachmodell*

In der Vorlesung wurde gezeigt, dass Sprachmodelle oft in Kombination mit einem Übersetzungsmodell, Rechtschreibmodell oder anderen Modellen kombiniert wird, um die wahrscheinlichste Wortfolge für eine gegebene Wortfolge zu berechnen. Nehmen Sie an, Sie haben die Suchanfragen einer Suchmaschine der letzten Wochen gesammelt und wollen ein Modell trainieren, welches für eine gegebene Suchanfrage (mit möglichen Fehlern) die wahrscheinlichste Suchanfrage zurückgibt. Formal berechnet sich die wahrscheinlichste Suchanfrage $X_1^*, X_2^*, \ldots, X_n^*$ für eine gegebene Suchanfrage $X_1, X_2, \ldots, X_n$ wie folgt:

$$X_1^*, X_2^*, \ldots, X_n^* = \arg\max_{Y_1,\ldots,Y_n} p_{\mathbf{SM}}(Y_1, \ldots, Y_n) \cdot p_{\mathbf{FM}}(Y_1, \ldots, Y_n | X_1, X_2, \ldots, X_n).$$

- Als Sprachmodell (**SM**) soll ein n-gram Modell mit n=2 trainiert werden. Schätzen Sie dafür alle Modellparameter (*Hinweis*: Schätzen Sie alle Wahrscheinlichkeiten $p(w_1), p(w_1|w_2), ...$ der Wörter w_i des Korpus). Damit ergibt sich als Sprachmodell:

$$p_{\mathbf{SM}}(Y_1, \ldots, Y_N) = p(Y_1) \cdot p(Y_2|X_1) \cdot \cdots \cdot p(Y_N|Y_{N-1})$$

- Als Fehlermodell (**FM**) sollen Sie ein einfaches Modell verwenden, welches testet, wie viele Buchstaben in zwei Wörtern übereinstimmen:

$$p_{\mathbf{FM}}(Y_1, \ldots, Y_N | X_1, \ldots, X_N) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\min(|Y_i|, |X_i|)} \sum_{j=1}^{\min(|Y_i|,|X_i|)} [[Y_{i,j} == X_{i,j}]],$$

  wobei $Y_{i,j}$ den j-ten Buchstaben des i-ten Wortes von Y referenziert.

Betrachten Sie den folgenden Korpus (in Klammern steht die Anzahl der Auftreten):

- berlin (15)
- berlin city (5)
- berlin tegel (6)
- berlin munich (1)
- berlin airport (3)
- berlin airport tegel (2)
- berlin train (2)
- berlin city hall (1)
- airport tegel (7)

1. Schätzen Sie alle Parameter des n-gram Modells.
2. Berechnen Sie die wahrscheinlichsten Suchanfragen für die folgenden gegebenen Suchanfragen:
   (a) berlon airprot
   (b) berlin thrain
3. Welche Probleme ergeben sich aus dem einfachen Fehlermodell? Überlegen Sie sich ein Fehlermodell, welches die beobachteten Probleme nicht besitzt.