Programmierkurs Python II – SS 2011

Übung 10

---

1 Vektoren (1 Punkt)

Vervollständige die Wörterbuch-basierte Klasse für Vektoren. Neben den vorgegebenen Methoden für Addition, Skalarmultiplikation, Cosinus und Distanz soll die Klasse auch Methoden für Subtraktion (`__sub__`), Division (`__truediv__`) und Centroide (`centroid(vecs)`, Klassenmethode) implementieren. Abhängig davon, wie Vektoren in Aufgabe 2 und 3 verwendet werden, kann es sinnvoll sein, zusätzlich die Methoden `__hash__`, `__eq__` und `__cmp__` zu implementieren, damit Vektor-Instanzen als Elemente von Mengen bzw. Schlüssel von Wörterbüchern verwendet werden können.

2 Semantische Ähnlichkeit (3 + 1 Punkte)

Implementiere ein Programm, das auf Grundlage einer gegebenen Dokumentsammlung die zu einem Eingabewort `w` die `n` ähnlichsten Wörter in der Dokumentsammlung berechnet. Das Programm soll die Dokumente einlesen, für alle Wörter im Dokument entsprechende Kontextvektoren berechnen, und diese dann mit dem Kontextvektor für das Eingabewort vergleichen. Ein Kontextvektor kodiert, wie häufig ein Wort mit anderen Wörtern innerhalb eines Kontextfensters – z.B. jeweils die 5 Wörter links vom aktuellen Wort – zusammen vorkommt (kookkurriert). Wir geben eine kleine Dokumentsammlung zum Testen vor (`nyt199407.tar.gz`). Experimentiere mit verschiedenen Kontextgrößen. Welche Werte liefern die intuitiv besten Ergebnisse? Welches sind die 10 ähnlichsten Worte zu *man*?

Bonus: Repräsentiere Wörter als Vektoren, die statt Kookkurenzen auf Kookkurenzen basierende PMI-Werte kodiert (siehe Einführungsfolien zu Machine Learning).

3 WSD mit dem kNN-Algorithmus (3 Punkte)

Implementiere den kNN-Algorithmus und benutze ihn für Word Sense Disambiguation. Trainiere Dein Programm mit den Trainingsdokumenten aus Übung 9 und klassifiziere die Test-Datei. Der kNN-Klassifizierer soll die gleichen Methoden unterstützen wie der Bayes-Klassifizierer aus Übung 9 (`readTrainingCorpus(class,file)`, `classify(file)`). Die Implementierung aus Aufgabe 1 bietet eine gute Basis für die Implementierung der 2. Variante (die Wort-Auftreten klassifiziert, mit Kontextfenstern). Wähle entweder die oder die Variante, die WSD als Dokument-Klassifikation auffasst.

Teste zwei unterschiedliche Distanzmaße für Vektoren und sage kurz, welchen Einfluss das Distanzmaß auf das Ergebnis hat.

4 k-means Algorithmus (Bonus, 3 Punkte)

Implementiere den k-means Algorithmus, mit einer der in der Vorlesung vorgestellten Abbruchbedingungen. Zum initialisieren kann man entweder zufällige Werte wählen, oder aus der Vektormenge zufällige Elemente wählen. Zufallszahlen können mit den Funktionen aus dem Modul `random` erzeugt werden.

Benutze die Dokumentsammlung `nyt199407.tar.gz` zum Testen.

Hinweis: Weitere Details und Implementierungshinweise finden sich in Manning, Raghavan und Schütze (2008), Kapitel 16:

`http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html`

Abgabe bis Donnerstag, 14.07.2011, 08:30 Uhr