Übung 5

Prof. Dr. Anette Frank
Tutorium: Thea Bartmann & Jakob Forstmann
Formale Semantik WiSe 2025/26
Frist: 27.11.2025 23:59

1 Aufgabe: Word Embeddings und Frames

In diesem Übungsblatt betrachten wir nicht-kontextualisierte Word Embeddings (GloVe) für Verben, die bestimmten FrameNet Frames zuzuordnen sind.

Da die Aufgaben Coding erfordern, nutzen Sie immer die Dokumentation zum jeweiligen Package. Für weitere Fragen steht Ihnen das Moodle Forum zur Verfügung.

1.1 Kosinusähnlichkeit in Frames - 10 Punkte

Abgabe: Code (py), Output (txt)

Im Moodle finden Sie Verben und zugehörige Vektoren für die Frames *Cutting* und *Damaging*. Die Vektoren aus den .npy Dateien können Sie mit dem folgenden Befehl laden. Er lädt einen 2-dimensionalen Numpy Array mit Shape (#AnzahlVerben, 300).

```
import numpy as np
vectors = np.load('Cutting.npy')
```

Berechnen Sie für jeden Frame einen Durchschnittsvektor (Centroid, **Zentroid**) auf Basis der Vektoren aller Verben des jeweiligen Frames.

Ermitteln Sie dann die Kosinus**ähnlichkeit**[1] (durch $1 - distance.cosine$ (Scipy)) zwischen jedem Verbvektor und dem Zentroiden und lassen Sie sich Verben samt Kosinusähnlichkeiten in **aufsteigender** Reihenfolge ausgeben.

Berechnen Sie außerdem die **durchschnittliche** Kosinusähnlichkeit von Verben und Zentroid als Maß für eine Intra-Frame-Similarity.

Machen Sie sich Gedanken zu Ihren Ergebnissen, indem Sie die folgenden Fragen beantworten (die Antworten sind Teil der Abgabe!):

[1] Kosinusähnlichkeit = 1 - Kosinusdistanz

1. Welcher der beiden Frames hat eine hohe Intra-Frame-Similarity und welcher eine nicht ganz so hohe?
   a) Wie erklären Sie sich dieses Ergebnis?
2. Was fällt Ihnen auf, wenn Sie die Verben nach ihrer Kosinusähnlichkeit zum Zentroiden ordnen?
   a) Welche Verben haben jeweils die höchste Ähnlichkeit zu ihrem Zentroiden?
   b) Was könnten Gründe dafür sein, dass andere Verben nicht so ähnlich zu ihrem Zentroiden sind?

Stöbern Sie hierfür auch gerne im Lexical Unit Index von FrameNet[2] und in WordNet[3]. Sie können beispielweise untersuchen, ob ein bestimmtes Verb auch noch in anderen Frames vorkommt oder ob es unterschiedliche Senses besitzt (WordNet Synsets). Vor dem Hintergrund, dass nicht-kontextualisierte Word Embeddings wie die in diesem Blatt verwendeten GloVe Embeddings nicht zwischen Senses unterscheiden, sollten Sie einige interessante Erkenntnisse gewinnen können.

Vermerken Sie Ihre Erkenntnisse bzw. die **Antworten auf die Fragen in Ihrer Abgabe** (als Kommentar im Code, als separate Datei oder als Teil des Jupyter Notebooks).

1.2 Plotten von Vektoren - 10 Punkte

Abgabe: Code (py), Plot (jpg, png, o.Ä.)

In dieser Aufgabe geht es darum, die Vektoren aus Aufgabe 1.1 zu visualisieren, um so die in der ersten Aufgabe erhaltenen Ergebnisse besser nachzuvollziehen zu können.

Da es sich um Vektoren mit 300 Dimensionen handelt, müssen wir zunächst die Dimensionalität reduzieren, um die Vektoren in einem zweidimensionalen Raum darstellen zu können. Verwenden Sie hierzu **PCA**[4] mit `n_components=2` und die Methode `fit_transform`.

Achtung: Übergeben Sie der Methode `fit_transform` **alle** Vektoren eines Frames gleichzeitig.

Plotten Sie die reduzierten Vektoren im Anschluss mit `matplotlib` und annotieren Sie die Punkte in Ihrem Plot (Hilfe finden Sie hier).

Speichern sie den Plot als Datei ab und reichen Sie sowohl Code als auch Plot ein.

[2] auch direkt via Python nutzbar, Hilfe im How-to und in der NLTK Dokumentation

[3] auch direkt via Python nutzbar, Hilfe im How-to und in der NLTK Dokumentation

[4] Mathematics for Machine Learning, Kapitel 10 bietet eine sehr gute, ausführliche und mathematische Erklärung zu PCA.

1.3 Clustern von Vektoren - 5 Punkte + 5 Bonuspunkte

Abgabe: Code (py), Output (txt)

Im Moodle finden Sie zusätzliche Dateien für verschiedene *Communication* Frames (*Communication-*{*manner, means, noise, response*}). Führen Sie auf Basis **aller** Verbvektoren **all** dieser Frames **Agglomeratives Clustering** mit `n_clusters=4` und der Methode `fit_predict` durch.

Machen Sie kenntlich, aus welchem Frame ein Verb jeweils stammt und welchem Cluster es zugewiesen wurde.

Evaluieren Sie ihr Clustering Ergebnis, indem Sie die **Purity** berechnen. Es kann dabei vorkommen, dass mehrere Cluster die gleiche majority class aufweisen.

Nennen Sie darüber hinaus zwei weitere Methoden, um Clustering Ergebnisse zu evaluieren.

Bonus Wählen Sie zwischen (a) oder (b):

(a) Wenden Sie PCA auf die Vektoren an und plotten Sie sie. Markieren Sie die Datenpunkte dabei nach ihrer Zugehörigkeit zu Cluster und Frame, indem Sie beispielsweise verschiedene Marker für die unterschiedlichen Cluster und verschiedene Farben für die unterschiedlichen Frames verwenden. Produzieren Sie einen Plot mit annotierten Datenpunkten (vgl. Aufgabe 1.2) und einen ohne.

Achtung: Übergeben Sie der Methode `fit_transform` (PCA) **alle** Vektoren **aller** Frames gleichzeitig.

(b) Wenden Sie neben dem Agglomerativen Clustering nun **K-Means Clustering** an. Experimentieren Sie mit verschiedenen Werten für `n_clusters` für jeden der beiden Algorithmen und vergleichen Sie jeweils die gebildeten Cluster. Für welchen Wert für `n_clusters` und mit welchem Algorithmus erhalten Sie das Ihrer Meinung nach beste (sinnvollste) Ergebnis und warum? Wenn möglich beziehen Sie sich bei der Beurteilung auf bekannte Cluster Evaluationsmetriken.

Führen Sie im Falle von K-Means stets mehrere Testläufe durch, da die anfänglichen Clusterzentren bei K-Means zufällig gewählt werden (der Algorithmus ist somit nicht deterministisch). Nähere Erläuterungen dazu finden Sie hier.

Vermerken Sie Ihre Erkenntnisse bzw. die **Antworten auf die Fragen in Ihrer Abgabe** (als Kommentar im Code, als separate Datei oder als Teil des Jupyter Notebooks).

1.4 Antonyme und Synonyme - 5 Punkte

Abgabe: Code (py), Output (txt)

Für Aufgabe 1.1 wurden Ihnen die Word Embeddings zur Verfügung gestellt. In dieser Aufgabe sollen Sie sie nun selbst extrahieren. Verwenden Sie hierzu das Python Package **Gensim** und laden Sie das Modell 'glove-wiki-gigaword-300' (Hilfe finden Sie hier).

Falls der Downloader Probleme bereitet, versuchen Sie es stattdessen über `last` oder laden Sie das Modell manuell herunter[5] und überführen Sie den Inhalt der entsprechenden Textdatei in eine geeignete Datenstruktur (Python Dictionary). Einen alternativen Workaround finden Sie im Materials Ordner in Moodle (*copy_ instructions.ipynb*).

Ermitteln Sie dann die Kosinusähnlichkeit zwischen den folgenden Wortpaaren:

- abundant - profuse
- fragile - sturdy
- radiant - luminous
- noisy - quiet
- jubilant - exultant
- temporary - permanent

Was fällt Ihnen auf, wenn Sie die Ähnlichkeitswerte für die Antonyme betrachten oder auch mit denen der Synonyme vergleichen?

[5] lädt mehrere Modelle, bitte das 300er verwenden