Goethe Universität
Institut für Informatik
Texttechnologie

NLP-gestützte Data Science

Übung 4

Sentiment-Klassifikation
SoSe 2022

Alexander Mehler Manuel Stoeckel Giuseppe Abrami

21.06.2022

Einleitung

Diese Aufgabe beschäftigt sich mit der automatischen Sentiment-Klassifikation. Sie werden sich mit Neuronalen Netzen, Word Embeddings und Text-Klassifikation auf Satz-Level beschäftigen. Als Grundlage für diese Übung dient die *Stanford Sentiment Treebank*[1] (Socher u. a., 2013).

Auf diese Übung gibt es insgesamt maximal **fünf Bonuspunkte**, die in Ihre Gesamtbewertung eingehen. Fall Sie mehr als fünf Bonuspunkte erreichen sollten, verfallen die übrigen Bonuspunkte ohne Effekt.

Aufgaben — **20 Punkte + max. 5 Bonus**

1 Klassifikation — **15 Punkte + max. 4 Bonus**

1. Implementieren Sie ein Bag-of-Words Modell für die Textklassifikation mit `PyTorch`.
   - Erstellen Sie dafür eine Klasse die `torch.nn.Module` erweitert oder verwenden Sie das Template aus OLAT.
   - Erstellen Sie eine modellhafte Skizze ihrer Architektur und erklären Sie Ihren Ansatz in einem der Abgabe beiliegendem **PDF Dokument**.
2. Laden Sie die Trainingsdaten und implementieren Sie notwendige Methoden für das Pre-Processing.
   - Implementieren Sie hierzu eine Klasse, die `torch.utils.data.Dataset` erweitert und die Trainingsdaten aus den gegebenen Dateien laden kann, und sie für einen Dataloader (`torch.utils.data.DataLoader`) verfügbar machen kann.
3. Erstellen Sie ein Trainingsskript für Ihr Modell (z.B. mit dem Template in `main.py`).

[1] Weitere Informationen unter: https://nlp.stanford.edu/sentiment/index.html

4. Trainieren Sie Ihr Modell auf dem Trainingsdatensatz aus OLAT (`train.tsv`, `dev.tsv`, `test.tsv`).

5. Dokumentieren Sie Ihre Ergebnisse in einem begleitenden **PDF Dokument**.
   - Wie gut sind Ihre Ergebnisse? Vergleichen Sie ggf. mit den Werten aus der Vorlesung *Kapitel 5: Sentiment Analysis II*, Folie 63.

Ihr Modell **muss** folgende Komponenten enthalten:

- Encoding-Layer: Erstellt Satz-Repräsentationen aus Word Embeddings.
- Classification-Layer: Ein Klassifikator der für eine Satz-Repräsentation eine Wahrscheinlichkeitsverteilung über die möglichen Klassen berechnet.

Hinweise

- Verwenden Sie die `torch.nn.Embedding` Klasse um Word-Embeddings zu erstellen und *poolen* Sie die resultierenden Embeddings in einzelne Satzrepräsentationen (z.B. mit *Average Pooling*, *Min Pooling* oder *Max Pooling*).
- Verwenden Sie die `torch.nn.Linear` Klasse für Ihren Klassifikator und die *Cross Entropy Loss* Funktion zum Training.

Bonus

SST-ternary — 1 Bonus

1. Vereinen Sie die beiden Varianten des positiven und negativen Sentiments aus dem gegebenen Datensatz jeweils in eine einzelne Kategorie (also insgesamt drei Kategorien).
2. Trainieren Sie auf den entstandenen Daten und vergleichen Sie die Ergebnisse.
3. Dokumentieren Sie die beobachteten Veränderungen im begleitenden PDF Dokument.

Pre-Trained Embeddings — 1 Bonus

1. Erweitern Sie Ihr Modell um vor-trainierte Embeddings, z.B. Word2Vec (Mikolov, Chen u. a., 2013; Mikolov, Sutskever u. a., 2013), GloVe (Pennington, Socher und Manning, 2014) oder fastText (Joulin u. a., 2017) Embeddings.
2. Trainieren Sie erneut auf den Daten und vergleichen Sie die Ergebnisse.
3. Dokumentieren Sie die beobachteten Veränderungen im begleitenden PDF Dokument.

Textklassifikation mit FastText — 1 Bonus

1. Laden Sie sich `fastText`[2] herunter (als Binary oder als Python Modul via `pip`).
2. Trainieren Sie ein `fastText` Modell auf den Trainingsdaten aus OLAT.
   - Sie müssen hierzu die Daten ggf. umformatieren. Dokumentieren Sie Ihre Vorgehensweise.
3. Werten Sie die Ergebnisse aus und dokumentieren Sie diese in Ihrem PDF Dokument.
   - Sind die Ergebnisse mit `fastText` besser oder schlechter, und warum?

[2] https://fasttext.cc/docs/en/support.html

Beyond Bag-of-Words **3 Bonus**

1. Implementieren Sie ein weiteres Modell zur Textklassifikation Ihrer Wahl, das über das einfache Bag-of-Words Modell hinausgeht.
   - Erstellen Sie dafür eine Klasse die `torch.nn.Module` erweitert oder verwenden Sie das Template aus OLAT.
   - Erstellen Sie eine modellhafte Skizze ihrer Architektur und erklären Sie Ihren Ansatz in einem der Abgabe beiliegendem **PDF Dokument**.
2. Trainieren Sie das Modell auf den Trainingsdaten aus OLAT.
3. Werten Sie die Ergebnisse aus und dokumentieren Sie diese in Ihrem PDF Dokument.

2 Evaluation **5 Punkte + 1 Bonus**

2.1 Precision, Recall, F_1-Score **3 Punkte**

Implementieren Sie:

- Precision,
- Recall und
- F_1-Score.

Berechnen Sie alle drei Werte für die Beispiel Ergebnisse für Aufgabe 1, die Sie in der Datei `test.preds.json` finden können, oder alternativ für Ihre eigenen Ergebnisse aus Aufgabe 1[3] – jeweils für jedes Label einzeln, sowie in micro-, macro- und gewichtetem Durchschnitt, wobei die Gewichte nach dem jeweiligen Anteil des korrekten Labels zu wählen sind. Fügen Sie die Ergebnisse Ihrer begleitenden PDF hinzu.

2.2 Inter-Annotator-Agreement **2 Punkte**

Berechnen Sie das Inter-Annotator-Agreement der Annotationen aus Übung 3.

- Nutzen Sie hierzu *Krippendorff's alpha*, mit Implementierung Ihrer Wahl.
- Die Annotationen finden Sie in OLAT in der Datei `uebung3-ergebnisse.json`.

Bonus

Statistik & Visualisierung **1 Bonus**

Erstellen Sie Visualisierungen für die Ergebnisse von Übung 3 (`uebung3-ergebnisse.json` zusammen mit `uebung3-tweets.json`) wie folgt:

1. Die Anzahl der Tweets pro Sentiment Label, z.B. in einem Bar Chart.[4]
2. Die Anzahl der Tweets pro Sentiment, die einen bestimmten Hashtag enthalten, z.B. in einem Stacked Bar Chart.
3. Die Anzahl der Tweets pro Sentiment, die einen bestimmten Smiley enthalten, z.B. in einem Stacked Bar Chart.

Betrachten Sie jeweils nur Hashtags oder Smileys, die ausreichend häufig vorkommen.

[3] Während Sie für die Evaluation von Aufgabe 1 Fremd-Code verwenden dürfen, **müssen** Sie die drei Funktionen hier selbst implementieren.

[4] Annotationswerte Schlüssel: -2 = Kein Sentiment, -1 = Negativ, 0 = Neutral, 1: Positiv+Negativ 2: Positiv

Abgabe der Aufgabe

Für die Bearbeitung dieser Aufgabe stehen Ihnen **vierzehn** (14) Tage zur Verfügung. Bitte geben Sie Ihre Lösungen bis Dienstag, den 05.07.2022, spätestens um 16:00 Uhr (d.h. vor Beginn der Vorlesung) ab. Die Abgabe sollte aus Ihrem dokumentierten Quell-Code und genau einem begleitendem PDF Dokument bestehen.

Literatur

Barnes, Jeremy, Lilja Øvrelid und Erik Velldal (Aug. 2019). „Sentiment Analysis Is Not Solved! Assessing and Probing Sentiment Classification“. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, S. 12–23. DOI: 10.18653/v1/W19-4802. URL: https://aclanthology.org/W19-4802.

Joulin, Armand u. a. (Apr. 2017). „Bag of Tricks for Efficient Text Classification“. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, S. 427–431. URL: https://aclanthology.org/E17-2068.

Mikolov, Tomas, Kai Chen u. a. (2013). „Efficient Estimation of Word Representations in Vector Space“. In: *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Ilya Sutskever u. a. (2013). „Distributed Representations of Words and Phrases and Their Compositionality“. In: *Advances in neural information processing systems*, S. 3111–3119.

Pennington, Jeffrey, Richard Socher und Christopher D Manning (2014). „Glove: Global Vectors for Word Representation“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, S. 1532–1543. DOI: 10.3115/v1/d14-1162.

Socher, Richard u. a. (Aug. 2013). „Parsing with Compositional Vector Grammars“. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, S. 455–465. URL: https://aclanthology.org/P13-1045.