Buchstabenbasierter Wortart-Tagger

Sie sollen den Tagger aus der letzten Übungsaufgabe verbessern, indem Sie die Wort-Embeddings durch eine Repräsentationen ersetzen, die aus den Buchstabenfolgen der Wörter berechnet werden. Dazu wird ein weiteres bidirektionales LSTM benutzt, welches über die Buchstabenfolge iteriert. Die beiden Endzustände dieses bidirektionalen LSTMs werden konkateniert und bilden die Repräsentation des Wortes.

Was müssen Sie ändern?

- Die Klasse **Data** erstellt nun eine Indextabelle für Buchstaben statt für Wörter. Buchstaben und Zeichen, die nur einmal aufgetaucht sind, erhalten keinen Index. Leerzeichen sollen von der Tabelle auf den Wert 1 abgebildet werden. Für unbekannte Zeichen wird der Index 0 reserviert. Analog zum Attribut `numTags` definieren Sie ein weiteres Data-Attribut `numChars`. Der Hyperparameter `numWords` fällt nun weg.
- Die Methode **words2IDs** wird durch eine Methode **words2IDvecs** ersetzt. **words2IDvecs** iteriert über alle Wörter und extrahiert für jedes Wort das Präfix und das Suffix der Länge 10. Die Reihenfolge der Präfixbuchstaben werden dabei umgedreht. Wenn nun ein Affix kürzer ist, wird es mit dem Befehl `rjust` am Anfang mit Leerzeichen aufgefüllt. Die Buchstaben werden schließlich mit Hilfe der Indextabelle auf Zahlen abgebildet, wobei unbekannte Zeichen den Index 0 erhalten. Die Funktion gibt eine Matrix (Liste von Listen) mit Präfix-Buchstaben-IDs und eine Matrix mit Suffix-Buchstaben-IDs zurück. Die i-te Zeile enthält jeweils die Buchstaben-IDs des Präfixes/Suffixes des i-ten Wortes.
- Die Wort-Embedding-Ebene im neuronalen Netz wird durch ein bidirektionales **buchstabenbasiertes LSTM** ersetzt. Da sich die Eingabe des Forward-LSTMs (Suffixe) von der des Backward-LSTMs (Präfixe) unterscheidet, kann dieses BiLSTM nicht mit der `bidirectional`-Option von `nn.LSTM` implementiert werden. Stattdessen erzeugen Sie für jede Richtung ein **separates** LSTM. Der 2-dimensionale Tensor mit den Buchstaben-Indizes der Wortsuffixe (analog Wortpräfixe) wird durch einen Buchstaben-Embedding-Lookup in einen 3D-Tensor transformiert, der dann mit dem buchstabenbasierten Forward-LSTM für Suffixe (analog einem Backward-LSTM für Präfixe) verarbeitet wird. Die Buchstabenfolgen aller Wörter werden dabei parallel verarbeitet. Dafür müssen Sie die `batch_first`-Option der Buchstaben-LSTMs auf `True` setzen, da LSTMs bei einem 3-dimensionalen Tensor per Default über die erste Dimension iterieren. Die Endzustände dieser beiden buchstabenbasierten LSTMs werden dann für jedes Wort konkateniert und bilden die Eingabe des übergeordneten wortbasierten LSTMs. Für die Größe der Forward/Backward-LSTMs definieren Sie einen neuen Hyperparameter `charLstmSize`.
- Nun müssen Sie noch Ihr Annotations-Programm `tagger-annotate.py` anpassen, damit es mit dem modifizierten Modell arbeiten kann.

Die Größe der Buchstaben-Embeddings können Sie im Bereich 50–100 wählen. Es sind Genauigkeiten bis zu etwa 95% möglich.

Abgabe: Ihren vollständigen Programmcode und die nach jeder Epoche erzielte Genauigkeit auf den Development-Daten mit Ihren besten Hyperparametern.