Homework 7: Information Extraction with Distant Supervision

Due April 14, 2021 (11:59 PM)

1 Introduction

For this homework, you will implement a distant supervision algorithm to align structured Wikidata triples to sentences from Wikipedia. We will then use the aligned dataset to train an Information Extraction classifier, which predicts the relation between two entities in a sentence.

Wikidata is a large database of structured information stored as triples. An example of a triple is (has_place_of_birth, Madonna, Michigan). Wikidata is incredibly large, containing 93,300,891 items[1] about entities included in Wikipedia articles. While these articles cover the same entities as Wikidata, we need to use *distant* supervision because we don't know exactly which sentences in a Wikipedia article describe any specific relational triple in Wikidata. For example, Wikidata contains the relational triple (has_place_of_birth, Madonna, Michigan), and the Madonna Wikipedia article includes the sentence:

> Born and raised in Michigan, Madonna moved to New York City in 1978 to pursue a career in modern dance.

Given this triple and sentence, we can use distant supervision to create the following input:

> Born and raised in **Michigan**$_{ENT2}$, **Madonna**$_{ENT1}$ moved to New York City in 1978 to pursue a career in modern dance.

Given this input, a classification model can predict the relation between $ENT1$ and $ENT2$ to be has_place_of_birth relation because we observe that is the relation between Madonna and Michigan in the Wikidata triple.

For this homework, we will be working with a dataset of American musicians and the entity relations date of birth, place of birth, school attended, start of musician career, and is member of band.

The link to this homework's colab is: `https://github.com/dbamman/nlp21/blob/main/HW7/HW_7.ipynb`

2 Deliverable 1: Implement Distant Supervision

We will be using the Distant Supervision algorithm described in SLP3 Ch. 17 to align Wikipedia triples and sentences from Wikipedia text. This algorithm assumes a relation is depicted in text if both entities are included in the sen-

[1] `https://www.wikidata.org/wiki/Wikidata:Statistics`

tence. For example, both of the following sentences would be identified as including (has_place_of_birth, Madonna, Michigan) despite their semantic differences:

> Born and raised in **Michigan**$_{ENT2}$, **Madonna**$_{ENT1}$ moved to New York City in 1978 to pursue a career in modern dance.

> **Madonna**$_{ENT1}$ recently sold-out an arena in **Michigan**$_{ENT2}$, selling 100,000 tickets in 5 minutes.

Your task for this deliverable is to first determine whether the two entities in an information triple are present in a given sentence from a Wikipedia article. For this homework, an entity is present if the entire string can be found in the sentence. For this deliverable, the *should_align* variable should be set to True if both entities are present in the sentence. (Do not overthink this deliverable!)

3 Deliverable 2: Implement positional embeddings

Similar to HW3, we will be using a CNN classification model with GLoVE embeddings. Your deliverable for this component is to implement the position-based embeddings discussed in lecture.

A figure of this model has been reproduced from lecture slides:

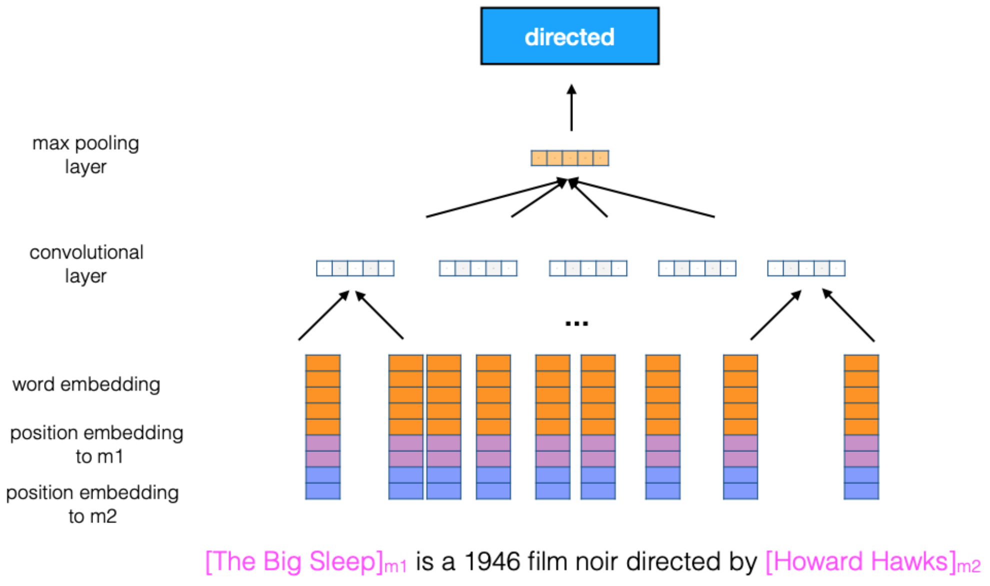

Figure 1: Position embedding model to be implemented.

For this deliverable, we will be working with a different, larger dataset than the one created in Deliverable 1. This dataset additionally includes examples labeled $no_relation_found$, indicating a sentence which was not aligned to a Wikidata triple but containing 2 entities identified by an Named Entity Recognition system[2].

This deliverable will contain two components: (1) determining the position embedding values and (2) initializing and indexing position embedding matrices for our CNN Classifier.

[2] https://spacy.io/usage/linguistic-features#named-entities

3.1 Deliverable 2.1: Determining Position Embedding Values

When we are processing the dataset for our model, we want to calculate the distance from each word to each entity (m1 and m2).

For this deliverable, you will add code to the format_data() function to calculate each word's distance from each entity and store them in the corresponding m1_pos_list and m2_pos_list variables.

You will first need to determine where m1 and m2 are located in the sentence (Hint: m1 includes the _ent1_ prefix we used in Deliverable 1 and m2 includes the _ent2_ prefix). Then, you will need to calculate each word's distance from each entity.

Since we cannot access an embedding matrix in pytorch via negative values, we will need to treat negative distances differently. To address this, we will take the maximum length of a sentence (300) and use the first 300 values as positive indices (e.g. position 5 would be stored as 5, position 17 would be stored as 17, etc.). For negative values, we will start indexing at position 300 (e.g. position -2 would be stored as 302, position -15 would be stored as 315, etc.). Your final m1_pos_list and m2_pos_list should contain all positive integers ranging from 0 to 600.

3.2 Deliverable 2.2: Initializing New Embedding Matrices and Indexing

Now, we will create and train our CNN network. For this deliverable, begin by initializing 2 embedding matrices in the __init__() method: one for position embedding to m1 and one for position embedding to m2. These should be embeddings initialized **from scratch** and should update during training. The number of embeddings should be 600, to account for all possible position values, and the size of each embedding should be 16. In the forward() method, fill in the code to access the appropriate embeddings for m1_pos_list and m2_pos_list.

4 How to Submit

1. Download your Colab notebook as an .ipynb file (File → Download .ipynb)
2. Submit **HW_7.ipynb** to the Homework 7 assignment on Gradescope. The files must be named **HW_7.ipynb** for the Gradescope autograder.