Lab03: Sentences' similarity based on their words

Abdelkrime Aries

Sentences' similarity measure can be beneficial for many NLP tasks. It can improve Information Retrieval (IR) by introducing the similarity between the query and the documents. It can help in reducing redundancy in Automatic Text Summarization (ATS) by deleting similar sentences from the summary. Another application is "Plagiarism Detection"; similar sentences can be detected from a database of documents. Similarity measure can even be used as a quality measure for Machine Translation (MT); how similar automatic translated sentences to manual ones.

1 Program description

Our objective is to create a program which measures two sentences' similarity. To do this, we want to use vectors similarity measures. In this case, a sentence must be represented as a vector.

There are many approaches to encoding a sentence. Since this lab is limited by time, we will use words' encoding to infer a sentence's encoding. Our architecture is illustrated as a simple class diagram in figure 1. The bold red attributes and methods must be completed by the students.

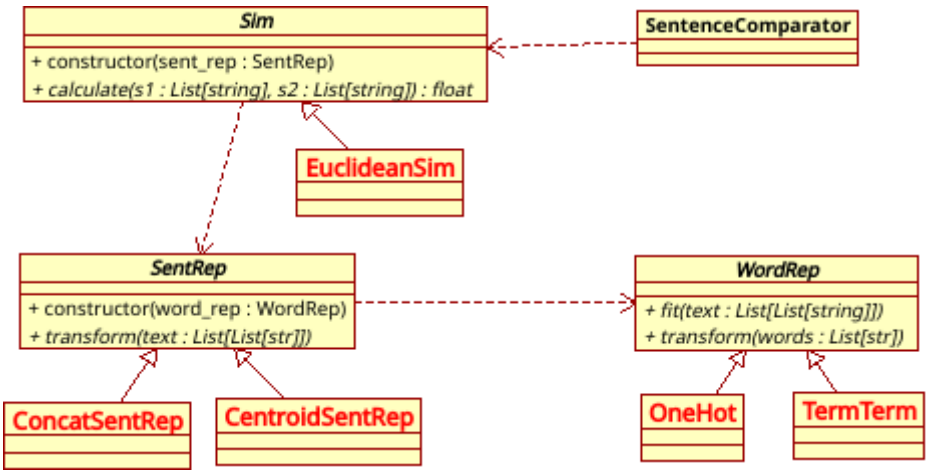

Figure 1: Class diagram of our similarity program

1.1 WordRep class

This class is an abstract class defining two methods to be overridden: fit and transform. The constructor can be overridden if we need more parameters. You have to implement two classes:

- OneHot: onehot representation of a word. It must add "[UNK]" as the first token. If special tokens are given, they must be considered after "[UNK]".
- TermTerm: term-term representation. It has to allow defining thee window.

The tokens are added by order of appearance, not by alphabet.

1.2 SentRep class

This class is an abstract class defining one method to be overridden: transform. The constructor can be overridden if we need more parameters. It uses **WordRep** as a base to encode a sentence's words, then it generates a vector representing a sentence based on its words. You have to implement two classes:

- ConcatSentRep: it has a maximum number of words. If a sentence exceeds this number, it will be truncated. If its length is less, the remaining will be completed by zeroes.
- CentroidSentRep: it calculates the centroid of all words in a sentence.

1.3 Sim class

This class calculates a similarity between two sentences. You have to implement the Euclidean distance based similarity given by thee following equation:

$$sim(X, Y) = \frac{1}{1 + ||X - Y||}$$

1.4 SentenceComparator class

*****NOTHING TO IMPLEMENT HERE*****

This class just compares many sentences.

2 Questions

Answer these questions at the beginning of your code, as comments:

1. Why the two sentences ”cats chase mice” and ”mice chase cats” are considered similar using all words and sentences encoding? Propose a solution to fix this.
2. Why using concatenation, the 2nd and 4th sentences are similar? Can we enhance this method, so they will not be as similar?
3. compare between the two sentence representation, indicating their limits.

3 Students' Grading

- Deadline: till midnight minus 1 minute
- Grade
  - each class 3 pts which means 15pts
  - questions: 3 points
  - timeliness: 2 points each 15 minutes after midnight is -0.25. Then, you'll have a 0.

  You have to send a ”.py” file; no cloud files are allowed.

4 Useful functions

There are some implemented functions, you may use:

- vec_plus(X: List[float], Y: List[float]) -¿ List[float]: element-wise addition.
- vec_divs(X: List[float], s: float) -¿ List[float]: division of a vector on a scalar.