Computational Linguistics

CSE/LIN 467/567 – Spring 2018

Homework 4

Due date: end of May 2nd

Instructions

Note: **Answers not submitted as specified below will not be graded**.

Upload to UBLearns a single zip file containing all the files corresponding to your answers. The zip file must be named '`hw2UBitNAME.zip`, where UBitNAME is your UB email's name, and the same naming convention should apply to any files in the zip file. Don't forget to click the button after uploading the file, or it will not be submitted.

Exercises

1. Construct the CKY parse chart for the sentence '*The evidence in the stolen car proves you stalked Robin*', using the following grammar rules (and an appropriate set of words). If the CNF constraint is not satisfied, convert the grammar into CNF.

NP → DT N	N → N PP	N → Adj N	PP → P NP	S → NP VP
VP → IV	VP → TV NP	VP → DTV NP NP	VP → VP PP	VP → SV S
NP → PN	NP → PRP			

2. As discussed in class, sparse word vectors can be of extremely high dimensionality (equal to the size of the corpus). Random indexing is a dimensionality reduction method in which a high-dimensional model is projected into a space of lower dimensionality without compromising distance metrics. Annexed to this homework you will find three files:

   - `trigramModel.pl`: a trigram model containing only common nouns, built from TDA.txt, by extracting only the nouns. The tagger is not perfect, and therefore there is some error, but it has little effect in the present task.
   - `wordVectors.pl`: a model of random vectors for all the nouns in the trigram model. Such vectors are of size 150, and encode all the 19k common nouns.
   - `rindex_cosine.pl`: an implementation of cosine word similarity using random indexing, which is incomplete.

   Your goal is to complete the implementation in `rindex_cosine.pl`, by implementing two predicates:

   (a) `cosine/3`, which implements the cosine similarity. Examples below are fictitious.

```
?- similarity(world,planet,X).
X = 0.813134
```

```
?- similarity(world,dog,X).
X = 0.410689

?- similarity(world,banana,X).
X = 0.230858
```

(b) `most_similar/2`, which replaces a word with its most similar word, according to the cosine similarity. Examples below are fictitious. Basically, it must find the similarity values between input word and all words in the vector model, and order their values.[1] The computation for the most similar word may take a minute or two to complete.

```
?- most_similar(planet,X).
X = world ;
X = country
```

[1]Going back to the slides in prolog1.pdf, it may be of use to note that a query like `findall( related(X,Y), is_ancestor_of(X,Y), L)` will instantiate `L` with a list of all entities directly or indirectly related by the ancestor relation, e.g. `[related(aragorn,eldarion), related(ilsidur,eldarion)....]`. Note that `related/2` need not exist anywhere in the database. Such `related/2` relations are created by `findall` on-the-fly. This technique may be of use to quickly collect all word-similarity pairs. The bult-in `sort/4` command will also be useful, e.g. `sort(0, @>, [ t(6,cat), t(3,dog) ], L)` orders the elements of a list by decreasing order of their first argument, e.g. yielding `L = [ t(6,dog), t(4,cat) ]`. See the SWI manual for more details.