Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology

D7041E "Applied artificial intelligence"

(Please report any found inconsistencies to Evgeny as soon as you find them)

LAB 2: Word embedding: connectionist and distributional approaches

1. Introduction

In this lab we will work with data-driven models for learning words' representations. The task is to form a vector representation for each word in the vocabulary given a corpus of text. The challenge here is to provide vector representations of words such that the relationship between vectors mirrors the linguistic relationship between words. In this lab you will work with two methods: one for each class of models. Word2Vec is a popular tool for building word embedding using connectionist models. Random Indexing is one of distributional models, which uses principles of hyperdimensional computing.

As the training data you will use TASA corpus of school reading materials from kindergarten through high school. It contains approximately 37,600 text samples. There is also the evaluation task for the learned embeddings: the Test of English as a Foreign Language (TOEFL) synonymy assessments. The dataset (file "new_toefl.txt") contains 80 synonym tasks. Each task includes a query word and 4 alternatives. One of the alternatives is a synonym of the query word. If the chosen word is the correct answer (i.e. the synonym) than the score on the overall dataset increases by 1. The overall score on the TOEFL synonymy assessment is an integer number between 0 and 80. Note that if choosing answers randomly (e.g. by tossing a coin twice) the score averaged after many runs will be close to 20 or 25% (20/80) (i.e. chance of randomly guessing the correct answer out of four alternatives).

In this lab you are provided with two Python projects GensimW2V (for connectionist approach to word embedding) and RI (Random indexing method), which are available in Canvas.

Note. In order to execute the provided code you might have to install several packages: the library implementing Word2Vec (e.g, by `easy_install -U gensim`) and natural language toolkit (ntlk). You may also have to download corpora for ntlk package in order to perform the lemmatization.

Note. Usually before building word embedding models the original text is pre-processed. Common pre-processing steps include disregarding the most frequent words (e.g., articles "a" and "the") and lemmatization (read more about it https://en.wikipedia.org/wiki/Lemmatisation), which changes words to their default forms (e.g., were -> be; goes -> go; tables -> table; etc.). You are already provided with the lemmatized text in the file "lemmatized.text". In order to see the difference take a look on the original TASA corpus (file tasa.text) in the root folder for this Lab.

Note. The lemmatization is still used in the code in order to process the words in the TOEFL dataset.

Note. The TASA corpus used in this lab was made available for use only in academic research, courtesy of Touchstone Applied Sciences Associates.

2. Word2Vec

1. You have got the code to train word embedding using neural networks.
2. The code also estimates the performance on TOEFL task
3. Get the performance of the model for three different dimensionalities. The choice of dimensionalities is on your own but use different values (e.g., 10, 100, 1000).
4. If your computational resources allow run simulations several times (e.g. 5) for each dimensionality
5. Report the accuracy on TOEFL for all simulations
6. Elaborate how accuracy changes with the dimensionality

3. Random indexing with permutations

1. You have got the code to train word embedding using Random indexing.
2. The code also estimates the performance on TOEFL task
3. The current code works adequately only when size of window is 2. What should be changed in order to overcome this issue?
4. Get the performance of the model for three different dimensionalities. The choice of dimensionalities is on your own but use different values (e.g., 1000, 4000, 10000).
5. If your computational resources allow run simulations several times (e.g. 5) for each dimensionality
6. Report the accuracy on TOEFL for all simulations
7. Elaborate how accuracy changes with the dimensionality

As a conclusion elaborate of the accuracy of both methods for synonyms tasks. Also compare how computational demanding are both methods. Remember that RI forms representations only for the subset of the vocabulary, which is needed for TOEFL tasks. You could train all words and check how much time is needed or estimate the training time based on the time needed to train the subset

Final Note. Be patient as training data-driven models take time :-). Word2vec reports its progress while executing. RI does not. You are more than welcome to implement this functionality.

Congrats, you have just become familiar with the frontier methods for data-driven machine semantics! Well done!