On Approximately Searching for Similar Word Embeddings

Kohei Sugawara, Hayato Kobayashi, Masajiro Iwasaki
{k sugawar, hakobaya, miwasaki}@yahoo-corp.jp

Introduction

Background

• Searching for the (k-nearest) similar word embeddings is one of the most basic operations in NLP applications, e.g., extracting synonyms, inferring the meaning of polysemous words, aligning words in two sentences in different languages, solving analogical questions, and searching for documents related to a query.
• Gorman and Curran (ACL 2006) reported that SASH (tree-based method) performed the best for count-based embeddings

Purpose

• Address how to quickly and accurately find similar embeddings

Contributions

• Focus on neural word embeddings (dense vectors) learned by a recently developed skip-gram model [Mikolov+, 2013]
• Show that a graph-based search method (NGT) clearly performs better than SASH from different aspects

Experiments

Task Settings

• Search for k-nearest neighbor embeddings close to a given vector in a test set after indexing with a training set
• 1K embeddings are extracted as a test set by random sampling
• Plot the average precision versus its computation time (log-scale) by changing the parameter for precision of each method

Basic Settings

• Distance Function : Normalized distance
• Dimension : 200-dimensional word embeddings
• Top@k : Top-10 nearest neighbors
• Data Size (for indexing): 2 million words
• Data Source (for training): English Wikipedia in February 2015
• Model : Skip-gram model with hierarchical softmax (word2vec)
• Task : Search task

Change Distance Function

• Precision versus computation time of Normalized, Angular and Euclidean

NTG (Normalized) performed the best for cosine similarity
SASH (Angular) performed relatively well, but the indexing time with the angular distance is larger than the Euclidean distance
“Normalized” performed generally better than “Euclidean”
Why is “Normalized” faster than “Euclidean”?

NGT (Normalized) performed the best for cosine similarity
SASH (Angular) performed relatively well, but the indexing time with the angular distance is larger than the Euclidean distance
“Normalized” performed generally better than “Euclidean”
Why is “Normalized” faster than “Euclidean”?

Normalization can align embeddings so as to divide the entire search space more efficiently.

Change Dimension, Top@k, Data Size

• NGT performed the best (Please see the details in our paper)

Similarity Search Algorithms

• Three types of metric-based indexing are generally used in approximate similarity search as below:
  • Hash-based indexing is a method to reduce the dimensionality of high-dimensional space by using some hash functions
  • Basically designed for radius search, not k-nearest search
  • Tree-based indexing is used to recursively divide the entire search space into hierarchical subspaces
  • Descending from the root node to the leaf nodes in the tree structure and scanning only neighbors belonging to the subspaces
  • Graph-based indexing is a method to find nearest neighbors using a approximate neighborhood graph
  • Traversing neighbors on the graph from a certain node

• Compared search algorithms (available online) as below:

Change Data Source and Model

• Precision versus computation time of GNews, C&W and Glove embeddings

NGT performed the best when changing data source and model
GNews and Wikipedia (skip-gram) had almost the same tendency

Performance can be affected by learning models
Why is “GloVe” so slow?

“GloVe” has several high kurtosis (tailedness) peaks, which means the large variations of nearest neighbor distances can harm efficient indexing.

Change Task

• Precision versus computation time of semantic/syntactic analogy task

NGT performed the best for analogy task
How is search task related to analogy task?

Analogical precision can be far different from estimated precision by search task.

• Might be sufficient for another task even if the precision of search task is not so good

When the search precision by NGT is 0.8, the analogy precision 0.75 is unexpectedly high, although its naive estimation (black dot) is 0.64

NGT is available on website (glue codes for comparing algorithms coming soon)
http://research-lab. lab. yahoo.co.jp/software/ngt/