Semantic Similarity Search

Josh Hedtke, Stephen Su, Albert Zuo

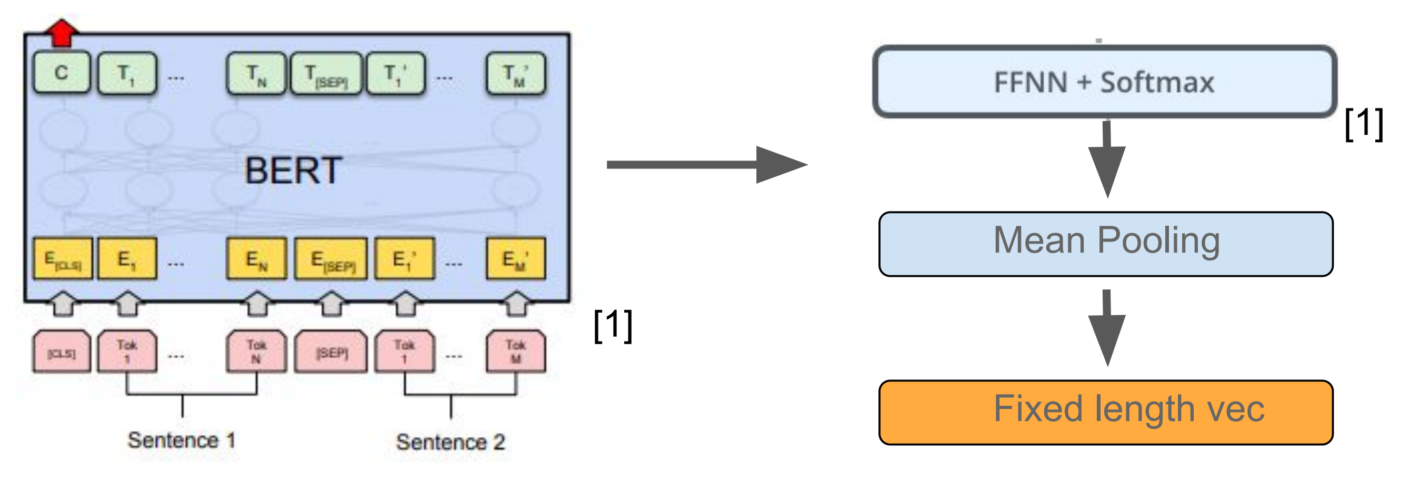
Stanford

Overview

- How do we search for similar, potentially lengthy, phrases in a database of N=**100b+ phrases**?
- We used transfer learning to fine tune the **BERT** model [4] using a fully connected single layer trained on labeled paraphrase pairs to build the comparison metric
- We then built an **approximate nearest neighbors** algorithm using **LSH** to search for nearest neighbors in BERT-space using cosine similarity as the distance

Comparison Metric

Fine-tuned BERT

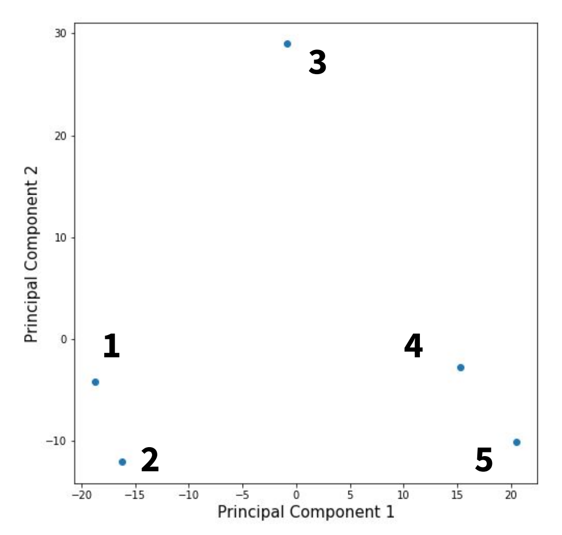


Data

- Train: MRPC (3600 paraphrase pairs)
- Validation: domain-specific, hand labeled paraphrase pairs (89 pairs)

Results

Query Phrase	Ranked Matches
'they said before i had a visa'	[' they told me I had a visa earlier', ' after we finish with this i have a totally separate question', ' I am outside of the country and need a notification on my card']
'how may I help you'	[' how can I be of assistance to you', ' thank you for calling usaa', ' bear with me one moment']



1: 'they said before i had a visa'
2: 'they told me I had a visa earlier'
3: 'when we have to we transfer funds into my checking account'
4: 'who do I have the pleasure of speaking with'
5: 'I did it online before'

N Components	Expl. Variance
2	0.619
3	0.867
4	1.000

LSH Algorithm

- **Intuition**: The key ingredient is the use of a family of locality-sensitive hash (LSH) functions, which are more likely to hash points that are close into the same bucket
- **Point Location in Equal Balls (PLEB):** To solve ANN, first solve an easier problem. With query sentence *q*, need to find a sentence that is within distance r_2 of *q*, but only if there is a sentence within distance r_1.

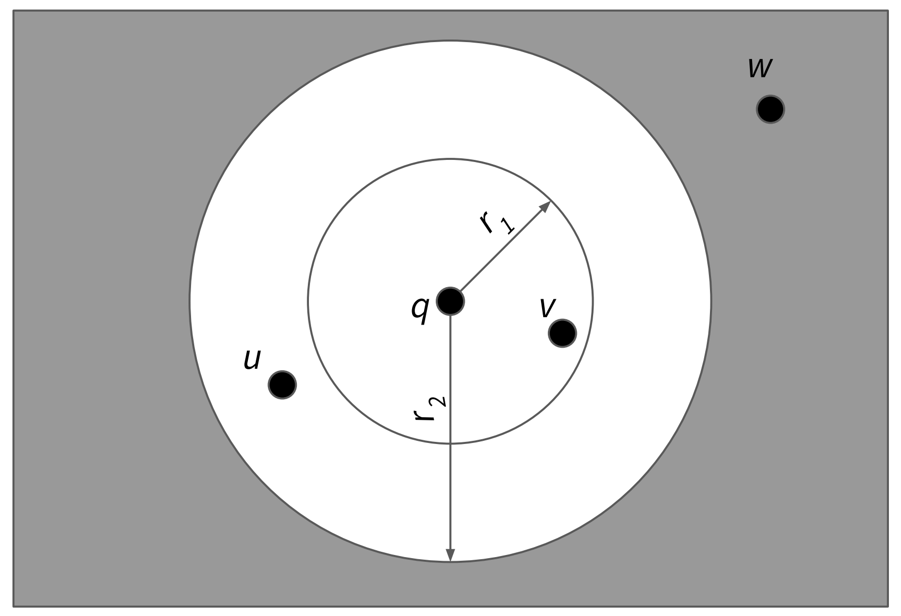


Figure 1: A solution to PLEB can return *u* or *v*, but not *w*. If there was no such *v*, the solution returns nothing. In our problem, these circles represent 768-dimensional balls (hyperspheres).

- To solve PLEB, construct a hash table using LSH functions.
- The family of LSH functions used outputs a 10-bit vector, where each bit is the dot product of the query point with a random Gaussian vector.

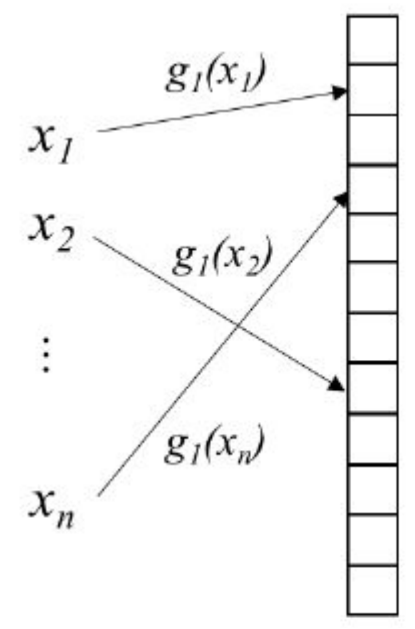


Figure 3: We hash the points in our dataset, x_1 to x_n, into the hash table. For a given query, we hash the query and compare it to points in the same bucket.

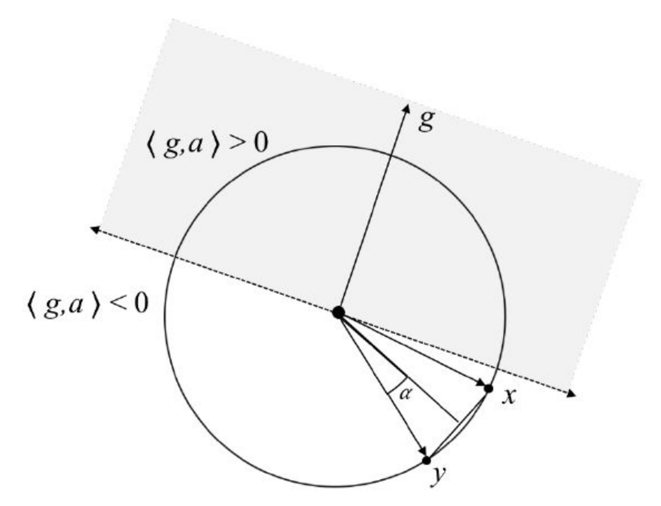


Figure 2: The vector *g* is the random Gaussian vector, and the dotted line is the hyperplane *g* is normal to. Points that are above the dotted line will hash to 1, and points below (e.g. *x* and *y*) will hash to 0.

- Finally, to solve ANN, solve PLEB on different values of r_1 and r_2, and find the smallest values that still returns a solution.

Theoretical Guarantees

- Input parameters
  - *d*: Number of dimensions in sentence embedding
  - *n*: Number of points in dataset
- Hyperparameters
  - *c*: Square of approximation factor
- Algorithm Guarantees
  - Space: $O(dn + n^{1+\frac{1}{c}})$
  - Time: $O(n^{\frac{1}{c}})$
- Linear scan
  - Space: $O(dn)$
  - Time: $O(n)$
- KD-Tree
  - Space: $O(dn \log_2 n)$
  - Time: $O(n^{1-\frac{1}{d}}) \approx O(n)$ (for $d > \log n$)
- **Summary**: Tradeoff between query time vs. accuracy

Search Results

- We compare our LSH algorithm with a random subsetting algorithm, in which we draw a random subset of the data and perform a linear scan to find the nearest neighbor in the subset

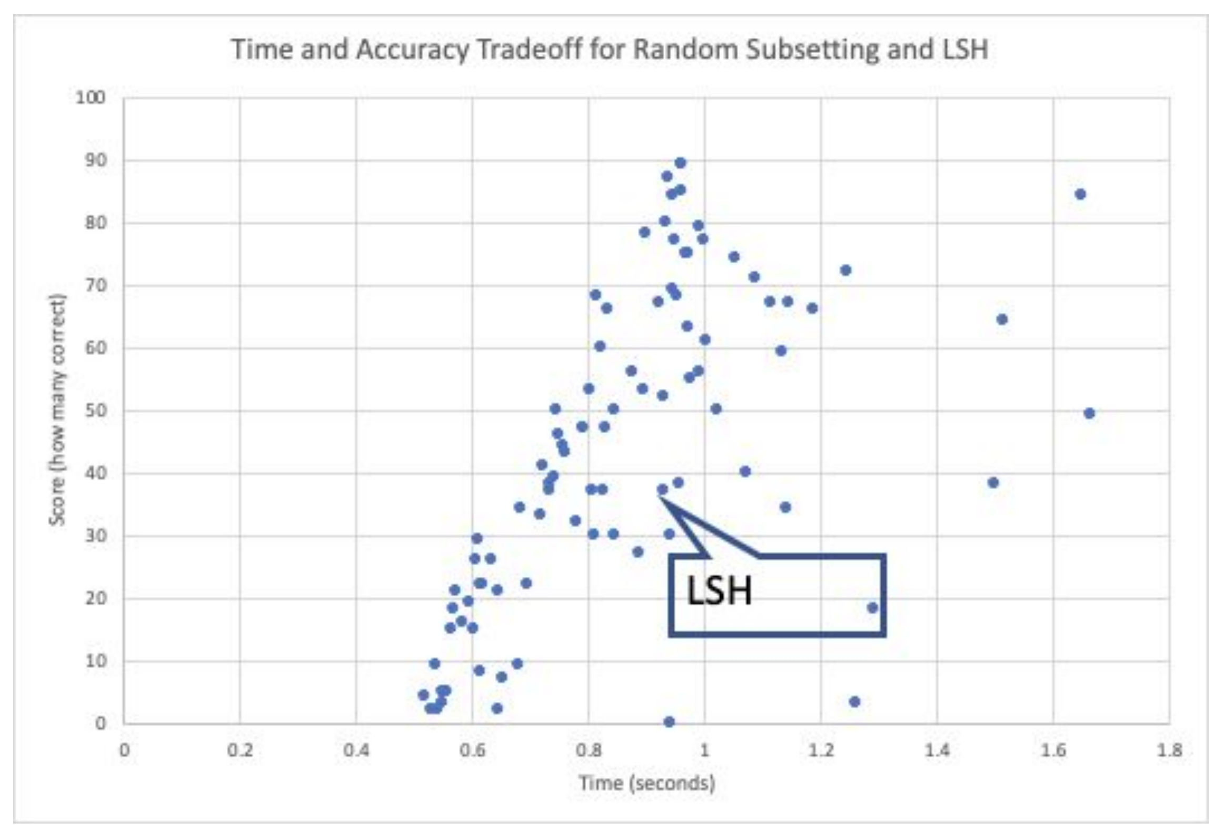


Figure 4: The dataset consists of 178 sentences. The random subsetting algorithm was run with a subset of size ranging from 1% to 80% of the total dataset. The dataset was labelled so the corresponding sentence of a query is known. Score is defined as the number of points that the algorithm was able to identify correctly. LSH performs worse given the same amount of time compared to RBF.

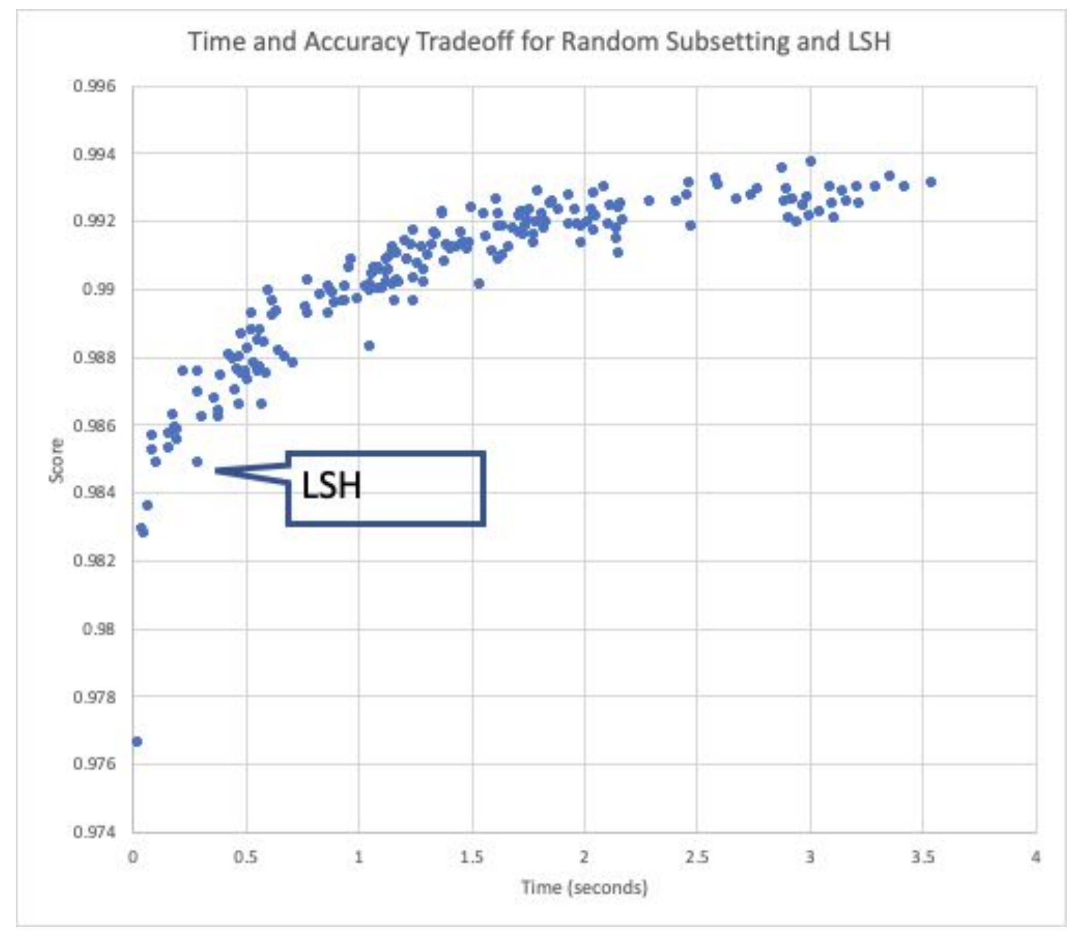


Figure 5: The dataset consists of 70k sentences. The random subsetting algorithm was run with a subset of size ranging from 0.01% to 2% of the total dataset. Score is defined as the reciprocal of the approximate calculated distance divided by the actual minimum distance to the nearest neighbor (actual nearest neighbor is determined from a brute force checking algorithm)

Conclusion

- The fine-tuned BERT model produces 768-dim vectors that give sensible matches on the validation set. PCA suggests reduction to 4-dim may preserve a sentence's semantic representation.
- On smaller datasets, the LSH algorithm performs noticeably worse compared to random subsetting algorithm.
- However, for larger datasets and higher number of queries, our algorithm starts producing comparable results for the accuracy time tradeoff

References

[1] Jay Alammar. The Illustrated BERT, ELMo, and co. https://jalammar.github.io/illustrated-bert/

[2] Moses S Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380-388. ACM, 2002.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[4] Christopher Musco. Lecture 12: Nearest Neighbor Search and Locality Sensitive Hashing. November 2018.