CSC 499 - Honours Thesis

Fall 2025

University of Victoria

Final Report:

Enhancing Relational Databases with Semantic Search Using Word Embeddings

Student: Gerry Peng (V00944715)

Supervisor: Dr Sean Chester

Abstract

Relational databases can be used to store and query structured data, but searching text is mostly limited to exact keyword matching. As a result, it can be difficult to retrieve conceptually related entries when different wording is used. This project explores how word embeddings can be used to represent semantic meaning in text and how these representations can be integrated into a relational database to support a form of fuzzy, meaning based search.

The project began by analyzing word embeddings to examine how semantic structure is captured in embedding spaces. Pretrained vector embeddings are compared with custom embeddings that trained on a domain specific dataset of tweets related to the 2016 American election. Nearest neighbour analysis and vector arithmetic are used to observe how training data size and bias affect the quality and coverage of the resulting embeddings.

Additionally, word embeddings are integrated into a movie database. Movie titles and plot descriptions are represented using word embeddings, and similarity comparisons are performed to retrieve movies based on semantic relevance rather than keyword matches. This is evaluated using a controlled dataset of movies across multiple genres, allowing assessment of how relevant the results are for different query words.

The results show that embeddings trained on more general text corpus tend to have more comprehensive semantic relationships, while embeddings trained on more niche text perform well primarily within their domain, but poorly elsewhere. The database experiments then demonstrated that even simple embedding comparison techniques can create a useful semantic search. Overall, this project shows that word embeddings provide a practical and accessible way to extend traditional database systems with a search that factors in word meaning, while also highlighting the limitations imposed by training data and representation choices.

Table of Contents

1. Introduction

Relational databases are used for storing and querying structured data. However, when applied to data involving text, the queries rely on exact keyword matching. As a result, entries are only returned if they contain the exact words used in the query, even when other entries are more similar conceptually. This causes semantically similar data to not be returned due to wording differences.

A word embedding maps individual words to multi-dimensional vectors which capture the semantic relationships between words. In an embedding space words that appear in similar contexts are represented by vectors that are close together. This allows for the semantic similarity of words to be measured using distance or cosine similarity between the vectors that represent those words. These vector representations give us a new method of comparison beyond string matching. This project demonstrates how word embeddings can be used to support a fuzzy search based around word meaning. An exploratory analysis of word embeddings is conducted to compare large pretrained embeddings with our own embeddings trained on a smaller, more specific dataset. This provides insight about how the training data and hyperparameter selection influences the resulting vectors.

Word embeddings were then implemented in a practical setting. Using an SQL database filled with text data, queries are designed to retrieve results based on semantic similarity (using their associated vectors) rather than word matches like in a typical query. This allows for conceptually relevant text to be identified even when the query words do not appear in the stored text. Using word embeddings can enhance and provide an alternative to standard relational databases. This report explores both the potential and limitations of semantic search and shows how retrieval focused on word meaning can be integrated into existing database systems.

2. Background

2.1 Relational Databases and Text Search

SQL systems are effective for numerical and categorical data, since precise filtering and joins can be used. However, when relational databases are used to store blocks of text such as

descriptions, the querying capabilities are much less practical [1]. Text queries use exact or partial string matching which can be delicate. Semantic meaning is not conveyed between different words, and slight variations of a word can cause unintended consequences. For example, misspelling a word or accidentally using the pluralized version will fail to match with what you intended. This can lead to data being missed when different wording is used to express similar ideas, limiting the practicality of searches using keyword matching for natural language data.

2.2 Word Embeddings and Semantic Representation

Word embeddings provide a numeric representation of words based on their usage in language [2]. Every word is mapped to a vector that is learned from contextual patterns in the training texts. Words that appear in similar contexts tend to have vectors that are close together in each of their dimensions, allowing word similarity to be calculated geometrically [3].

One of the reasons that word embeddings are so powerful is because of how much information they encode. Similarity, association, and relational patterns are all reflected in the structure of the embedding space [3]. This allows us to perform not only vector comparison but also vector arithmetic. A famous embedding known as the Global Vectors for Word Representation (GloVe) was created using unsupervised learning [1]. Using this embedding allows for clear connections to be made between words when their vectors are summed. For example, using each word's vector from the GloVe and calculating:

king + woman - man

results in a vector that is very close to the vector representation for the word queen. This demonstrates how adding and subtracting words can have meaning when using their vector representation.

The quality and usability of these vector representations largely depends on the training data. Models that are trained on large, diverse corpora typically capture broad semantic structure [4]. On the other hand, models trained on smaller or domain specific datasets reflect narrower and more biased relationships. In some cases, certain words may be completely omitted from the embedding space if the word does not appear in the corpora.

2.3 Comparing and Combining Embeddings

To compare embeddings, vector similarity measures were used to quantify how close two vectors are in the embedding space. Cosine similarity is commonly used, since it measures the angle between vectors rather than their magnitude [5]. While an embedding represents individual words, many applications require representing several words together in a sentence or paragraph. A simple and popular approach is to combine the embeddings of all words in a document into a single vector by averaging them all. This produces a single vector that encapsulates the content while remaining computationally efficient. Although this method ignores word order and syntax, it often performs well for short texts.

2.4 Embedding-Based Fuzzy Search in Relational Databases

Using word embeddings for queries inherently creates a fuzzy search since it compares vector representations instead of raw text. This allows for words to be ranked by similarity to one another, whereas plain text can only match or not match the words in a given sentence. Similarity measures are used to retrieve results based on meaning rather than exact term overlap. When applied within a database, this vector search enables semantic retrieval to be used alongside existing SQL infrastructure. This hybrid approach enhances databases with the ability to query by meaning while preserving the structure of the relational model.

3. Methods and Implementation

This project explored word embeddings in an applied semantic search system built on a relational database. The implementation consisted of comparing the pretrained and custom-trained word embeddings, as well as the integration of embeddings into a SQL database to support a fuzzy semantic search.

3.1 Pretrained and Custom Word Embeddings

To examine relationships between words in the English language, pretrained GloVe word embeddings were used as a baseline model. These embeddings were trained on a large and diverse corpus and map each word to a vector in an embedding space. Due to their broad

training data, these embeddings provide stable and general-purpose semantic representations for a wide range of vocabulary.

In addition to the pretrained model, a custom word embedding model was used to understand how training data, scope, and bias affect learned semantic structure. This model was trained using a one gigabyte dataset of tweets from alleged Russian bot accounts during the 2016 American election. The dataset is relatively small and specific, making it well suited for investigating how embeddings behave when trained on biased data.

Prior to training, the tweet text was cleaned to remove URLs, mentions, and other irrelevant content that isn't useful for word extraction. A Word2Vec model was then trained on the processed corpus to obtain new embeddings. Hyperparameters such as vector dimensionality, context window size, and number of training epochs were specified to balance quality and efficiency, although hyperparameter tuning was not the main focus.

```
RANDOM_SEED = 42
VECTOR_SIZE = 100
WINDOW      = 5
MIN_COUNT   = 2
SG          = 0
NEGATIVE    = 5
EPOCHS      = 5
SUBSAMPLE   = 1e-3

w2v = Word2Vec(
   sentences=tokenized_tweets,
   vector_size=VECTOR_SIZE,
   window=WINDOW,
   min_count=MIN_COUNT,
   sg=SG,
   negative=NEGATIVE,
   epochs=EPOCHS,
   workers=4,
   seed=RANDOM_SEED,
   sample=SUBSAMPLE
)
```

3.2 Exploratory Evaluation of Embedding Models

The pretrained and custom-trained embeddings were compared using a variety of short vector arithmetic expressions to determine how closely their outcomes were to one another. The evaluation focused on nearest neighbor queries to assess how well the embeddings trained on tweets mirrored those from the GloVe model.

Nearest neighbor analysis involved retrieving words with the highest cosine similarity to the query word, which made it easier to see how combining words can lead to new meaning. Vector arithmetic was used to see if natural linguistic relationships were maintained in a more qualitative sense. Applying the same evaluation techniques to both models allowed for direct comparison between the pretrained and custom embeddings. This comparison gave insight into the differences in stability, and semantic coverage of the custom embeddings.

3.3 Database Construction and Text Representation

To apply word embeddings in a practical system, a cloud hosted MySQL database was created to store a small list of movies and their associated plot descriptions. Each entry was made up of a movie title, along with its plot summary which was a few sentences long. In order to perform semantic search, each movie was represented as a single vector. This was achieved by tokenizing the words in the movie's title and plot, retrieving all of the corresponding word embeddings from the GloVe model, and computing their average. The resulting vector served as a compact representation of the movie and was stored alongside the relational data.

Other averaging approaches such as putting heavier weighting on title words or eliminating stop words was also explored, but general averaging was used for the analysis and comparison. While this approach doesn't factor in word order and syntactic structure, it provides an efficient way to represent the content.

3.4 Semantic Search Using Cosine Similarity

Since semantic search is performed by comparing the query word's embedding to each of the stored movie vectors, a fast and logical way to compare the distance between two vectors is needed. Cosine similarity was used for this purpose, meaning that the system retrieves each

movie's embedding and computes similarity scores between the query vector and each movie vector in the database. Movies are then ranked according to their similarity scores, and the highest ranked results are returned as the most related movies to the query word. This approach allows movies to be retrieved based on conceptual relevance rather than exact keyword matching. Queries related to a particular genre can return relevant movies even when the query term does not specifically appear in the movie title or plot.

```
def cosine(u, v):
   u, v = np.asarray(u), np.asarray(v)
   nu, nv = norm(u), norm(v)
   return 0.0 if nu == 0 or nv == 0 else float(np.dot(u, v) / (nu * nv))
```

3.5 Movie Dataset and Evaluation

To evaluate the effectiveness of the fuzzy semantic search using word embeddings, the database was populated with a small, controlled set of 30 movies from 3 genres. These consisted of 10 sci-fi and action, 10 romance and drama, and 10 crime and thriller films. These categories were chosen because of how distinct most of the movies are from one another, while still allowing for some logical overlap. This makes them suitable for evaluating whether semantic similarity could capture genre from a single word and find the corresponding movies. For example, querying a word such as “love” would be expected to retrieve movies from the romance and drama category, while queries related to science fiction or crime should favor movies from their respective genres. A complete list of the movies included in the database is shown in Table 1.

Table 1. List of Movies in the Database.

Sci-fi & Action	**Romance & Drama**	**Crime and Thriller**
Inception	Titanic	The Godfather
Interstellar	The Notebook	Goodfellas
The Matrix	La La Land	Pulp Fiction
Star Wars: A New Hope	Pride and Prejudice	The Dark Knight
The Terminator	The Fault in Our Stars	Heat

Blade Runner	Notting Hill	Se7en
Avatar	The Holiday	The Departed
Dune	Crazy Rich Asians	Scarface
The Martian	Me Before You	Casino
Guardians of the Galaxy	Casablanca	John Wick

4. Results

This section presents the results of both the embedding analysis of the custom trained embeddings and the integrated semantic search system. The results are organized to first see the behavior of pretrained and custom-trained word embeddings when performing vector arithmetic, and then evaluate how these embeddings perform when used to support fuzzy search within a relational database.

4.1 Nearest Neighbour of Pretrained and Custom Word Embeddings

Table 2 below shows the top 5 most similar words to “Putin”, “Russia”, and “King” in the GloVe model.

Table 2. Similar Words Ranked Using Nearest Neighbour Analysis with GloVe Embeddings.

Similarity Rank	**Putin**	**Russia**	**King**
#1	yeltsin	ukraine	prince
#2	medvedev	russian	queen
#3	kremlin	moscow	son
#4	kuchma	kremlin	brother
#5	moscow	putin	monarch

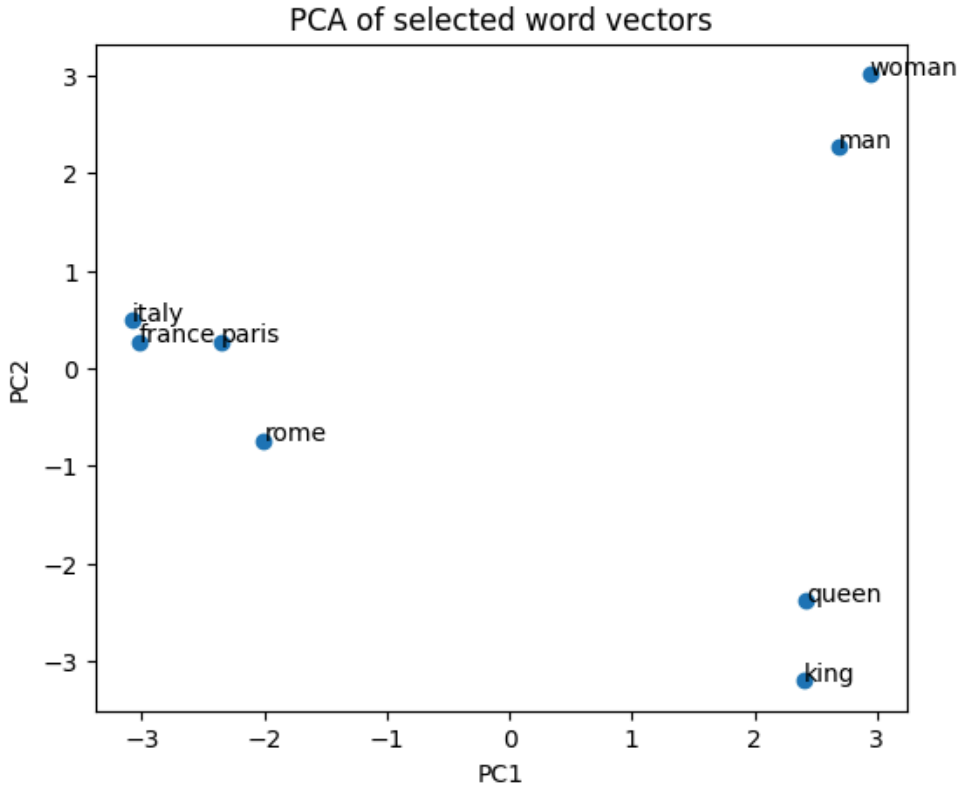
Table 3 shows the results of nearest neighbour analysis using the custom embeddings trained on the Russian tweet dataset.

Table 3. Similar Words Ranked Using Nearest Neighbour Analysis with Custom Embeddings.

Similarity Rank	Putin	Russia	King
#1	russian	putin	dr
#2	kremlin	ukraine	jr
#3	iran	iran	prince
#4	vladimir	cold	miller
#5	moscow	china	stevenson

4.2 Visualization of Embedding Space

To further see the structure of the pretrained embedding space, principal component analysis (PCA) was used to project high-dimensional word vectors into two dimensions.



4.3 Semantic Search Results on Movie Database

Using averaged word embeddings to represent movie titles and plot descriptions enabled effective fuzzy search within the relational database. When a query word was provided,

cosine similarity scores were computed between the query embedding and each movie vector, and movies were ranked accordingly. Tables 4, 5, and 6 outline the results for the query words “space”, “crime”, and “love”.

Table 4. Semantic Search Results for “Space”.

Rank	Movie	Similarity Score
#1	Interstellar	0.732
#2	The Martian	0.717
#3	Avatar	0.673
#4	The Matrix	0.665
#5	Titanic	0.646

Table 5. Semantic Search Results for “Crime”.

Rank	Movie	Similarity Score
#1	Se7en	0.686
#2	Goodfellas	0.661
#3	Scarface	0.665
#4	Pulp Fiction	0.649
#5	The Departed	0.614

Table 6. Semantic Search Results for “Love”.

Rank	Movie	Similarity Score
#1	Pride and Prejudice	0.773
#2	La La Land	0.757
#3	Me Before You	0.744
#4	Crazy Rich Asians	0.705
#5	A Star is Born	0.703

5. Discussion

As shown in Table 2, the pretrained GloVe embeddings had its nearest words for each query heavily related. Each word nearest to “Putin” and ”Russia” has a clear geopolitical connection. Furthermore, the words nearest to “King” are all clearly related and relevant in context.

In Table 3, the most similar words to “Putin” and “Russia” are also quite relevant and similar. However, the words most similar to “King” seem unrelated and have little obvious connection. This suggests that the words “Putin” and “Russia” appeared many different times in the dataset of tweets, which made their meaning and context more clear. However, a word like “king” that is not particularly related to Russia in a geopolitical sense likely did not appear often in the dataset, making it more difficult for the model to accurately represent it. This demonstrates how the more biased dataset of tweets can still perform well for words related to the topic of the training data, but perform much worse for other unrelated words.

The resulting PCA visualization revealed clear clustering patterns among semantically related words. The PCA for some words in the GloVe model, shows how words that represent cities tend to cluster together, while gendered terms and royalty titles are closely related semantically as expected.

The rankings for the query words “space”, “crime”, and “love” are shown in Table 4, Table 5, and Table 6 respectively. These tables show that movies from the expected genres generally appear near the top of the similarity rankings. For the query “space”, four of the top five results are science fiction films, while the “crime” query returns crime and thriller movies such as Se7en, Goodfellas, and Scarface. Similarly, the “love” query produces results dominated by romance and drama films, including Pride and Prejudice, La La Land, and Me Before You.

While some results do not strictly belong to the expected genre, the similarity scores indicate that these movies still share thematic overlap with the query term. For example, Titanic appears in the results for “space” which does not seem as relevant as some of the other available movies. Overall, the results demonstrate that averaging word embeddings produces movie representations for genre information, allowing related movies to be retrieved even when the query word does not appear in the movie title or plot.

6. Conclusion

This project explored how word embeddings can be used to analyze semantic structure in language and to create a form of fuzzy search within a relational database. By comparing pretrained and custom trained embeddings, the project demonstrated how training data size and domain impact the quality of the semantic relationships between words. The analysis showed that embeddings trained on large corpora allow for more consistent semantic patterns, as shown by the relevance of the nearest neighbour results for a variety of query words in Table 2, while embeddings trained on specialized data reflect more specific relationships, such as in Table 3 where only specific terms yielded relevant results.

By integrating word embeddings into a movie database, the results showed that vector representations allow for a type of semantic search that isn't feasible with keyword matching queries. Using averaged embeddings and cosine similarity, the system was able to retrieve conceptually relevant results without needing complex models. Overall, the project demonstrates that using word embeddings as part of a query is a practical and accessible way to extend relational databases with meaning-aware search capabilities.

7. Future Work and Limitations

Although this project shows that word embeddings can support semantic search in a relational database, there are some limitations to the current approach. One limitation is how the vector representations for sentences are calculated. Averaging the word embeddings is simple, but it ignores word order, syntax, and the relative importance of individual terms. Another limitation is that the quality of the search depends heavily on the embeddings used and how they were trained. Word embeddings trained on general purpose corpora may not perform well when applied to more niche text, where some important words may be underrepresented. As a result, the usefulness of embeddings can vary depending on how closely the training data aligns with the verbiage of the text being queried.

Future work could explore several ways to address these limitations including incorporating weighted averaging or even sentence embeddings. Supporting queries that involve multiple words and basic query expansion would also make the search system more useful. Finally,

applying this approach to larger datasets would allow for evaluation of scalability and performance beyond these small samples.

8. References

[1] J. Barnard, *What Are Word Embeddings?*, IBM Think. Available: https://www.ibm.com/think/topics/word-embeddings

[2] *A Guide on Word Embeddings in NLP*, Turing.com, Feb. 10, 2022. Available: https://www.turing.com/kb/guide-on-word-embeddings-in-nlp

[3] M. J. Hussein, "Word Embeddings in NLP," *ResearchGate*, Sep. 2025. Available: https://www.researchgate.net/publication/395919901_Word_Embeddings_in_NLP

[4] A. Mandelbaum and A. Shalev, "Word Embeddings and Their Use in Sentence Classification Tasks," *arXiv preprint arXiv:1610.08229*, Oct. 2016.

[5] N. Birajdar, "Word2Vec Research Paper Explained," *Towards Data Science*, Mar. 2021. Available: https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30