ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Enhancing Chatbot Response Relevance through Semantic Similarity Measures

Naveen Edapurath Vijayan

Data Science Manager, Amazon Seattle, WA 98765, USA

ABSTRACT

Semantic similarity measures have shown promise in enhancing natural language understanding by quantifying the likeness between textual elements. This paper investigates the application of semantic similarity measures to improve chatbot response relevance. By leveraging word embeddings and similarity metrics, this study aims to bridge the gap between simple keyword-based responses and contextually rich, relevant answers. The proposed approach integrates both traditional lexical measures and advanced vector-based embeddings to enhance user intent interpretation and ensure a more suitable response generation. By refining the candidate response set with these techniques, the chatbot is able to generate highly relevant and contextually accurate replies. The results demonstrate that combining deep learning-based embeddings with traditional semantic metrics can significantly enhance response relevance, leading to more meaningful and human-like user interactions. Additionally, a feedback loop is incorporated to continuously learn from user interactions, further improving the quality of responses over time.

***Corresponding author**

Naveen Edapurath Vijayan, Data Science Manager, Amazon Seattle, WA 98765, USA.

Received: January 13, 2022; **Accepted:** January 17, 2022; **Published:** January 24, 2022

Keywords: Chatbot, Semantic Similarity, NLP, BERT, Elasticsearch

Introduction

Chatbots have become an essential tool for customer service, personal assistance, and information retrieval, playing a crucial role in enhancing user experiences and improving operational efficiency [1]. Their success relies heavily on the ability to provide accurate, contextually appropriate responses, which directly impacts user satisfaction and overall effectiveness. Traditional methods for generating responses often rely on rule-based systems or keyword matching, which frequently result in irrelevant or unsatisfactory answers, particularly in complex or nuanced interactions. Rule-based systems, while easy to implement, lack the flexibility needed to handle diverse user queries, and keyword-based approaches fail to capture the deeper intent behind user inputs, leading to an overall poor conversational experience.

To address these limitations, semantic similarity measures have emerged as a promising solution. By quantifying the similarity between a user's query and potential responses, chatbots can provide more relevant and contextually accurate replies. These measures enable chatbots to better understand user intent, leading to more natural and human-like interactions. The use of advanced NLP techniques such as word embeddings and deep learning models allows chatbots to move beyond simple keyword matching and towards a more meaningful understanding of language. This paper aims to address the current challenges of chatbots and proposes a methodology for enhancing response relevance using semantic similarity measures.

Semantic Similarity in Chatbots

Semantic similarity refers to the degree of relatedness between two pieces of text based on their meaning rather than their syntactic form. In the proposed approach, the chatbot leverages Elasticsearch to efficiently search through the current dataset and retrieve the most relevant response based on semantic similarity measures. This integration allows the chatbot to handle large and diverse datasets, ensuring that the response is contextually appropriate and aligns with the user's intent. By focusing on meaning rather than exact word matches, the chatbot improves the quality of conversations, enabling more accurate and contextually relevant replies. For example, if a user asks "How can I fix my login issue?" a keyword-based chatbot might focus on the word "login" without understanding the context of "fix," whereas a semantic similarity approach would better capture the user's intent to troubleshoot a problem.

Various approaches to measure semantic similarity can be grouped into three main categories: lexical similarity, vector-based embeddings, and hybrid techniques. Lexical similarity measures such as TF-IDF and Jaccard similarity are useful for identifying common words or phrases. Vector-based embeddings, including Word2Vec, GloVe, and BERT, are more sophisticated and capture the deeper meaning of text by converting it into high-dimensional vectors. Hybrid techniques combine the strengths of lexical and vector-based approaches, using the former for initial filtering and the latter for deeper semantic matching. This ensures that responses are both efficient and contextually accurate, providing a balanced solution for effective chatbot communication.

Techniques for Measuring Semantic Similarity

Lexical Similarity

Lexical similarity methods rely on comparing words in a traditional sense, such as through the use of Jaccard similarity or cosine similarity applied to Term Frequency-Inverse Document

Frequency (TF-IDF) vectors. These methods work well for identifying overlapping words or phrases but are limited in capturing deeper contextual meanings. For example, they may fail to understand that "purchase" and "buy" represent the same action, often resulting in less effective responses. Lexical similarity is useful for quickly finding candidate responses that contain similar words or phrases to the user's input. However, these methods cannot differentiate between different meanings of the same word or recognize paraphrased content.

To address these limitations, lexical similarity is often combined with other advanced techniques to improve overall response relevance. Despite their limitations, lexical similarity measures are computationally efficient, making them suitable for initial filtering stages in chatbot response generation, where a quick comparison can help reduce the number of potential responses before applying more computationally intensive techniques.

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

where A and B are the TF-IDF vectors of two texts.

Vector-Based Embeddings

Recent advancements in natural language processing (NLP) have popularized vector-based techniques, such as Word2Vec, GloVe, and BERT. These techniques are integrated with Elasticsearch to allow the chatbot to efficiently search through stored data and generate relevant responses. These approaches embed words, sentences, or entire documents into high-dimensional vectors that capture semantic nuances, enabling a deeper understanding of the context and intent behind the user's input.

Word2Vec and GloVe are early embedding techniques that create dense vector representations of words by learning from large corpora. Word2Vec captures semantic relationships through continuous bag-of-words and skip-gram models, while GloVe uses co-occurrence statistics to learn word vectors. These models, though effective, often struggle with capturing the contextual meaning of words in different situations.

BERT (Bidirectional Encoder Representations from Transformers), introduced in 2019, takes embeddings a step further by providing contextualized representations. Unlike Word2Vec and GloVe, BERT considers the entire sentence structure, taking into account both the left and right context of a word, which allows it to capture the nuances of meaning that depend on the context. This makes BERT particularly effective for chatbots, as it can understand the intent behind user queries more accurately.

In the chatbot architecture, utilizing these embeddings allows the model to identify the best response by determining which response vector is closest to the query vector. By integrating BERT with Elasticsearch, the chatbot can not only efficiently narrow down the set of potential responses but also rank them based on their contextual similarity to the user's query. This results in more accurate, relevant, and contextually aware responses, ultimately enhancing the overall conversational experience for users.

Hybrid Techniques

Hybrid approaches combine lexical and vector-based methods to balance precision and computational efficiency. In the proposed approach, Elasticsearch plays a key role in narrowing down the set of possible responses by utilizing lexical similarity techniques like TF-IDF or Jaccard similarity to perform an initial filtering. This allows the chatbot to quickly eliminate responses that are not semantically related to the user's query, making the process more efficient.

After this initial filtering, transformer-based embeddings like BERT are employed to refine the candidate responses further. BERT's contextual understanding is used to rank the filtered responses by determining the degree of semantic match between the query and the candidate responses. This two-step approach ensures that the computationally expensive deep learning models are only applied to a narrowed-down set of candidates, significantly improving both response quality and efficiency.

For example, when a user submits a query, Elasticsearch first performs a fast and computationally inexpensive lexical search to generate a candidate set of responses that have a high likelihood of containing relevant information. The BERT model then processes this candidate set to determine which response best captures the user's intent, considering the context and subtleties of the language. By combining the speed of lexical methods with the depth of transformer-based embeddings, hybrid techniques provide a powerful way to achieve both accuracy and efficiency in chatbot responses. This approach also helps in maintaining scalability, as the initial filtering step can handle large datasets, allowing only the most promising candidates to be processed by more resource-intensive models.

Proposed Methodology

This section provides a detailed methodology for enhancing chatbot response relevance using semantic similarity measures. The proposed approach integrates traditional lexical similarity metrics with advanced transformer-based embeddings, utilizing attention mechanisms to generate contextually accurate responses. The chatbot architecture leverages Amazon Lex for natural language understanding and conversation management, and Elasticsearch for efficient indexing and retrieval of potential responses. The Methodology Comprises Several Key Components:

- System Architecture Overview
- Data Preparation and Indexing
- Initial Candidate Selection Using Lexical Similarity
- Refinement Using Transformer-Based Embeddings
- Incorporation of Attention Mechanisms
- Feedback Loop Implementation

System Architecture Overview

The Proposed Chatbot System Consists of the Following Components

- **User Interface (UI):** The platform where users interact with the chatbot (e.g., website, mobile app, messaging service).
- **Amazon Lex:** Provides natural language understanding (NLU), handling intent recognition and entity extraction.
- **AWS Lambda Functions:** Serve as intermediaries, orchestrating the flow between Amazon Lex, Elasticsearch, and the Semantic Similarity Module.
- **Elasticsearch Cluster:** Stores and indexes potential responses and associated metadata, enabling efficient search and retrieval.
- **Semantic Similarity Module:** Calculates similarity scores using both lexical and vector-based methods, incorporating attention mechanisms.
- **Feedback Mechanism:** Collects user feedback to refine and improve response relevance over time.
- **Data Storage:** Maintains logs, user interactions, and updated model parameters for continuous learning.

Data Preparation and Indexing

Dataset Compilation

A comprehensive dataset is compiled, including.

- **Frequently Asked Questions (FAQs):** Common user queries and standard responses.
- **Transactional Records:** Historical interaction data, capturing diverse user intents.
- **Knowledge Base Articles:** Detailed documents providing in-depth information.

Data Preprocessing

Data Preprocessing Ensures Consistency and Optimal Performance During Similarity Calculations.

- **Tokenization:** Splitting text into individual words or tokens.
- **Normalization:** Converting text to lowercase, removing punctuation and special characters.
- **Stop Word Removal:** Eliminating common words that do not contribute to semantic meaning (e.g., "the," "is," "at").
- **Stemming/Lemmatization:** Reducing words to their root forms (e.g., "running" → "run").

Table 1: Data Preprocessing Example

Original Text	Processed Text
"How can I reset my password if I've forgotten it?"	"Reset password forgotten"
"What's the status of my recent order #12345?"	"Status recent order 12345"
"I need help with setting up my new device, please."	"Need help setting new device"

Indexing with Elasticsearch

After Preprocessing, the Data is Indexed in Elasticsearch for Efficient Retrieval.

- **Term Frequency-Inverse Document Frequency (TF-IDF) Vector Generation Term Frequency (TF):**

$$\mathrm{TF}_{t,d} = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

Inverse Document Frequency (IDF):

$$\mathrm{IDF}_t = \log\left(\frac{N}{n_t}\right)$$

TF-IDF Weight

$$\text{TF-IDF}_{t,d} = \mathrm{TF}_{t,d} \times \mathrm{IDF}_t$$

- **Metadata Tagging:** Responses are tagged with relevant metadata (e.g., categories, intents) to enhance filtering.
- **Embedding Storage:** Precomputed embeddings (e.g., BERT embeddings) for each response are stored for semantic similarity calculations.

Initial Candidate Selection Using Lexical Similarity

- **Query Processing**

User Queries Undergo the same Preprocessing Steps as the Dataset:

- o **User Query:** "I can't access my account; how do I reset my password?"
- o **Processed Query:** "can't access account reset password"

- **Lexical Similarity Calculation**

Lexical similarity measures are computed between the processed query and indexed documents.

- o **Cosine Similarity with TF-IDF Vectors**

Cosine Similarity Formula:

$$\cos(\theta) = \frac{A \cdot B}{|A| \times |B|}$$

Where:

- o A = TF-IDF vector of the query.
- o B = TF-IDF vector of a document.

Table 2: TF-IDF Vectors and Cosine Similarity Calculation

Term	TF-IDF (Query)	TF-IDF (Doc1)	TF-IDF (Doc2)
can't	0.3	0	0.2
access	0.4	0.1	0.3
account	0.5	0.4	0.5
reset	0.6	0.5	0.6
password	0.7	0.6	0.7

- **Norms:**

$$|A| = \sqrt{0.3^2 + 0.4^2 + 0.5^2 + 0.6^2 + 0.7^2} = \sqrt{1.35} \approx 1.162$$
$$|B| = \sqrt{0.0^2 + 0.1^2 + 0.4^2 + 0.5^2 + 0.6^2} = \sqrt{0.78} \approx 0.883$$

- **Cosine Similarity (Doc1):**

$$\cos(\theta) = \frac{0.96}{1.162 \times 0.883} = \frac{0.96}{1.025} \approx 0.936$$

Table 3: Lexical Similarity Scores

Document ID	Cosine Similarity	Selected (Yes/No)
Doc1	0.936	Yes
Doc2	0.982	Yes
Doc3	0.65	No
Doc4	0.4	No

Responses with cosine similarity above a predefined threshold (e.g., 0.7) are selected as candidates.

- Query Embedding (Q): Q="BERT" ("Processed Query")
- Response Embeddings (R_i):R_i="BERT" ("Candidate " "Response$_i$)

Incorporation of Attention Mechanisms

The attention mechanism allows the model to focus on relevant parts of the input.

Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where:

- o Q = Query matrix.
- o K = Key matrix (from responses).
- o V = Value matrix (from responses).
- o dk = Dimensionality of the key vectors.

Calculation of Semantic Similarity Scores

- Compute Attention Weights (α_i):

$$\alpha_i = \text{softmax}\left(\frac{QK_i^T}{\sqrt{d_k}}\right)$$

- Compute Context Vector (Ci):

$$C_i = \sum_j \alpha_{ij} V_{ij}$$

- Calculate Cosine Similarity between Query and Context Vectors:

$$\text{Similarity Score}_i = \frac{Q \cdot C_i}{|Q| \times |C_i|}$$

Table 4: Semantic Similarity Scores

Candidate Response ID	Semantic Similarity Score	Rank
Resp2	0.99	1
Resp1	0.97	2

Response Selection

The candidate response with the highest semantic similarity score is selected.

Incorporation of Attention Mechanisms

Attention mechanisms enhance the model's ability to capture context by assigning different weights to different words.

- Example: In the query "can't access account reset password," the words "access," "account," "reset," and "password" are more significant.
- Attention Weight Calculation:

$$\alpha_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)}$$

Attention Weights Example

Word in Query	Key Vector(K_i)	Score (e_i)	Attention Weight (α_i)
can't	[0.1, 0.2, ...]	0.5	0.1
access	[0.3, 0.4, ...]	1.2	0.25
account	[0.2, 0.3, ...]	1	0.2
reset	[0.4, 0.5, ...]	1.5	0.3
password	[0.5, 0.6, ...]	1.8	0.35

Note: The Vectors and Scores are Illustrative.

Feedback Loop Implementation

User Feedback Collection

- **Explicit Feedback:** Users rate responses (e.g., thumbs up/down).
- **Implicit Feedback:** User behavior such as follow-up questions or session duration.

Model Refinement

- Adjusting Weights:

$$w_{\text{new}} = w_{\text{old}} + \eta(\text{Feedback Score} - \text{Expected Score})$$

Where:

- o w = Weight assigned to similarity measures.
- o η = Learning rate.
- **Updating Thresholds:** Based on feedback, thresholds for candidate selection can be adjusted.
- **Model Retraining:** Incorporate new interaction data into model training for continuous improvement.

Conclusion

Semantic similarity measures play a pivotal role in enhancing chatbot response relevance by enabling a deeper understanding of user intents beyond mere keyword matching. This paper presented a comprehensive methodology that integrates both lexical and advanced vector-based techniques, such as BERT embeddings, within an efficient retrieval framework powered by Elasticsearch. The incorporation of attention mechanisms and a continuous feedback loop further refines the chatbot's ability to generate contextually accurate and meaningful responses.

By successfully combining these approaches, the proposed system addresses the limitations of traditional methods, offering significant improvements in response relevance and user satisfaction. Future work will focus on leveraging the latest advancements in transformer architectures and exploring real-time adaptation through reinforcement learning. The ongoing enhancement of semantic understanding in chatbots promises to deliver increasingly natural and human-like interactions, benefiting users and organizations alike [2-16].

References

1. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.
2. Devlin J, Chang M W, Lee K, Toutanova K (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT 4171-4186.
3. Pennington J, Socher R, Manning C (2014) GloVe: Global Vectors for Word Representation. EMNLP 1532-1543.
4. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, et al. (2017) Attention is All You Need. NeurIPS https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
5. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, et al. (2019) XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv https://arxiv.org/abs/1906.08237.
6. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving Language Understanding by Generative Pre-Training. OpenAI https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
7. Cer D, Yang Y, Kong SY, Hua N, Limtiaco N, et al. (2018) Universal Sentence Encoder. arXiv preprint arXiv:1803.11175.
8. Reimers N, Gurevych I (2019) Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP https://aclanthology.org/D19-1410/.
9. Karpukhin V, Oguz B, Min S, Lewis P, Wu L, et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. EMNLP https://aclanthology.org/2020.emnlp-main.550/.
10. Clark K, Khandelwal U, Levy O, Manning CD (2019) What Does BERT Look At? An Analysis of BERT's Attention. ACL https://aclanthology.org/W19-4828/.
11. Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, et al. (2020) ALBERT: A Lite BERT for Self-supervised Learning

of Language Representations. ICLR https://arxiv.org/abs/1909.11942.

12. Howard J, Ruder S (2018) Universal Language Model Fine-tuning for Text Classification. ACL https://aclanthology.org/P18-1031/.
13. Liu Y, Ott M, Goyal N, Du J, Joshi M, et al. (2019) RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.
14. Zhang Y, Yang Q (2021) A Survey on Multi-Task Learning. IEEE Transactions on Knowledge and Data Engineering.
15. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT https://ieeexplore.ieee.org/document/9392366.
16. Pennington J, Socher R, Manning C (2014) GloVe: Global Vectors for Word Representation. EMNLP https://aclanthology.org/D14-1162/.