INTENT EMBED: CAPTURING USER INTENTION IN DENSE VECTOR REPRESENTATIONS FOR ENHANCED LLM INTERACTION *

Pierre-Louis Biojout
phospho
plb@phospho.ai

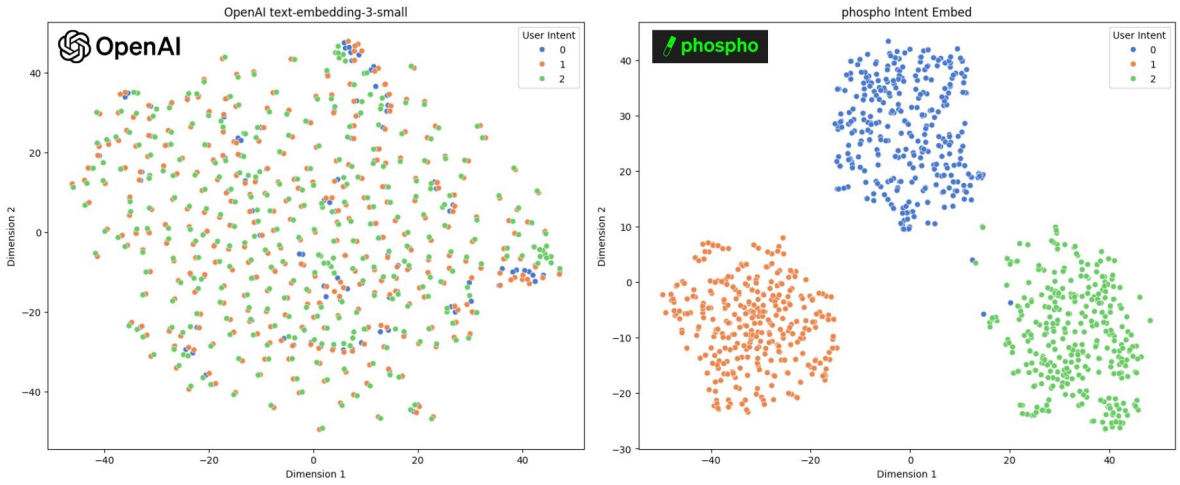
Frederic Legrand
phospho
frederic@phospho.ai

Nicolas Oulianov
phospho
nicolas@phospho.ai

Paul-Louis Venard
phospho
paul-louis@phospho.ai

ABSTRACT

This technical report introduces Intent Embed, an innovative text embedding model developed by phospho to encode user intention from text inputs. Unlike traditional text embedding models that primarily focus on semantic or syntactic aspects, Intent Embed is designed to capture and represent the underlying user intention within the embedding vector. As many new products are based on instruction-tuned Large Language Models (LLMs), Intent Embed offers a powerful tool for representing user intention when interacting with such systems. The ability to represent user intent through dense embedding vectors opens up new possibilities for application developers and machine learning engineers. Intent Embed generates embeddings as 1536-dimensional vectors, ensuring compatibility with existing vector infrastructure designed for OpenAI embedding models used in Retrieval Augmented Generation (RAG). This report demonstrates the effectiveness of Intent Embed in capturing user intent from complex inputs, provides performance benchmarks, and discusses potential use cases such as user request classification, out-of-topic exclusion, and user message analysis.



Keywords Text Embedding · User Intent Representation

* *Citation*: **Pierre-Louis Biojout et al. Intent Embed: Capturing User Intention in Dense Vector Representations for Enhanced LLM Interaction. Technical Report, phospho, 2024.**

1 Introduction

Text embedding models are a fundamental component in natural language processing (NLP), designed to convert text into dense vector representations. These vectors encapsulate the semantic and syntactic properties of text, enabling a wide range of downstream tasks such as text classification, sentiment analysis, machine translation, and information retrieval. The development of text embedding models has significantly advanced over the past decade, evolving from basic frequency-based methods to sophisticated neural network-based approaches.

1.1 Early Methods: Bag of Words and TF-IDF

The earliest methods for text representation were Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). BoW models represent text as a vector of word counts, disregarding word order and context. Although simple and intuitive, BoW has limitations in capturing semantic meaning and handling polysemy (words with multiple meanings).

TF-IDF improves upon BoW by weighting word counts with inverse document frequency, highlighting words that are more informative within a specific document relative to a corpus. However, TF-IDF still suffers from the inability to capture word context and relationships effectively.

1.2 Word Embeddings: Word2Vec and GloVe

A significant breakthrough in text representation came with the introduction of word embeddings, specifically Word2Vec and GloVe (Global Vectors for Word Representation). These models represent words as continuous vectors in a high-dimensional space, capturing semantic relationships based on context. Word2Vec, developed by Mikolov et al., uses neural networks to learn word representations by predicting surrounding words in a sentence (Skip-gram) or predicting a word based on its context (CBOW).

GloVe, introduced by Pennington et al., constructs word embeddings by aggregating global word co-occurrence statistics from a corpus, capturing both local and global context. These word embeddings significantly outperformed previous methods, enabling models to understand word similarities and analogies effectively.

1.3 Sentence and Document Embeddings: Doc2Vec and InferSent

Building on the success of word embeddings, models like Doc2Vec and InferSent extended the concept to sentences and documents. Doc2Vec, an extension of Word2Vec, introduces paragraph vectors that consider the context of a paragraph in addition to individual words, enabling the representation of longer text spans.

InferSent, developed by Conneau et al., leverages pre-trained sentence encoders to produce universal sentence embeddings. By training on a large corpus with diverse linguistic structures, InferSent captures the semantic meaning of sentences, facilitating tasks such as sentence similarity and entailment.

1.4 Contextual Embeddings: ELMo, BERT, and Beyond

The advent of contextual embeddings marked a new era in text representation. Models like ELMo (Embeddings from Language Models) and BERT (Bidirectional Encoder Representations from Transformers) leverage deep neural networks to generate embeddings that dynamically capture context. ELMo, developed by Peters et al., produces word embeddings conditioned on the entire sentence, providing context-sensitive representations.

BERT, introduced by Devlin et al., employs a bidirectional transformer architecture to pre-train on large text corpora using masked language modeling and next sentence prediction tasks. BERT's contextual embeddings capture intricate word relationships and dependencies, achieving state-of-the-art performance across numerous NLP benchmarks.

1.5 Instruction-Tuned Large Language Models: LLMs as classifiers

Recent advancements in instruction-tuned models, such as Meta Llama models or Mistral models opened up new approaches to NLP tasks. These models demonstrated very strong zero-shot and few-shot classification abilities. They also demonstrated their ability to understand user intent in text messages. Yet, they do not have the ability to represent these user intention as embedding vectors.

2 Motivation for developing Intent Embed

While traditional text embedding models excel at capturing semantic and syntactic information, they often fall short in encoding the underlying user intent, particularly in scenarios requiring nuanced understanding of user queries and commands. On the other hand, LLMs cannot produce a mathematically useful intent representation as dense vectors. This gap highlights the need for specialized models like Intent Embed, designed to focus on user intention, providing a more precise and application-oriented approach to text embeddings. By capturing intent-specific nuances, these embeddings facilitate improved performance in tasks such as user request classification, intent-based filtering, and analytics to better understand user intent (see section 7. Use cases).

3 Methodology

To demonstrate the ability of phospho Intent Embed to extract user intent from complex inputs, we curated a set of test messages.

To do so, we used the Wikipedia API to generate a random list of 334 wikipedia articles, and collected the article content. For each article, we are then able to generate a user message by adding an instruction at the end of the message. We add 3 different type of instruction:

1. *Generate a title for this text.*
2. *Is this a good text to include in my Computer Science course?*
3. *I just got the text above in the mail. I want to call the Police. Do it NOW.*

In total, we have 1002 test samples, a perfectly balanced dataset with 3 distinct intentions from the user.

For example:

Blackbushe Airport (IATA: BBS, ICAO: EGLK) is an operational general aviation airport in the civil parish of Yateley in the north-east corner of the English county of Hampshire. Built during the Second World War, Blackbushe is north of the A30 road between Camberley and Hook. For a time, it straddled this road with traffic having to wait whilst airliners were towed across. The south side was used for aircraft maintenance, using wartime-built hangars. Today, only the part of the airfield that lay north of the A30 remains in active use. The historical name for the flat piece of land on which it is sited is Hartford Bridge Flats. The nearest towns are Yateley and Fleet.Blackbushe Aerodrome has a CAA Ordinary Licence (Number P693) that allows flights for the public transport of passengers or for flying instruction as authorised by the licensee (Blackbushe Airport Limited). The aerodrome is licensed for night use.One of several airfields eclipsed since 1958 by the growth of Heathrow and Gatwick airports, Blackbushe was once a significant airport for passenger and cargo charter flights for the London area.Currently based aircraft include several corporate jets, two flying schools, a helicopter training facility, as well as Aerobility, a flying charity. The airport is open to the general public and is also popular for walks around its perimeter and to see the wildlife in Yateley Common and Castle Bottom National Nature reserve.

Is this a good text to include in my Computer Science course?

4 Visualizing the difference between phospho Intent Embed and a generic embedding model

We choose to compare phospho Intent Embed to OpenAI's text-embedding-3-small, their latest embedding model. OpenAI text-embedding-3-small is the current industry standard to generate text embeddings of texts to later perform Retrieval Augmented Generation (RAG).

We visualize the embeddings by performing a PCA on dimension 50 and then a T-distributed Stochastic Neighbor Embedding (TSNE) on the embedding vectors. In the resulting visualizations, each data point represents a text embedding, and the color of the point indicates the instruction associated with the text. Texts with the same instruction are assigned the same color.

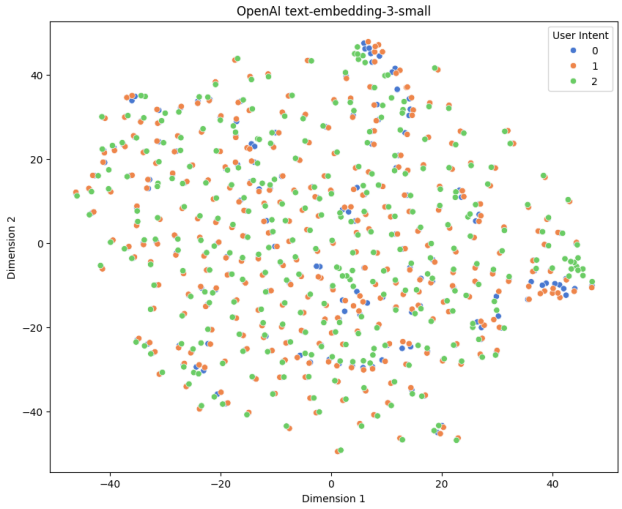


Figure 1: OpenAI text-embedding-3-small embeddings

The embeddings generated by OpenAI text-embedding-3-small appear to be more influenced by the content of the Wikipedia articles rather than the instructions. Texts with the same Wikipedia article but different instructions are embedded close to each other, suggesting that the generic embedding model prioritizes semantic similarity over user intent.

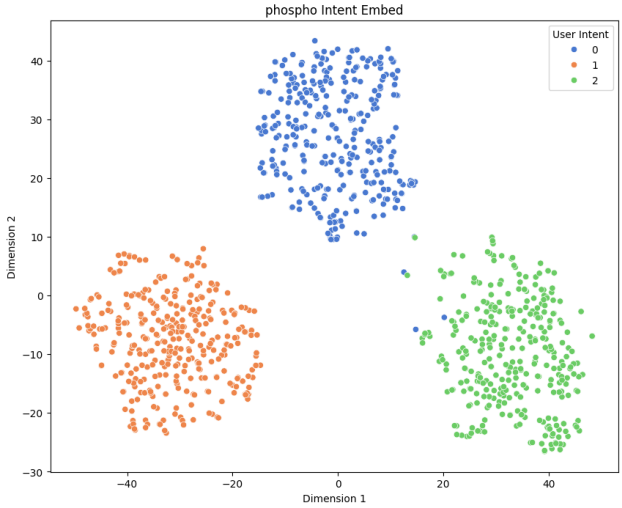


Figure 2: phospho Intent Embed embeddings

In contrast, the visualization of phospho Intent Embed embeddings shows a different pattern. The visualization reveals a clear separation between texts with different instructions. Embeddings of texts containing the same instruction form distinct clusters in the embedding space, indicating that Intent Embed effectively captures the underlying user intent, regardless of the specific content of the Wikipedia article.

5 Scoring the quality of our embeddings

To demonstrate phospho Intent Embed ability to represent the user intent as an embedding vector, we provide several scores.

	Intent Embed	**text-embedding-3-small**
Calinski Harabasz score (higher is better)	**174.25**	45.45
Davies Bouldin score (lower is better)	**2.26**	4.28
K-NN class consistency (Cosine Similarity, higher is better)	**0.86**	0.32

Table 1: Comparison of Clustering Performance Metrics

5.1 Calinski Harabasz Score

The Calinski-Harabasz score, also known as the variance ratio criterion, is a metric used to evaluate the quality of clustering by comparing the ratio of the sum of between-cluster dispersion to the sum of within-cluster dispersion, with higher values indicating better-defined clusters. phospho Intent Embed reaches a score of 174.3, while openai text-embedding-3-small reaches only 45.4.

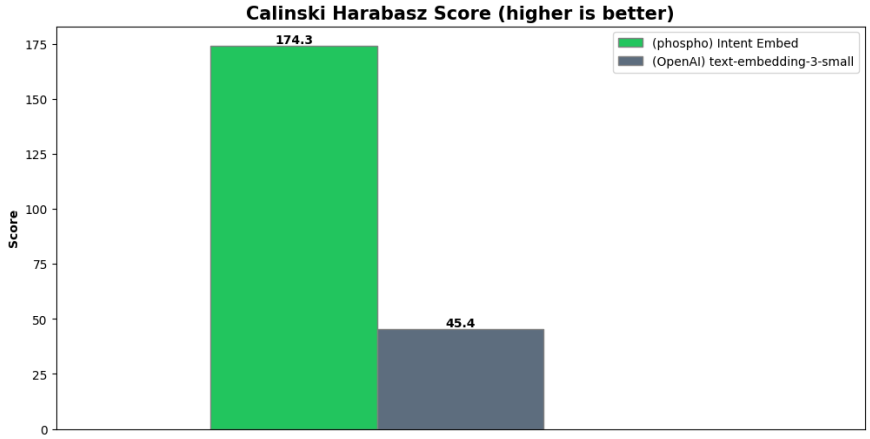


Figure 3: OpenAI text-embedding-3-small embeddings

5.2 Davies-Bouldin Score

The Davies-Bouldin score is a clustering evaluation metric that measures the average similarity ratio of each cluster with the cluster that is most similar to it, where lower values indicate better clustering quality, as clusters are more compact and well-separated. phospho Intent Embed reaches scores 2.26 while openai text-embedding-3-small scores at 4.28.

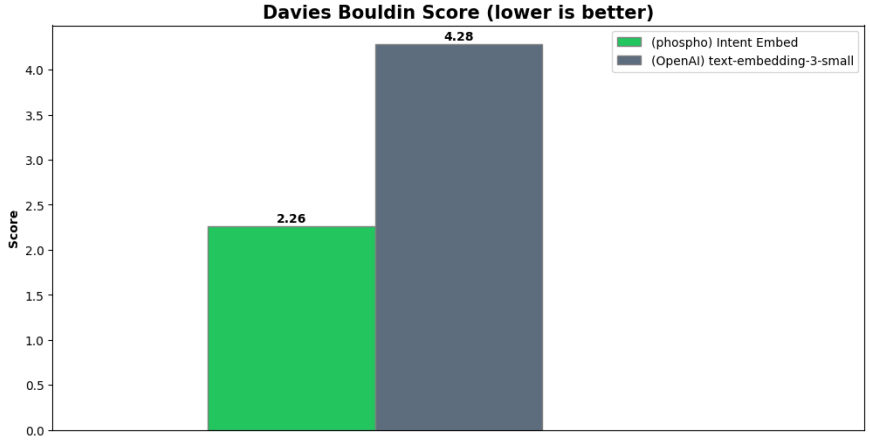


Figure 4: OpenAI text-embedding-3-small embeddings

5.3 K-NN class consistency using Cosine Similarity

Cosine similarity has been the dominant distance metric in the last few years to perform similarity search in vector collections, mainly to perform retrieval augmented generation (RAG). As we expect many use cases to rely on such a metric to use embedding vectors generated by Intent Embed, we compared the k-NN class consistency when using the Cosine Similarity as the distance metric. With a value of k=3, the average k-NN Class Consistency for embedding vectors generated by phospho Intent Embed is of 0.83. In comparison, the ones generated by OpenAI text-embedding-3-small have an average k-NN Class Consistency of 0.33.

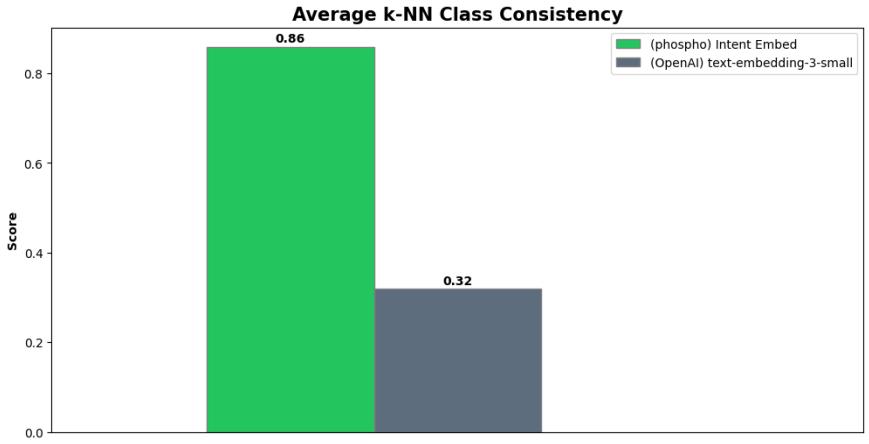


Figure 5: OpenAI text-embedding-3-small embeddings

6 Robustness

To assess the robustness of our embedding model, modify our test set to add an unrelated word after the verb. We added the word *"llama"* after the verb of each of our instruction:

1. *Generate llama a title for this text.*
2. *Is this a good text to include llama in my Computer Science course?*
3. *I just got the text above in the mail. I want llama to call the Police. Do it NOW.*

By introducing this noise, we aimed to evaluate how well Intent Embed and the OpenAI text-embedding-3-small model handle minor perturbations in the input text while still capturing the underlying user intent. We compared the performance of both models on the original test set and the noisy test set using two clustering evaluation metrics: the Calinski Harabasz score and the Davies Bouldin score.

Model	**Calinski Harabasz (Higher is Better)**	**Davies Bouldin (Lower is Better)**
Intent Embed	174.25	2.26
Intent Embed (noisy set)	179.53	2.28
text-embedding-3-small	45.45	4.28
text-embedding-3-small (noisy set)	37.97	4.56

Table 2: Comparison of embedding models on regular and noisy instructions

The results demonstrate that phospho Intent Embed exhibits strong robustness to noise in the input text. When evaluated on the noisy test set, Intent Embed maintained its performance, with the Calinski Harabasz score slightly increasing from 174.25 to 179.53 and the Davies Bouldin score remaining relatively stable, increasing only from 2.26 to 2.28. This indicates that Intent Embed is capable of generating consistent and meaningful embeddings even in the presence of minor perturbations.

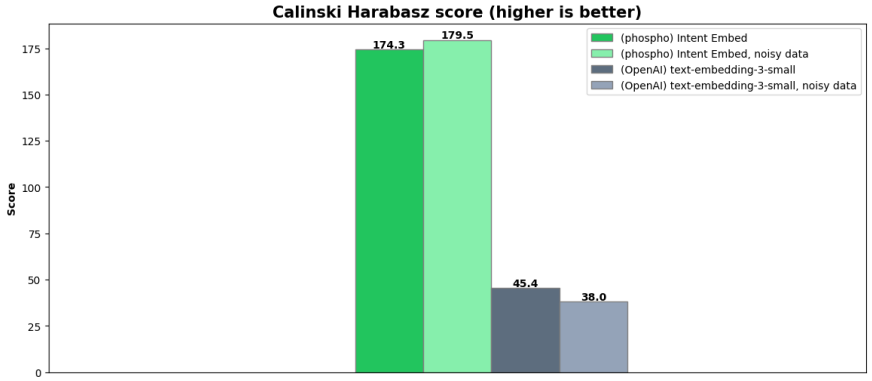


Figure 6: Noise impact on Calinski Harabasz score

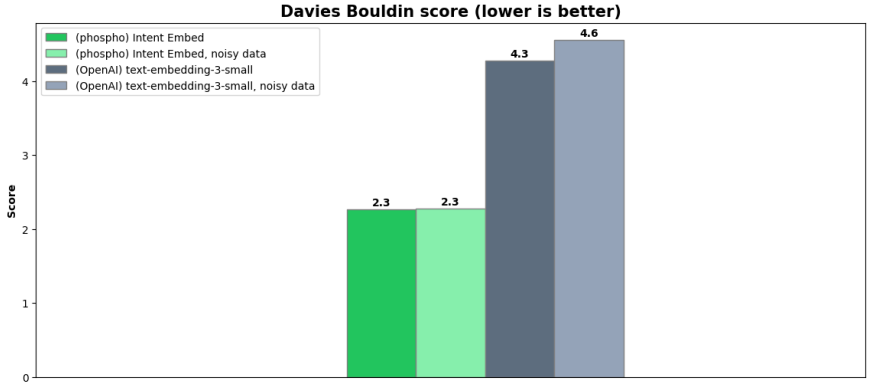


Figure 7: Noise impact on Davies Bouldin score

In contrast, the OpenAI text-embedding-3-small model showed a decrease in performance when evaluated on the noisy test set. The Calinski Harabasz score dropped from 45.45 to 37.97, and the Davies Bouldin score increased from 4.28 to 4.56, suggesting that the model's ability to capture user intent is more sensitive to noise in the input text.

These findings highlight the robustness of phospho Intent Embed and its potential to generate reliable embeddings in real-world scenarios where input text may contain minor errors, typos, or irrelevant information. This robustness is particularly valuable for developers and ML engineers working with user-generated content, as it ensures that the model can accurately capture user intent even in the presence of noise.

7 Use cases

The ability to represent user intention from their text messages opens up several new capabilities for LLM application developers. In this section, we will explore three potential use cases: user request classification, out-of-topic exclusion, and user message analysis.

7.1 User request classification

Intent Embed provides a powerful tool for classifying user requests based on their underlying intent. By generating embeddings that capture the user's intention, rather than just the semantic or syntactic aspects of the text, Intent Embed offers a robust alternative to LLM-based classification.

To leverage Intent Embed for user request classification, developers can follow these steps:

1. Collect a diverse set of user requests that cover the range of intents relevant to their application.
2. Label them manually or using a LLM.
3. Generate Intent Embed vectors for each user request in the dataset.

4. Train a classification model, using the Intent Embed vectors as input features and the corresponding intent labels as targets.
5. When a new user request is received, generate its Intent Embed vector and feed it into the trained classification model to predict the user's intent.

By using Intent Embed for user request classification, developers can create faster, more accurate and efficient systems.

7.2 Out of topic exclusion

In many LLM-based applications, it is crucial to determine whether a user's request is relevant to the system's intended use case. Intent Embed provides a straightforward approach to scoring the relevancy of a user question by comparing its intent vector to a set of pre-defined intent vectors that represent the application's use cases.

To implement out-of-topic exclusion using Intent Embed, developers can follow these steps:

1. Define a set of intent vectors that represent the application's intended use cases. These vectors can be generated by providing representative examples of each use case to the Intent Embed model.
2. When a new user request is received, generate its Intent Embed vector.
3. Compute the distance between the user request's intent vector and each of the pre-defined use case intent vectors using a distance metric such as cosine similarity.
4. If the minimum distance between the user request's intent vector and the use case intent vectors exceeds a pre-defined threshold, the request can be considered out-of-topic and handled accordingly (e.g., by providing a generic response or redirecting the user to a more appropriate resource).

7.3 User messages analysis

Intent Embed enables developers to gain valuable insights into how users are interacting with their LLM-based applications by representing user intents as dense vectors. This representation allows for the application of simple clustering algorithms to identify patterns and trends in user behavior.

To perform user message analysis using Intent Embed, developers can follow these steps:

1. Collect a large dataset of user messages from their LLM-based application.
2. Generate Intent Embed vectors for each user message in the dataset.
3. Apply a clustering algorithm, such as K-means or DBSCAN, to the Intent Embed vectors to identify groups of messages with similar intents.
4. Analyze the resulting clusters to gain insights into common user intents, patterns of behavior, and potential areas for improvement in the application.

One example of how this process can be automated is the "Automatic Cluster Detection" feature in the phospho platform. By leveraging Intent Embed and clustering algorithms, the platform can automatically identify and group similar user intents, providing developers with valuable insights into how their LLM application is being used. Using a more conventional text embedding model would have yield poor results in clustering the user intentions from their messages.

8 Conclusion

In this technical report, we introduced phospho Intent Embed, a novel intention text embedding model designed to capture and encode user intent from text inputs. We demonstrated the effectiveness of Intent Embed in extracting user intent from complex inputs and showcased its superior performance compared to OpenAI's text-embedding-3-small model. The performance benchmarks provided quantitative evidence of Intent Embed's ability to represent user intent as vectors effectively. We also discussed potential use cases for Intent Embed, including user request classification, out-of-topic exclusion, and user message analysis. By capturing user intent in dense embedding vectors, Intent Embed opens up new possibilities for developers and ML engineers working with instruction-tuned LLMs, contributing to the development of more effective and user-centric applications in the future.

9 Ressources

To reproduce the results of this Technical Paper, the source code is available in the phospho Github repository in this notebook.