# Intent Discovery for Enterprise Virtual Assistants: Applications of Utterance Embedding and Clustering to Intent Mining

**Minhua Chen**
Interactions LLC
mchen@interactions.com

**Badrinath Jayakumar**
Interactions LLC
bjayakumar@interactions.com

**Michael Johnston***
Amazon Alexa AI
mjohnstn@amazon.com

**Eman Mahmoodi**
Interactions LLC
smahmoodi@interactions.com

**Daniel Pressel**
Interactions LLC
dpressel@interactions.com

## Abstract

A key challenge in the creation and refinement of virtual assistants is the ability to mine unlabeled utterance data to discover common intents. We develop an approach to this problem that combines large-scale pre-training and multi-task learning to derive a semantic embedding that can be leveraged to identify clusters of utterances that correspond to unhandled intents. An utterance encoder is first trained with a language modeling objective and subsequently adapted to predict intent labels from a large collection of cross-domain enterprise virtual assistant data using a multi-task cosine softmax loss. Experimental evaluation shows significant advantages for this multi-step pre-training approach, with large gains in downstream clustering accuracy on new applications compared to standard sentence embedding approaches. The approach has been incorporated into an interactive discovery tool that enables visualization and exploration of intents by system analysts and builders.

## 1 Introduction

To build an enterprise virtual assistant capable of providing effective interaction with customers, intent detection – automatically detecting the customer's intent based on their input – is an indispensable component. Large-scale pre-trained language models have shown promising abilities in few-shot classification, and high accuracy intent detection can now be obtained with limited amounts of labeled data (Devlin et al., 2019; Henderson et al., 2020; Vulic et al., 2021). However, there are still situations where no labeled data is available. This situation is commonly encountered when designing a dialogue system for a brand new application. In this case, we would like to identify common intents from any available unlabeled data in the domain, such as call transcripts or chat logs. Also, for deployed applications, intent discovery can be applied to 'no-match' data; that is, utterances that the current system does not handle.

At first glance, the problem of intent discovery from unlabeled data appears similar to text clustering, which is well-studied in the literature. One natural approach for clustering is to use Transformer-based sentence embedding methods (Devlin et al., 2019; Reimers and Gurevych, 2019) to represent each utterance as a fixed-length vector, and then apply standard clustering methods such as K-means to extract intent clusters (Wu and Xiong, 2020; Aharoni and Goldberg, 2020). However, these models are mostly pre-trained on generic data such as Wikipedia or Natural Language Inference (NLI) datasets, which are quite different from typical enterprise customer service data. Hence there is a domain mismatch between the pre-trained model and the downstream application task.

In our experience, enterprise virtual assistant data has several key characteristics. First, utterances are mostly short, containing only a few words. Additionally, they contain some level of noise from Automatic Speech Recognition (ASR) transcriptions. Moreover, the data distribution is affected by the customer service communication channel. For example, there are many calls asking to speak to a live agent in order to bypass the system. Finally, the semantics differ at some points from ordinary language, because of business logic and design constraints. For example, the two utterances "my screen is broken" and "power button not responding" may be treated as containing the same semantic intent of "TECH-SUPPORT", while in common datasets (e.g., NLI), they would likely be considered different.

For these reasons, directly applying a generically pre-trained Transformer encoder (such as BERT (Devlin et al., 2019), or even an adapted model like Sentence-BERT (Reimers and Gurevych, 2019)) may not be optimal for the intent discovery problem. However, in our data lake we have accumu-

*Work completed at Interactions LLC



*Proceedings of NAACL-HLT 2022: Industry Track Papers*, pages 197 - 208

lated large amounts of utterance data from a large number of applications across multiple business verticals, all with intent labels either from production understanding models or human annotators. We, therefore, hypothesized that continuing pre-training on our existing virtual assistant data, to obtain a domain-specific utterance encoder, could be beneficial for downstream intent discovery tasks.

We propose a three-step solution (Figure 1) to the intent discovery problem: **1)** Generic Transformer Pre-training, **2)** Domain-adaptive Pre-training, **3)** Downstream Embedding and Clustering.

This approach is related to the don't-stop-pretraining paradigm proposed in (Gururangan et al., 2020), in which a generic Language Model (LM) is adapted to the target domain (or task) through domain (or task) adaptive pre-training. However, a key difference is that we leverage supervised data for the domain-adaptive pre-training in the second step; on the contrary, they only leverage unlabeled data for the continued pre-training. Recently Vulic et al. (2021) proposed ConvFIT where supervised contrastive learning is performed after generic LM pre-training. ConvFIT uses a small amount of labeled data from the same task in the adaptive pre-training step, which can be viewed as supervised task-adaptive pre-training for few-shot learning. In contrast, we use a collection of labeled data across a large number of customer service use cases in the second step, which can be viewed as supervised domain-adaptive pre-training for downstream clustering tasks. Notice that no access to the downstream data is needed for our domain-adaptive pre-training step, which makes our model reusable for new and unseen applications and use cases.

## 2 The Three-step Approach

### 2.1 Step 1: Generic Transformer Pre-training

Transformer-based pre-training such as BERT (Devlin et al., 2019) and GPT and its variants (Radford et al., 2018, 2019; Brown et al., 2020) have changed the landscape of natural language processing. Through self-supervised pre-training on large amounts of public data, Transformers can learn many language regularities, from syntax to semantics to even commonsense knowledge (Manning et al., 2020; Tenney et al., 2019). These pre-trained models then serve as excellent starting points for downstream tasks through model fine-tuning. Instead of adopting a publicly available pre-trained model, we pre-trained our own Transformer-based language model from public dialogue data, including three years of Reddit (Al-Rfou et al., 2016; Henderson et al., 2019), online forums, as well as customer reviews and Wikipedia. We use a masked language model (MLM) training loss and train an 8-layer model with eight attention heads and relative positional encoding (Shaw et al., 2018) on full conversations (Pressel et al., 2022), where each turn is demarcated with a special end-of-utterance token. We also place the layer norm at the front of each sub-layer in the Transformer to simplify training and improve performance (Nguyen and Salazar, 2019; Xiong et al., 2020; Wang et al., 2019). We found that, despite its smaller size, our model often outperforms much larger previously created models on many downstream dialogue related tasks, including intent detection, slot-filling, belief state tracking, probing, and few-shot learning. We use this in-house pre-trained model as our starting point for the intent discovery task, and leverage the mead-baseline (Pressel et al., 2018) package for the implementation.

### 2.2 Step 2: Domain-adaptive Pre-training

The premise of the domain-adaptive pre-training step is that the model can learn from a broad spectrum of existing use cases covering different business verticals so that the adapted encoder is applicable to new previously unseen use cases. To achieve this goal, we drew a balanced amount of (utterance, intent) sample pairs from each of 20 applications in our enterprise virtual assistant database thereby ensuring that applications with larger data volume do not dominate the pre-training data. The applications cover a broad range of verticals including insurance, telecommunications, consumer electronics, financial, retail, travel, and utilities.

Additionally, we note that many of the application models were designed independently, yielding differences in the naming conventions for intent labels across applications, even for intents with the same semantic meaning. For example, the technical-support intent could be named "TECH-SUPPORT" in one application but "TECHNICAL-SERVICE" in another. Consequently, we could not simply merge data from different applications and pre-train one single model. To deal with this problem, we employed a multi-task approach to pre-training where the Transformer encoder is shared between different applications, but the intent clas-

sifier is specific for each application. This ensures that varying labels across applications do not compete with each other in the softmax loss.

If we directly use a linear classifier with standard softmax loss for each task, the embeddings (i.e., feature inputs) to the softmax loss will be trained to yield linear discrimination, but may not preserve the distance metrics which are critical for our downstream clustering task. To preserve distance-metrics in the geometry, we replace the standard softmax with cosine softmax, where the logit score inside softmax is computed via the cosine similarity between the embeddings and the classifier weights. The resulting approach has a novel multi-task cosine softmax loss for domain-adaptive pre-training to accommodate the nature of our enterprise virtual assistant data and the downstream clustering task. This is the key contribution of this paper. We will present more details in Section 3.

### 2.3 Step 3: Downstream Embedding and Clustering

After the model is pre-trained and adapted to our enterprise virtual assistant domain, we can apply it to any new application for intent discovery. We apply the encoder to each utterance from the new application to obtain $\ell_2$ normalized utterance embeddings, and run K-means clustering on the embeddings to extract intent clusters.

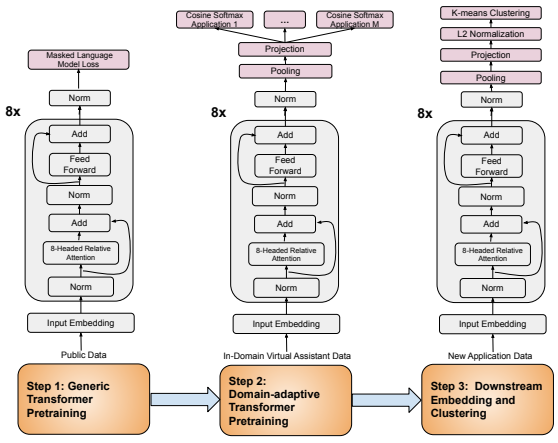


Figure 1: The Three-step Approach

## 3 Model Description

We describe here the model formulation for Step 2 of our intent discovery pipeline. We collected pre-training data from $M = 20$ applications in our data lake, and for each application $m$ we sample $n^{(m)}$ (utterance, intent) pairs $(\boldsymbol{x}_i^{(m)}, y_i^{(m)})$, $(i = 1, 2, \cdots, n^{(m)})$. Here $\boldsymbol{x}_i^{(m)}$ is a raw utterance from ASR transcription, $y_i^{(m)} \in \{1, 2, \cdots, C^{(m)}\}$ is the intent label for that utterance, the total number of intents $C^{(m)}$ for application $m$ is about 1000, and the number of samples $n^{(m)}$ for application $m$ is around one million.

Our multi-application intent classifier is a multi-task learning model where the utterance encoder is shared across applications but the classifier is built separately for each application. Mathematically the model can be formulated by minimizing the following loss function $L(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{m=1}^{M} \sum_{i=1}^{n^{(m)}} \ell^{(m)} \left( \boldsymbol{h}(\boldsymbol{x}_i^{(m)}), y_i^{(m)} \right)$ where $\boldsymbol{h}$ is the utterance encoder which is shared across applications and initialized from the generic pre-training on public data in Step 1. It consists of the pre-trained Transformer layers followed by mean-pooling to yield a fixed-length representation of the utterance, and a Multilayer Perceptron (MLP) to project it to a desired embedding space. During pre-training we randomly sample a mini-batch of utterances from the pre-training data, pass it through the same Transformer encoder $\boldsymbol{h}$, and then use different classifier heads for different utterances depending on which applications they come from. This multi-task learning approach has been used successfully in the literature to learn sentence embeddings (Liu et al., 2019; Wei et al., 2021). However, our approach differs in its use of the cosine softmax loss in $\ell^{(m)}$ with a distance-metric preserving property as explained below. A standard softmax loss computes the cross-entropy between the intent prediction distribution and the label as follows:

$$\ell^{(m)} \left( \boldsymbol{h}(\boldsymbol{x}_i^{(m)}), y_i^{(m)} \right) = - \sum_{c=1}^{C^{(m)}} 1(c = y_i^{(m)}) \times \log \frac{\exp \left( \boldsymbol{h}(\boldsymbol{x}_i^{(m)})^\top \boldsymbol{\theta}_c^{(m)} \right)}{\sum_{c'=1}^{C^{(m)}} \exp \left( \boldsymbol{h}(\boldsymbol{x}_i^{(m)})^\top \boldsymbol{\theta}_{c'}^{(m)} \right)} \quad (1)$$

where $\boldsymbol{\theta}_c^{(m)}$ is the classifier vector for intent $c$ in application $m$. However, as discussed in Section 2.2, this loss is not a good option for our downstream intent discovery task. For example, two nearby utterances in the embedding space could belong to different intent classes, if they lie on different sides of the linear boundary. To remedy this, we replace the above standard softmax loss with a cosine softmax loss as follows

$$\ell^{(m)}\left(\boldsymbol{h}(\boldsymbol{x}_i^{(m)}), y_i^{(m)}\right) = -\sum_{c=1}^{C^{(m)}} 1(c = y_i^{(m)}) \times \log \frac{\exp\left(\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(m)})^\top \bar{\boldsymbol{\theta}}_c^{(m)}/\tau\right)}{\sum_{c'=1}^{C^{(m)}} \exp\left(\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(m)})^\top \bar{\boldsymbol{\theta}}_{c'}^{(m)}/\tau\right)} \tag{2}$$

Here $\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(m)}) = \boldsymbol{h}(\boldsymbol{x}_i^{(m)})/\|\boldsymbol{h}(\boldsymbol{x}_i^{(m)})\|$ and $\bar{\boldsymbol{\theta}}_c^{(m)} = \boldsymbol{\theta}_c^{(m)}/\|\boldsymbol{\theta}_c^{(m)}\|$ are normalized unit vectors which will be used as the final embeddings for the utterances and intents, and $\tau$ is a pre-defined temperature parameter. Since the cosine similarity is related to the distance metric via $\boldsymbol{v}_1^\top \boldsymbol{v}_2 = -\|\boldsymbol{v}_1 - \boldsymbol{v}_2\|^2/2 + 1$ for $\ell_2$ normalized vectors, the cosine softmax pushes the embeddings of the utterances and the corresponding intents to be close to each other, which will yield the distance-metric preserving property we desire. Appendix A.3 's figure visually represents the difference between standard softmax and cosine softmax.

This is the novel multi-task cosine softmax loss we propose in this paper. Notice that both the utterance encoder $\boldsymbol{h}$ and the unnormalized intent embeddings $\boldsymbol{\theta}$ are learned in this domain-adaptive pre-training process, where the Transformer is initialized from generic pre-training in Step 1 and the intent embeddings are initialized randomly as a look-up table. After this Step 2, the adapted utterance encoder, which summarizes all enterprise virtual assistant data characteristics and business logics from multiple existing applications, can then be applied to downstream intent discovery tasks for new applications in Step 3. We also mention alternative modeling approaches in Appendix A.1.

# 4 Experiments

## 4.1 Experimental Methodology

To monitor the quality of the domain-adaptive pre-training in Step 2, we randomly select $4\%$ from the supervised pre-training data as a validation set and predict the intent following the multi-task cosine softmax loss proposed in Section 3,

$$\begin{aligned}\tilde{y}_i^{(m)} &= \underset{c\in\{1,2,\cdots,C^{(m)}\}}{\text{argmax}} \log \frac{\exp\left(\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(m)})^\top \bar{\boldsymbol{\theta}}_c^{(m)}/\tau\right)}{\sum_{c'=1}^{C^{(m)}} \exp\left(\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(m)})^\top \bar{\boldsymbol{\theta}}_{c'}^{(m)}/\tau\right)} \\ &= \underset{c\in\{1,2,\cdots,C^{(m)}\}}{\text{argmin}} \|\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(m)}) - \bar{\boldsymbol{\theta}}_c^{(m)}\|^2\end{aligned} \tag{3}$$

which is essentially performing a nearest neighbor intent search in the embedding space. We compare the true intent label $y_i^{(m)}$ with the above predicted intent $\tilde{y}^{(m)}$ to compute the pre-training accuracy for Step 2. The domain-adapted Transformer model achieved an accuracy of $91.5\%$ on this validation set, which demonstrates the consistency of the true intent labels and the high quality of the adapted Transformer model.

However, the ultimate goal of our model is not to test the intent classification accuracy for our existing applications but to perform intent discovery on a new application. It is not possible to directly apply equation (3) for intent discovery, as we do not have the intent embeddings $\bar{\boldsymbol{\theta}}^{(m)}$ for the new application. A natural approach for this problem is to embed all utterances from a new application $t$ using the pre-trained application-independent encoder $\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(t)})$ and apply a clustering algorithm. Then, we simultaneously identify the centroids as the discovered intent embeddings and the cluster indices as intent predictions. This is the Step 3 in our intent discovery pipeline. For simplicity and a fair comparison with other methods, we use the K-means algorithm to perform clustering on the new application's data,

$$(\tilde{y}_i^{(t)}, \tilde{\boldsymbol{\theta}}^{(t)}) = \underset{c_i\in\{1,2,\cdots,K\}, \bar{\boldsymbol{\theta}}^{(t)}}{\arg\min} \sum_{i=1}^{n^{(t)}} \|\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(t)}) - \bar{\boldsymbol{\theta}}_{c_i}^{(t)}\|^2 \tag{4}$$

From equation (4), we can see that we are essentially using an enterprise virtual assistant domain adapted utterance encoder for this clustering problem. We will evaluate the performance of the proposed intent discovery pipeline objectively and subjectively.

## 4.2 Objective Evaluation

For objective evaluation of the proposed method, we collected four additional existing applications from different business verticals and treated each of them as a "new" candidate application for intent discovery. Notice that these four applications are different from the 20 existing applications used in our Step 2 pre-training. Thus, we ensure that this "new" application simulation process is valid and the evaluation is fair. The details of the four applications are provided in Table 2 in the appendix. The intent distribution for each of the four applications is imbalanced, and about 50% samples of each application contains the top five intents of the application.

Evaluating clustering results is a challenging task, as there might be different (but all valid) ways to partition the data (Färber et al., 2010). In our objective evaluation we choose the production intent

label as the ground truth label, as this is the real business model we deployed for the application, and it is consistent with our enterprise virtual assistant pre-training process in Step 2. Inevitably there will be noise in the production intent labels. However, according to our offline human evaluation, the noise level is reasonably low.

For testing application $t$, we collect samples $(\boldsymbol{x}_i^{(t)}, y_i^{(t)})$ from our data lake, and only leverage the utterance part $\boldsymbol{x}_i^{(t)}$ for intent discovery in Step 3 via equation (4). We then evaluate the intent discovery performance through the standard clustering evaluation metrics (Wagner and Wagner, 2007; Xu et al., 2017), which take the ground truth intent labels $y_i^{(t)}$ and the clustering indices $\tilde{y}_i^{(t)}$ from equation (4) as inputs, and output evaluation scores (the higher the better) to measure the clustering quality. We use the following clustering evaluation metrics: **ACC** (Accuracy with label set alignment), **ARI** (Rand Index Adjusted for chance), **NMI** (Normalized Mutual Information), and **AMI** (Adjusted Mutual Information). **ACC** is defined as $\text{ACC} = \max_g \sum_{i=1}^{n^{(t)}} 1\{y_i^{(t)} = g(\tilde{y}_i^{(t)})\}/n^{(t)}$ where $g$ ranges over all possible one-to-one mappings between cluster indices and ground truth labels. In practice, we use the Hungarian algorithm (Kuhn, 1955) to identify the optimal mapping that produces the best accuracy score. The definition of **ARI**, **NMI** and **AMI** together with training details are available in Appendix A.4 and A.5.

We evaluate the following intent discovery approaches using the above evaluation metrics.

1. **Step 1 + Step 2 + Step 3.** This is the intent discovery procedure we propose in this paper. See Figure 1 for the full pipeline.

2. **Step 1 + Step 2 (Standard) + Step 3.** This approach is similar to Step 1 + Step 2 + Step 3; however, in Step 2 we use the standard softmax loss in equation (1) instead of the cosine softmax loss in equation (2). Consequently, we do not have the $\ell_2$ normalization for Step 3 in Figure 1.

3. **Step 2 + Step 3.** In this baseline approach, we skip the generic Transformer pre-training in Step 1 and randomly initialize the Transformer weights to start the enterprise virtual assistant domain adaptive pre-training in Step 2.

4. **Step 1 + Step 3.** In this baseline approach, we only use the utterance encoder pre-trained on generic data for intent discovery, and no enterprise virtual assistant domain adaptation in Step 2 is applied. As a result, we do not have the projection layer and $\ell_2$ normalization for Step 3 in Figure 1.

5. **SBERT + Step 3.** In this baseline approach, we directly borrow a publicly available sentence encoder Sentence-BERT model (Reimers and Gurevych, 2019) to replace the $\bar{\boldsymbol{h}}(\boldsymbol{x}_i^{(t)})$ encoder in (4).

6. **DEC.** In this baseline approach, we re-implemented the Deep Embedded Clustering (DEC) algorithm (Xie et al., 2016). Here, the utterance encoder is pre-trained on testing data alone, and it is fine-tuned during the clustering process (Hadifar et al., 2019).

Since we do not know the true number of clusters in advance, we presented **ACC** result with different $K$ values in Figure 2. The results for **ARI**, **NMI**, and **AMI** with different $K$ values are presented in Appendix A.7. From the results, we can make the following observations:

1. The models with domain-adaptive pre-training perform much better than all other methods without domain-adaptive pre-training (including the state-of-the-art Sentence-BERT model) across different testing datasets. The performance improvement can be as high as $20\%$ absolute in clustering accuracy. This performance clearly shows the benefit of domain-adaptive pre-training for downstream clustering tasks.

2. Step 1 + Step 2 (Standard) + Step 3 performs much worse than Step 1 + Step 2 + Step 3 in downstream clustering accuracy, according to Figure 2. This performance gap is expected as pre-training with the standard softmax loss in Step 2 does not induce the distance-metric preserving property, which is important for distance-based clustering such as K-means in downstream tasks.

3. In most cases, Step 2 + Step 3 performs similarly as Step 1 + Step 2 + Step 3, which means that the generic pre-training in Step 1 is not helping much for the downstream clustering task. The reason might be that we already

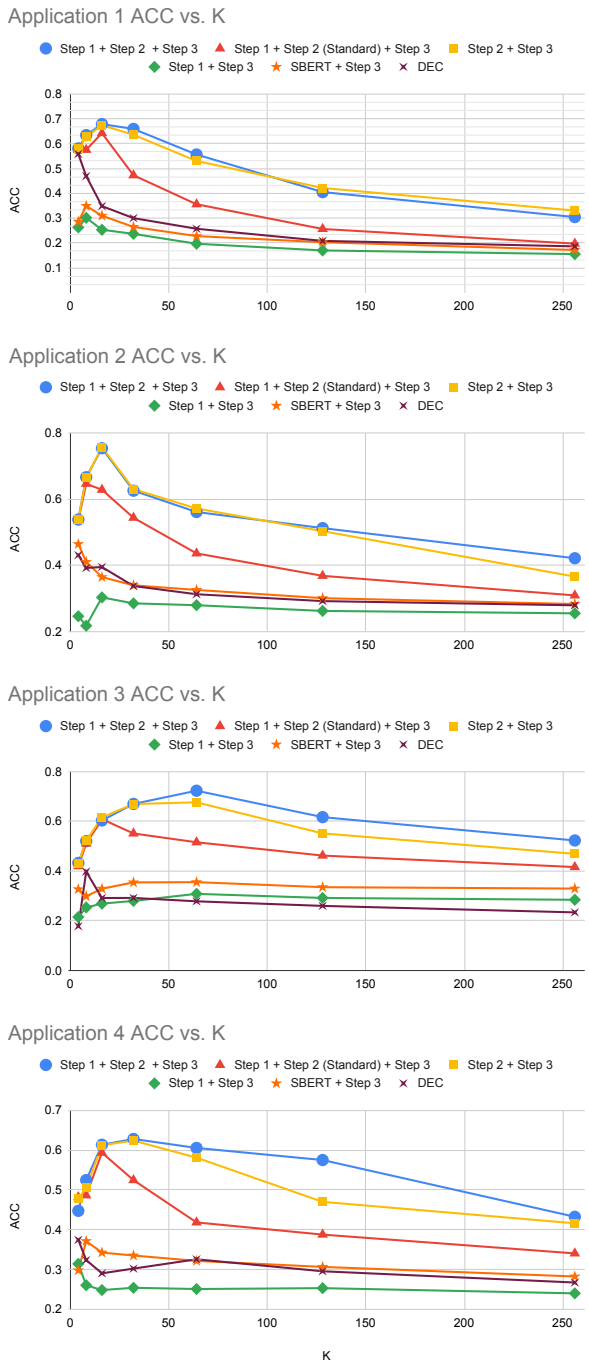


Figure 2: ACC of intent discovery for four applications with $K = [4, 8, 16, 32, 64, 128, 256]$. Resolution of sub-figures is automatically adjusted to show differences in the curves, which causes y-axis' scale to be different.

have a large amount of enterprise virtual assistant domain pre-training data in Step 2. As a result, the Transformer initialization from the generic pre-training will not have a lot of impact on the domain-adaptive pre-training result. Nevertheless, we still see that in some instances (e.g., **ACC** curves for Application 4 in Figure 2 ), Step 2 + Step 3 experienced performance drop compared with Step 1 + Step 2 + Step 3, indicating that the generic pre-training in Step 1 improves the robustness of the model across different domains.

4. In our error analysis, we observe that Step 1 + Step 2 + Step 3 tends to keep semantically similar utterances (e.g., "pay my bill" and "make a payment") into one cluster; however, other methods tend to split semantically similar utterances into multiple clusters. This leads to degradation in performance in other methods. In addition to that, our applications contain Spanish data with low frequency. Interestingly, on Step 1 + Step 2 + Step 3 results, we find that the cluster centroid where Spanish speakers want to speak to a live agent (e.g., "o hablar con un representante") and the cluster centroid where English speakers want to speak to a live agent (e.g., "i need to talk to a representative") are proximal. Such proximity is not well pronounced in other methods.

In summary, the proposed intent discovery pipeline with domain-adaptive pre-training outperforms other methods by a large margin. Hence, by continuing pre-training the Transformer on in-domain enterprise virtual assistant data from multiple existing applications, we can effectively distill the knowledge of enterprise virtual assistant data characteristics and business logic into the Transformer encoder, which provides high-quality utterance embeddings and excellent intent discovery results on unseen applications.

### 4.3 Intent Discovery Portal

For new applications and for unexpected inputs to a deployed application, we generally do not have labels that can be used for objective evaluation. In order to extract value from the results of semantic embedding on new data we built out an interactive intent discovery portal that enables visualization, interactive inspection of intent clusters, and subjective evaluation. Figure 3 shows the intent discovery user interface. The left panel contains an un-directed graph in which semantic closeness among clusters is represented by graph links in a minimum spanning tree. The graph is interactive and when the user selects a node in the graph, the middle panel shows a list of examples of members of the cluster along with an interactive phrase cloud.

The right panel supports replay of audio (if available) and presents a list of related examples and intents from a large database of existing application data. More details on the intent discovery portal are available in Appendix A.2 and A.6.

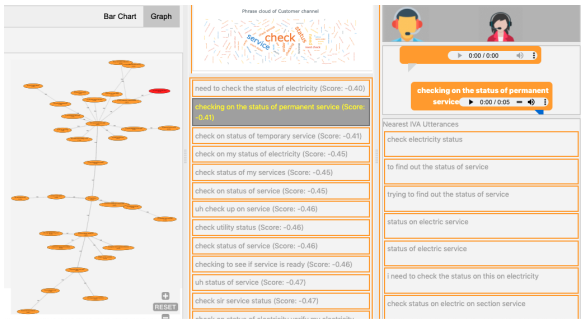


Figure 3: Intent Discovery Portal View.

## 5 Conclusion

This paper proposed a practical three-step solution to the challenge of intent discovery for virtual assistants. From our experiments, we found that a domain-adaptive pre-training step is essential for achieving good downstream clustering performance. Through supervised pre-training, enterprise virtual assistant data characteristics and associated business logic are all distilled into the resulting utterance encoder. This three-step approach can be viewed as an extension of the don't-stop-pretraining paradigm (Gururangan et al., 2020) to the downstream clustering tasks. We also found that state-of-the-art generic sentence encoders may not yield the best sentence representations to specific industrial applications such as enterprise virtual assistants. A consequence of this domain-adaptive pre-training is that the resulting sentence encoder is no longer generic, hence cannot necessarily be applied to data beyond the domain of enterprise virtual assistants. However, the same methodology could be applied to any enterprise database, so that people can pre-train domain-specific sentence encoders to match their own needs.

## Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. We would also like to thank colleagues at Interactions LLC for their discussions and help on this work, especially Lou Nicotra for all the support on computing infrastructure.

## References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283.

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763.

Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2016. Conversational contextual cues: The case of personalization and history for response ranking. *arXiv preprint arXiv:1606.00372*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Ines Färber, Stephan Günnemann, Hans-Peter Kriegel, Peer Kröger, Emmanuel Müller, Erich Schubert, Thomas Seidl, and Arthur Zimek. 2010. On using class-labels in evaluation of clusterings. In *MultiClust: 1st international workshop on discovering, summarizing and using multiple clusterings held in conjunction with KDD*, page 1.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Amir Hadifar, Lucas Sterckx, Thomas Demeester, and Chris Develder. 2019. A self-training approach for short text clustering. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 194–199.

Matthew Henderson, Paweł Budzianowski, Iñigo Casanueva, Sam Coope, Daniel Gerz, Girish Kumar, Nikola Mrksic, Georgios P. Spithourakis, Pei hao Su, Ivan Vulic, and Tsung-Hsien Wen. 2019. A repository of conversational datasets. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 1–10.

Matthew Henderson, Iñigo Casanueva, Nikola Mrkvsi'c, Pei hao Su, Tsung-Hsien, and Ivan Vulic. 2020. Convert: Efficient and accurate conversational representations from transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2161–2174.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Daniel Pressel, Sagnik Ray Choudhury, Brian Lester, Yanjie Zhao, and Matt Barta. 2018. Baseline: A library for rapid modeling, experimentation and development of deep learning algorithms targeting nlp. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 34–40.

Daniel Pressel, Wenshuo Liu, Michael Johnston, and Minhua Chen. 2022. Lightweight transformers for conversational ai. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, page 1.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilyas Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*, page 1.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, page 1.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Ivan Vulic, Pei hao Su, Sam Coope, Daniel Gerz, Paweł Budzianowski, Iñigo Casanueva, Nikola Mrkvsi'c, and Tsung-Hsien Wen. 2021. Convfit: Conversational fine-tuning of pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1151–1168.

Silke Wagner and Dorothea Wagner. 2007. *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Chien-Sheng Wu and Caiming Xiong. 2020. Probing task-oriented dialogue representation from language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5036–5051.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR.

Jiaming Xu, Bo Xu, Peng Wang, Suncong Zheng, Guanhua Tian, and Jun Zhao. 2017. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31.

# A Appendix

## A.1 Alternative Models

The cosine softmax loss in (2) is reminiscent of the cosine-similarity based loss used in contrastive learning (Chen et al., 2020; Radford et al., 2021). For example, in OpenAI CLIP (Radford et al., 2021) two modalities (text and image) are aligned in the latent space via a dual-encoder model and a cosine-similarity based loss. The cosine softmax loss we use in equation (2) is also aligning two modalities (utterance and intent). However, since the set of all possible intents for an application is finite and known in advance, we do not need to use the in-batch negative sampling approach as in the contrastive learning loss. Instead we contrast with all possible intents in the denominator of equation (2).

Another alternative approach is supervised contrastive learning (Khosla et al., 2020; Vulic et al., 2021), where the contrastive learning is done using just the utterance modality. In this approach, the positive utterance pair is obtained by sampling utterances with the same intents, while the negative utterance pair is obtained by sampling utterances with different intents. Then the supervised contrastive loss will impose the constrain that embeddings for the positive pair should live nearby and the embeddings for the negative pair should live further apart.

We note that both the multi-model contrastive learning and the supervised contrastive learning are valid modeling approaches for our domain-adaptive pre-training. However, we focus on the cosine softmax loss in equation (2) in this paper due to its simplicity and its straightforward nature, and leave the comparison to the above alternative models in our future work.

Another approach is to use the standard softmax (equation (1)) in Step 2, and then use spectral clustering in Step 3. There are a few issues in this approach. Firstly, the embedding geometry learned with standard softmax is not friendly to downstream clustering tasks. Secondly, there is no guarantee that the spectral embedding step in spectral clustering can infer the right semantic geometry, as no supervised pre-training data is used in Step 3. Lastly, spectral clustering is quite computationally expensive and could not scale to large downstream datasets. Empirically we also observe that this approach is inferior to our proposed approach. In contrast, our proposed approach shifts the heavy-lifting geometric manifold learning task to the cosine softmax (equation (2)) in Step 2, so that K-means is enough for the downstream clustering task.

## A.2 Intent Unification and Vector Search

A by-product of the above domain-adaptive pre-training is a mechanism for intent unification across multiple applications. Suppose intent $c$ in application $m$ and intent $c'$ in application $m'$ are semantically similar but named differently. Since the utterances (which are callers' realizations of the intents) associated with these two intents are similar, and they share the same utterance encoder, the embeddings for these utterances will live nearby. This again will drive the intent embeddings for the above two intents close together, according to the multi-task cosine softmax loss function. As a result, after the pre-training, semantically similar intents across applications will live close-by in the embedding space. Hence by simply exploring the intent embedding space through K-means clustering, we could unify intents across applications by grouping

semantically related intents together.

Another by-product is vector search for utterances. Since we have learned an utterance encoder specific for the enterprise virtual assistant domain in the Step 2 pre-training, we can embed any new utterance and convert it into a fixed-length vector. Since we already have utterance embeddings and intent embeddings for our pre-training data from the existing applications, for each new utterance or discovered cluster centroid, we can obtain the nearest utterances and intents from the pre-training data via K-nearest neighbor search. These nearest neighbors can be used to help to augment the query utterances or name the discovered intent clusters, which can greatly improve the interpretability of our intent discovery results. A potential risk of this vector search is that private information from our existing applications could be revealed to a new user of our system. Hence, careful considerations are needed when we activate this vector search feature, and all sensitive information should be redacted from the utterance pool used for this K-nearest neighbor search.

### A.3 Standard Softmax vs. Cosine Softmax

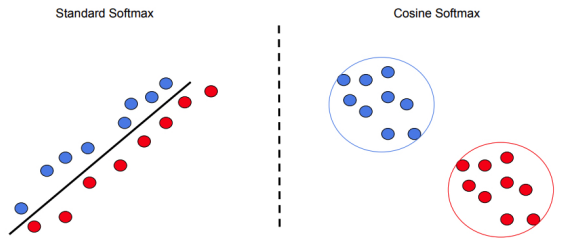


Figure 4: Standard softmax and cosine softmax induce different geometries in the embedding space

### A.4 Definition of Clustering Evaluation Metrics

Here are some descriptions for the clustering evaluation metrics. More formal definitions could be found in (Wagner and Wagner, 2007).

**ARI**: In the Rand Index, we consider all pairs of samples and see how often pairs are grouped consistently under the clustering results and the ground truth labels. In Adjusted Rand Index (ARI), this consistency frequency is adjusted by a base model to correct the impact of consistency by chance.

**NMI**: The Mutual Information directly measures the correspondence between the clusters and the ground truth classes via a probability measure of common samples between them. This measure is normalized by entropy of the partitions to yield the Normalized Mutual Information (NMI).

**AMI**: The above Normalized Mutual Information is further adjusted to account for chance and size of the clusters, to yield this Adjusted Mutual Information (AMI) measure.

| Hyperparameter | Description | Value |
|---|---|---|
| Pooling | Mean pooling layer output before projection (Figure 1) | 512 |
| Projection | Projection layer output before cosine softmax (Figure 1) | 256 |
| Epochs | Training epochs | 3 |
| Softmax | Type of softmax used in loss function (Figure 1) | cosine |
| Dropout | Layer dropout | 0.1 |
| Cosine Temperature ($\tau$) | Cosine softmax temperature (Equation 3) | 0.125 |
| Optimizer | Training optimizer | AdamW |
| Learning Rate | Learning rate for optimizer | 1.e-5 |
| Weight Decay | Weight decay for optimizer | 1.0e-3 |
| Clip | Gradient clip | 1.0 |
| Batch Size | Training batch size | 360 |
| Input Length | Maximum input length | 64 |
| Layers | Transformer layers | 8 |
| Multi-head attention | Number of head in the attention module | 8 |

Table 1: Step 2 Hyperparameters

| Name | Business | Samples | Intents | Average words |
|---|---|---|---|---|
| Application 1 | Collections | 707907 | 82 | 5 |
| Application 2 | Power Utility | 616145 | 432 | 4 |
| Application 3 | Insurance | 1000000 | 1830 | 4 |
| Application 4 | Photography | 921440 | 2521 | 5 |

Table 2: Testing Applications for Step 3

### A.5 Training Details

In this section we provide training details for each Step in the pipeline (Figure 1). For Step 1, we trained using Apache Licensed TensorFlow (Abadi et al., 2016) on a single v3 Tensor Processing Unit (TPU). For best performance on TPUs, we use bucketing based on full conversation lengths, scaling the number of samples for each bucket length so that the number of tokens is constant per batch. We use AdamW with a peak learning rate of 4e-4, a weight decay of 1e-3, and a linear warmup of 10,000 steps followed by cosine decay. For Step 2, hyperparameters are listed in Table 1, and we built our model using open source BSD-licensed PyTorch library (Paszke et al., 2019). Our model has 49 million parameters. The pre-training in Step 2 were conducted on two NVIDIA GeForce 1080Ti GPUs, and it took about 30 hours to finish. For Step 3, we used the open source BSD-licensed scikit-learn library (Pedregosa et al., 2011) for K-means clustering with default hyperparameters. More specifically, we used "k-means++" initialization with multiple runs to make the results more robust. The intent discovery results for downstream applications in Table 2 are reported in Figure 2, Figure 7, Figure 8, and Figure 9 with $K = [4, 8, 16, 32, 64, 128, 256]$.

## A.6 Additional Details on Intent Discovery Portal

Figure 5 provides a more detailed view of the graph representation of the automatic intent discovery results from the intent discovery portal. The nodes are automatically labelled with keywords from each cluster determined using TF-IDF, e.g., 'make bill payment pay' for a cluster associated with bill payment. Figure 6 shows an alternative view of the cluster data as a bar chart capturing the relative size of clusters in the data. The different views are interconnected, and the user can select an intent in one and see where it is in the alternate view.

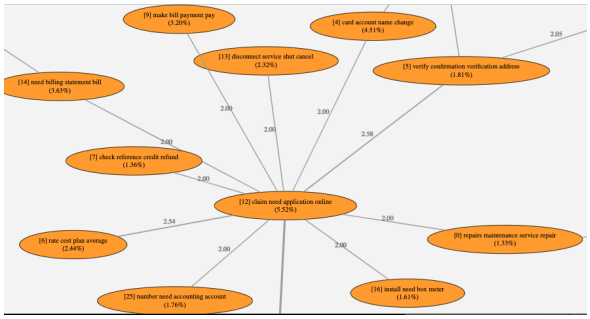


Figure 5: Detail view of the portal zoomed on few nodes in the un-directed graph.

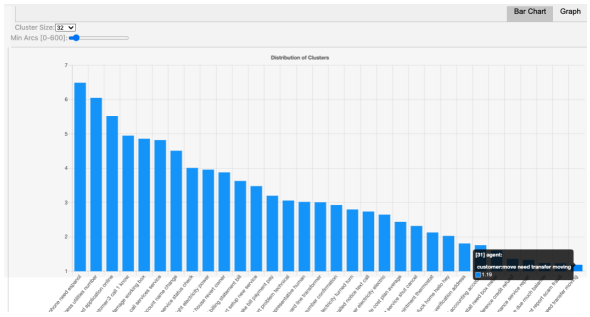


Figure 6: Cluster-size Bar Chart View.

## A.7 More Objective Evaluation (ARI, NMI, and AMI) on Testing Applications

Figure 7, Figure 8, and Figure 9 provide **ARI**, **NMI** and **AMI** results on our testing applications.

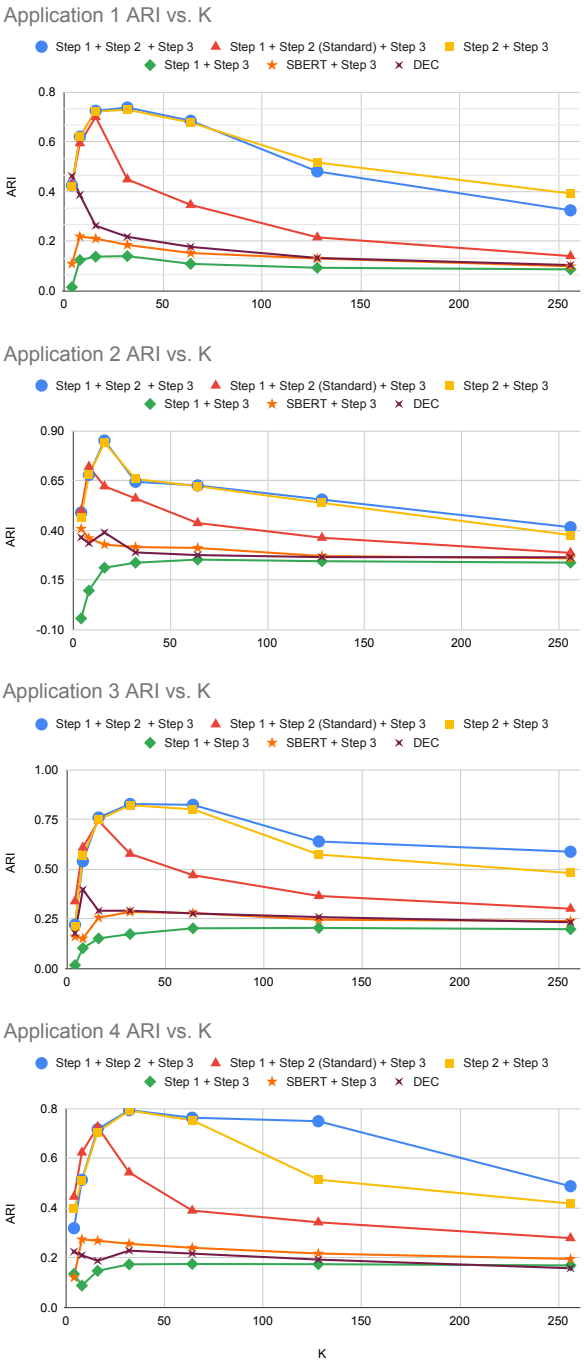


Figure 7: ARI of intent discovery for four applications with $K = [4, 8, 16, 32, 64, 128, 256]$.

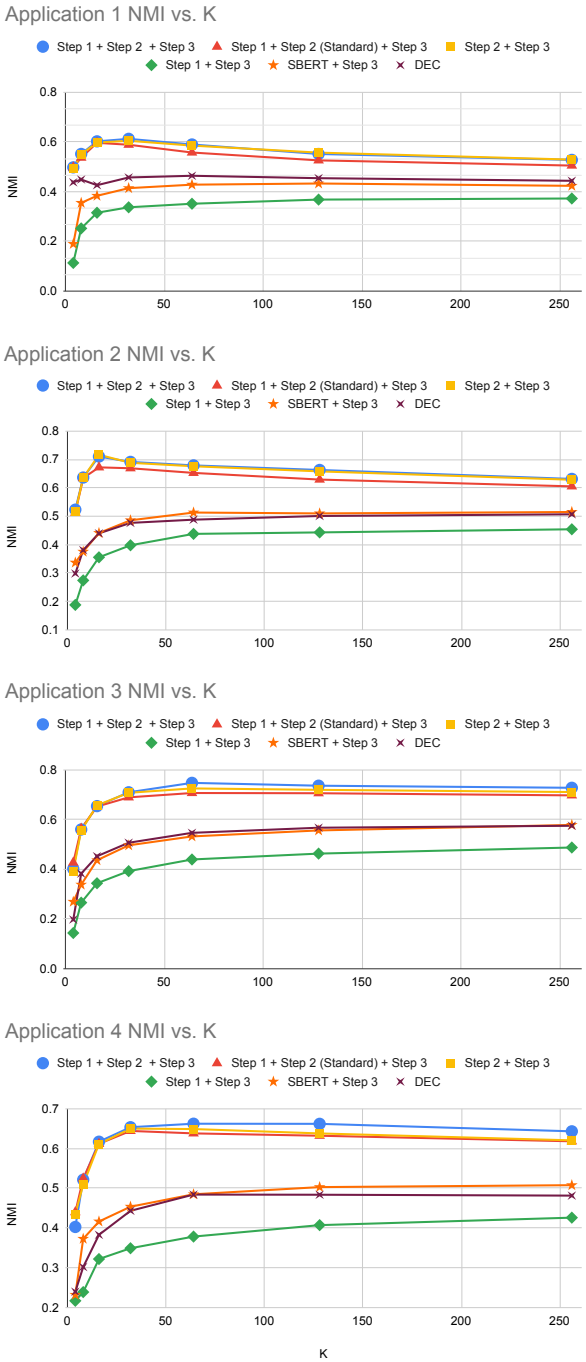


Figure 8: NMI of intent discovery for four applications with $K = [4, 8, 16, 32, 64, 128, 256]$.

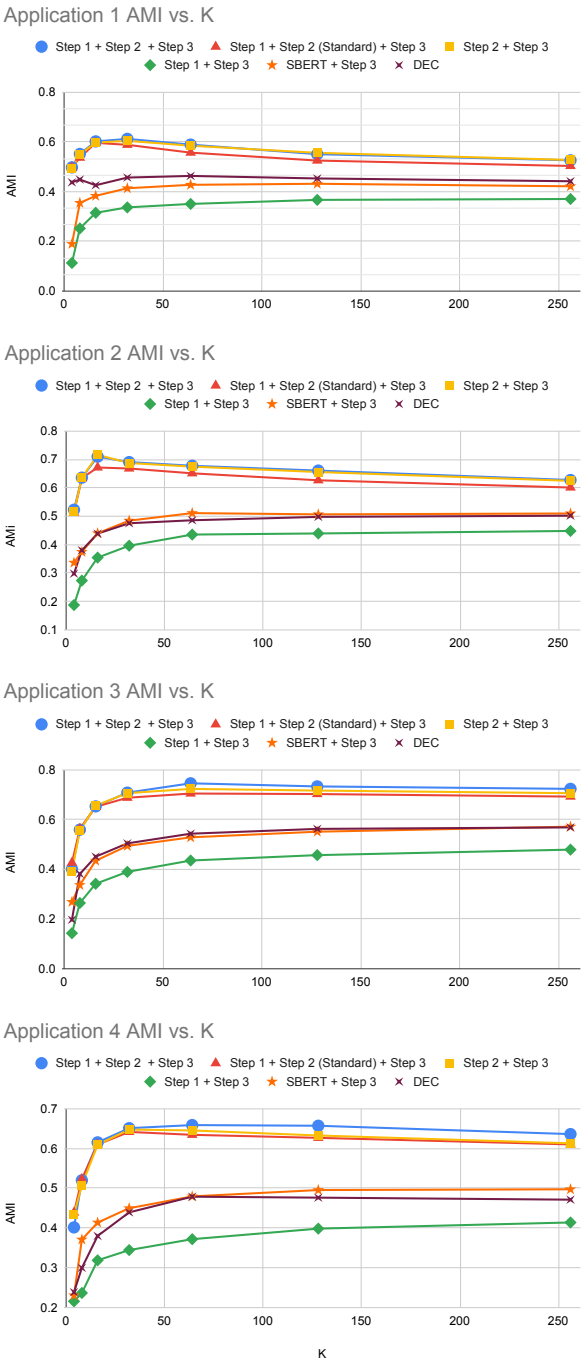


Figure 9: AMI of intent discovery for four applications with $K = [4, 8, 16, 32, 64, 128, 256]$.