SIMVECS: Similarity-based Vectors for Utterance Representation in Conversational AI Systems

Ashraf Mahgoub
Purdue University / West Lafayette, IN
amahgoub@purdue.edu

Youssef Shahin
Microsoft / Redmond, WA
Youssef.Shahin@microsoft.com

Riham Mansour
Microsoft / Redmond, WA
rihamma@microsoft.com

Saurabh Bagchi
Purdue University / West Lafayette, IN
sbagchi@purdue.edu

Abstract

Conversational AI systems are gaining a lot of attention recently in both industrial and scientific domains, providing a natural way of interaction between customers and adaptive intelligent systems. A key requirement in these systems is the ability to efficiently parse user queries, understand the intent behind each query, and provide adequate responses to users. Therefore, many applications such as conversation bots and smart IoT devices has a natural language understanding (LU) service integrated within. One of the greatest challenges of language understanding services is efficient utterance (sentence) representation in vector space, which is an essential step for most ML tasks. In this paper, we propose a novel approach for generating vector space representations of conversational utterances using pair-wise similarity metrics. The proposed approach uses only a few corpora to tune the weights of the similarity metric without relying on external general purpose ontologies. Our experiments confirm that the generated vectors can improve the performance of LU services in unsupervised, semi-supervised and supervised learning tasks over state-of-the-art prior works.

1 Introduction

Challenges of conversational AI systems: Conversational AI systems empower virtual assistants in many applications such as conversation bots (also known as Chatbots) and smart IoT devices. Their ability to understand user spoken commands and identify the user's intent(s) is one of their main benefits. However, this is a very challenging task due to the diversity of domains and languages they are required to support. For example, Microsoft LUIS[1] currently supports 12 languages, whereas IBM Watson [2] supports 10 languages in their language understanding services. Moreover, conversational text queries are often very short and sparse, which hinders conventional text representations such as Bag-of-words (BOW) from capturing adequate features of the utterance.

In LU services, a language understanding application is often represented as a group of intents and entities that serve a specific business application. For example, a restaurant application may define intents such as *Order Food*, *Show Menu* and *Cancel Order*. The task of the language understanding service then is to classify newly received queries (utterances) into one (or more) of the defined intents.

Although many prior works were focused on constructing word-level vector representation such as (Mikolov et al., 2013) and (Pennington et al., 2014), generating an utterance-level vector representation is still a challenging task. Existing language modeling (LM) based approaches such as Para2Vec (Le and Mikolov, 2014) rely on deep neural networks to generate vector representations for the paragraph-level. However, these approaches are normally trained with an abundance of utterances to achieve the required performance, which is usually not present in the conversational text domain (Boyanov et al., 2017). Moreover, pre-trained vectors (i.e., vectors generated from an external corpus) provide the same vector representation for all domains (i.e. static). The desirable characteristic on the other hand is that for the same utterance to have different vector representations, and correspondingly, different intents, in different domains. Additionally, the current approaches make the embeddings more prone to bias towards the domain of the training data (Bolukbasi et al., 2016).

[1] www.luis.ai/home

[2] www.ibm.com/watson/

We propose a vector representation method, SIMVECS that generates dynamic utterance-level vector representations for different LU applications. With SIMVECS, each utterance is represented as a vector of similarity scores to a set of automatically identified "representative utterances" within the same application. This way the same utterance can have different representations depending on the application. As an example: the utterance *"I want a large pizza"* can be of type *Order Food* for a restaurant application, while the same utterance can be of type *None* (i.e., outlier) for a bus-tracking application. Therefore, application-determined vector representations are essential for capturing the semantics of utterances in LU services and hence accurate mapping to user-defined intents.

The following list represents our key contributions:

1. A novel approach to combine multiple similarity sub-metrics into one metric, automatically adjusting the weight for each sub-metric to the overall similarity score.

2. A novel approach for the vector representation of each utterance in a conversational AI system.

3. A semi-supervised algorithm for refining the vector representations based on user's feedback.

The rest of the paper is organized as follows. Section 2 covers related work and separates our approach from existing solutions. Section 3 gives an overview of the main components in our proposed solution. Section 4 describes how we calculate the similarity scores between utterances and generate the vector representations. Sections 5, 6, and 7 evaluate the generated vectors in unsupervised, semi-supervised, and supervised learning tasks respectively and compares the performance of SIMVECS to several baselines.

2 Related Work

2.1 Similarity metrics

Measuring the similarity between natural language sentences is crucial in many tasks such as: Question-answering systems (Achananuparp et al., 2008b) and information retrieval (Li et al., 2006; Zahran et al., 2015). Authors in (Achananuparp et al., 2008a) provide an evaluation for 14 different similarity metrics. The authors reported the best metric found was a composite between word-order and ontology-based similarity. However, the weights for how much each of the two sub-metrics contributes to the overall similarity metric is selected based on human intuition. Such selection becomes harder when several sub-metrics are considered and multiple domains have to be satisfied. We consider these weights as hyper-parameters and hence use an automated technique that applies Genetic-Algorithms (GA) (Mitchell, 1996) to find the optimal weights as it has been used in multiple hyper-parameter tuning tasks such as in (Friedrichs and Igel, 2005), (Lam et al., 2001), and (Mahgoub et al., 2017).

2.2 Vector Representation

One of the most common vector representation for both documents and sentences is the Bag-of-Words (BOW) model (Harris, 1954). In BOW, the sentence is represented as a binary vector that donates the existence or absence of individual words (i.e. vocabulary) in that sentence. One of the major disadvantages of BOW representation is the loss of word order. Consequently, two sentences with totally different meaning can have very close representation just because they use similar vocabulary. A variation of the model is bag-of-ngrams (Mcnamee and Mayfield, 2004), which aims at preserving the word order. However, both representations are very sparse and generate vectors with very high dimensions. (Mikolov et al., 2013) proposed a technique that uses deep neural-networks to learn efficient representations for the word level. Although the trained vectors capture many semantic features, generating a sentence-level representation from individual words vectors is still a challenging task. One simple approach is to represent the sentence as weighted average of all the words in the document. However, this approach has the same weakness of not preserving the word order (Le and Mikolov, 2014). A recent approach proposed by (Le and Mikolov, 2014) generates both word-level and paragraph-level representations. However, the approach relies on training the network with a large corpus with billions of tokens, which is rarely available in the conversational text domain. Moreover, the generated vectors can still suffer from being biased by the train-

ing data domain and cannot generalize for different domains. (Dai and Le, 2015) proposed an approach to improve the vector representations with pre-trained recurrent neural networks. The proposed approach showed significant improvement over both BOW and Paragraph vectors. SIMVECS uses only a few corpora to tune the weights of the similarity metric without relying on external general purpose ontologies.

3 Overview of SIMVECS

SIMVECS relies on pair-wise similarity metrics. Each metric serves as a function that assigns similarity (or distance) scores to a pair of utterances. Many similarity metrics have been proposed in the literature. Although we use only six, our solution is generic and can incorporate any additional similarity sub-metrics.

3.1 Similarity sub-metrics definition

The six similarity sub-metrics which SIMVECS uses are:

(1)Unigrams: measures similarity based on the inverse-document-frequency (IDF) scores of common unigrams. The resulting score is then normalized by dividing over the sum of IDF scores of all unique words in the two utterances:

$$Sim_{uni}(U_i.U_j) = \frac{\sum_{w \in U_i \cap U_j} IDF[w]}{\sum_{w' \in U_i \cup U_j} IDF[w']} \quad (1)$$

(2)Character N-grams: measures similarity based on the overlapping character n-gram tokenization. We use an equation similar to the unigram except that words (and their corresponding IDF scores) are replaced by overlapping character N-grams. We use tokens of size n=4 as recommended by literature (Mcnamee and Mayfield, 2004).

(3)Bigrams: similar to unigram except that it uses tokens of two adjacent words. For each bigram, we set the IDF score to be the max between the two words in the token (presented as IDF_{bi}).

(4)Trigrams: similar to bigrams except that it uses tokens of three adjacent words.

(5)Utterance Length: this captures the similarity between a pair of utterances based on their number of tokens. Although this might be a dangerous feature to rely on individually, it becomes very useful in the domain of conversational text when combined with other features. For example, Table. 1 shows the average utterance lengths per intent in the WebApp corpus. We observe a large variance in utterance lengths of different intents. This is because one might need more tokens to specify a complex intent such as booking a flight (which requires lots of details) than simpler intents like asking for help or canceling a request. We use the following formula for utterance length similarity:

$$Sim_{len}(U_i.U_j) = \frac{min(Length(U_i), Length(U_j))}{max(Length(U_i), Length(U_j))} \quad (2)$$

(6)Word order: measures the normalized difference of word order between the two utterances.

$$Sim_{wo}(U_i.U_j) = 1 - \frac{||ri - rj||}{||ri + rj||} \quad (3)$$

where ri and rj are word order vectors (a vector which represents the order of each word in the utterance) of utterances U_i and U_j respectively.

3.2 IDF scores calculation

The first 4 metrics (i.e. unigram, bigram, trigrams, and character N-grams) rely on pre-calculated IDF scores. All these IDF scores are pre-calculated from a large corpus of conversational text generated from Cortana virtual assistant[3]. This corpus contains 18M utterances that resembles conversations between a user and Cortana in different domains. With the calculated IDF scores, these sub-metrics can be calculated and used to calculate a composite similarity metric as follows:

$$Sim_{comp}(U_i.U_j, \bar{W}) = \bar{W} \quad . \quad \bar{Sim}$$
$$= [W1..W6] \begin{bmatrix} Sim_{uni}(U_i.U_j) \\ Sim_{char}(U_i.U_j) \\ Sim_{bi}(U_i.U_j) \\ Sim_{tri}(U_i.U_j) \\ Sim_{len}(U_i.U_j) \\ Sim_{wo}(U_i.U_j) \end{bmatrix} \quad (4)$$

where W_i's are normalized weights which represent the collaboration of each sub-similarity metric to the overall metric. W_i's serve as hyper-parameters that control the quality of the composite similarity metric. The different sub-metrics have different relative importances in detecting similarities between utterances within the application. Therefore, the weights should be tuned automatically according to the LU application.

[3]https://www.microsoft.com/en-us/cortana

3.3 Application-based weights tuning

In this section, we describe our method in tuning the weight (W_i) for every sub-metric in Eq. 4. First, we use 15 real-world applications from a popular language understanding service to serve as our training and testing data set. These applications represent different domains (e.g. restaurants, flight booking services, smart homes, etc.) and they are created by system admins. Therefore, they contain a user-defined label for every utterance, which serves as our ground truth. For every pair of utterances with the same label, we assign a similarity score of 1 (max similarity). Similarly, for every pair of utterances with different labels, we assign a similarity score of 0 (min similarity). Now the task of tuning the weights for the sub-metrics can be viewed as an optimization problem, which is given by the following formula:

$$\bar{W}^* = \underset{\bar{W}}{\operatorname{argmin}} \sum_{\forall i,j} RMSE(Sim_{gt}(U_i.U_j), Sim_{comp}(U_i.U_j, \bar{W})) \quad (5)$$

Where Sim_{gt} is the ground truth similarity (either 0 or 1), $RMSE$ is the root mean square error between the estimated similarity score and the ground truth. Therefore, the target of equation 5 is to find the vector of weights that minimizes the differences between the estimated similarity scores and the ground truth similarly provided by application users. We use genetic algorithms (GA) to find the best values of $\bar{W}^*$. GA is a metaheuristic optimization algorithm that is inspired by biological evolution. It has the nice feature of balancing between exploration (a.k.a mutation) and exploitation (a.k.a crossover) of different solution candidates (a.k.a chromosome) in the search space (Črepinšek et al., 2013). GA is favorable in solving optimization problems which convexity is not known, since it does not rely on derivative information in finding good search directions (i.e. derivative-free). We used all pair-wise utterances from the 15 applications to train and validate our approach. Each candidate solution in GA simply represents a vector of weights in Eq. 4, and the fitness function is the resulting sum or RMSEs across all pairs of utterances (the lower the better). We perform 5-fold cross validation on the 15 applications and take the average of the best vectors of each fold. The resulting vector is then used for all subsequent experiments.

Intents	Avg. Tokens	Std. Tokens
Change Password	8.625	1.4
Delete account	7.35	1.11
Download video	7	0
Export data	10.2	2.28

Table 1: Average number of tokens per utterance for the WebApp Corpora.

4 Similarity-based utterance representation

We estimate all pair-wise similarity scores using Eq. 4 and store them in a matrix of size NxN where N is the number of utterances. We then apply Principal Component Analysis (PCA) (Wold et al., 1987) to reduce the dimensionality of this matrix. This pair-wise similarity matrix, *SimMatrix* for short, is then used to generate the vector representation for each utterance. Consider the example shown in Fig. 1. On the left hand, we show 10 utterances (selected from (Coucke et al., 2018)) and their corresponding intents. On the right hand, we show the corresponding *SimMatrix* (before we apply PCA). The resulting matrix is a symmetric matrix with all its diagonal values = 1, representing maximum similarity. Then we use PCA to reduce the number of columns, loosely speaking, creating a set of representative utterances in a data-driven manner. At this point, we use each row as the vector representation of the corresponding utterance. Thus, each representative utterance serves as a dimension in the vector space, allowing utterances with similar neighbors to have similar vector representations.

This approach has a number of advantages over conventional vector representations (such as BOW) and LM based techniques (such as Para2Vec) in the domain of conversational text understanding. The data collected by (Braun et al., 2017) shows that conversational text tends to be very short with an average of 7.8 tokens per utterance. Moreover, 80% of the collected utterances are shorter than 9 tokens. This makes BOW representation very sparse. Also for LM-based techniques, it is hard to learn efficient vector representations because of the shortness of the context sequences used for training. Another advantage of SIMVECS is the easier detection of utterances that have no intent (i.e. the "None" intent utterances). As shown Fig. 1, the "None" intent utterances are expected to have similarity scores close to zero to

all other utterances, including other "None" utterances. This makes them located closer to the origin in SIMVECS's vector space and allows distance-based clustering techniques (such as K-means) to group "None" intent utterances in the same cluster. The second advantage is representative vector lengths: instead of using vectors of arbitrary lengths or length equal to the vocabulary size, we use vectors of length equal to the number of representative utterances in the given application. This can reduce the size of the generated vector representation significantly, especially when large vocabularies are used, while the LU application itself may have only a few tens or hundreds of utterances.

5 Unsupervised learning with conversational text

The problem of applying unsupervised learning techniques to text documents has been studied by many researchers in several domains such as (Beil et al., 2002),(Aggarwal and Zhai, 2012), and (Huang, 2008). The problem becomes more challenging with conversational text because of the shortness and the sparsity of the documents (Chen et al., 2011). We can categorize these techniques based on the input they need to perform clustering into two categories: 1) Techniques that require a vector representation for the data points, such as K-means and SVD. 2) Techniques that require a similarity (distance) function such as Affinity-Propagation (Frey and Dueck, 2007) and DBSCAN (Ester et al., 1996). However, the second category still requires an efficient vector representation to estimate the distances between pairs of utterances. We evaluate the efficacy of SIMVECS generated vectors in the unsupervised learning task against several baselines. We vary the vector representation while the clustering algorithm itself (K-means) remains the same. We show a comparison against the following techniques:

Spherical K-means (Buchta et al., 2012): This baseline uses BOW representation for the utterances and cosine-similarity as a distance function.

LDA (Blei et al., 2003): This baseline represents documents as probability distributions over latent topics. Documents with similar topic assignments are grouped together. Similar to K-means, it takes the number of latent topics (clusters) as an input. We set the number of topics to the number of intents in the corpus and then assign each utterance to the cluster with the maximum probability.

LDA + K-means: Here we use LDA's latent topic distribution as a vector representation to the utterance. Then we apply K-means for clustering the utterances.

Seq-Auto-encoder + K-means (Dai and Le, 2015): A Neural-network based technique using recurrent language models. The network is trained on a large corpus of conversational text generated from the same Cortana corpus used for generating the IDF scores. Afterwards, the network is used to encode each utterance of the test data in a vector of length 1024. K-means is then used to cluster the utterances.

Fast-text + K-means (Joulin et al., 2016): Also a Neural-network based technique that incorporates several features such as BOW and N-gram features. The model generates word-level vectors which are averaged together to form sentence-level representations. We used English pre-trained word vectors[4]. These vectors are pre-trained with 16 billion tokens collected from *Wikipedia 2017, UMBC webbase* corpus and *statmt.org* news.

SIMVECS+ K-means: K-means applied to our similarity-based vectors.

To make the comparison fair, K-means algorithm with the same number of clusters (K*) is used for the all techniques (Except LDA). Here K* is the number of intents in the corpora. For LDA, we set number of latent topics to be also K*.

Corpora	# Intents	# Utterances	# Tokens
AskUbuntu	5	162	1289
Chatbot	2	200	1539
WebApp	8	89	717
Combined	15	451	3545

Table 2: Conversational text corpora details.

5.1 Dataset description

We use the conversational text dataset presented in (Braun et al., 2017)[5]. The dataset represents a collection of three corpora, two corpora were extracted from StackExchange (Ask Ubuntu & WebApp), while the third one was extracted from a Telegram chatbot. *Combined* is a corpus that combines the intents of all three. Table 2 shows the number of intents, number of utterances, and number of tokens for each corpus in the dataset.

[4]https://fasttext.cc/docs/en/english-vectors.html

[5]The dataset is publicly available and can be obtained here: https://github.com/sebischair/NLU-Evaluation-Corpora

Utterance ID	Intent	Utterance
U1	AddToPlaylist	add the current tune, to my, Rock Gaming, playlist
U2	AddToPlaylist	add villotta, to The MetalSucks Playlist, playlist
U3	SearchCreative Work	Please look up the Atheist Manifesto: The Case Against Christianity, album, .
U4	SearchCreative Work	Please look up the painting, Beyond Iconic: Photographer Dennis Stock, .
U5	BookRestaurant	book a spot for 8, at The Kitchin, on october the 13th, 2039,
U6	BookRestaurant	Book a reservation for 8, at a restaurant, that serves chicken fried bacon, in Aruba,
U7	RateBook	Give the current, book, im reading zero, points, out of 6,
U8	RateBook	rate the current, book, three, out of 6,
U9	None	PayPal
U10	None	OkCupid

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}
U_1	1	0.3	0.03	0.03	0.03	0.02	0.1	0.12	0	0
U_2	0.3	1	0.03	0.03	0.03	0.01	0.03	0.04	0	0
U_3	0.03	0.03	1	0.23	0.03	0.01	0.03	0.03	0	0
U_4	0.03	0.03	0.23	1	0.03	0.01	0.03	0.03	0	0
U_5	0.03	0.03	0.03	0.03	1	0.19	0.08	0.1	0	0
U_6	0.02	0.01	0.01	0.01	0.19	1	0.06	0.07	0	0
U_7	0.1	0.03	0.03	0.03	0.08	0.06	1	0.41	0	0
U_8	0.12	0.04	0.03	0.03	0.1	0.07	0.41	1	0	0
U_9	0	0	0	0	0	0	0	0	1	0.01
U_{10}	0	0	0	0	0	0	0	0	0.01	1

Figure 1: An example showing extracting the vector representation for each utterance. The table on the left shows 10 utterances (examples) for 5 different intents, whereas the table on the right shows the corresponding SimMatrix for the 10 utterances. Each row (or column) is then used as the vector representation for the corresponding utterance

5.2 Unsupervised learning results

In this section, we show the efficacy of SIMVECS's vector representation and compare to other baseline techniques. We collected both purity and normalized mutual information (NMI) scores for the generated clusters. We omit NMI scores for space reasons as it shows the same trend as purity. As shown in table 3, SIMVECS shows better performance (in terms of Purity) in comparison to other baseline except in one corpors, "Chatbot". Our solution (SIMVECS+ K-means) outperforms all baselines by at least 10% on average. Fast-text, Seq-Auto-Encoder, and BOW have very similar performance. We also notice that LDA+K-means outperforms LDA by 9%, suggesting that using topic distributions as vector representations is more efficient than mapping the utterance to the topic with the maximum corresponding probability. We also notice that Fast-text outperforms all other techniques (including SIMVECS) in one corpus (Chatbot). The reason is that Fast-text through training on billions of words, can find similarities between words that belong to the same domain. For example: the vector representation for the words "Pizza" and "Burger" have a cosine-similarity of 0.63 using Fast-text vectors as both words are very frequent in the "Restaurants" domain. Although this might be useful in some situations, it can be very misleading in others, causing non-similar utterances to have high silimarity scores. For instance, a particular restaurant might only serve burgers while any pizza orders are not supported and therefore should be considered outliers (i.e. "None" intent). This means the similarity between the two words will cause an overlap between these two intents ("Order" and "None"), driving the clustering algorithm to mistakenly combine the two intents into one cluster. This behavior is highlighted in the poor performance of Fast-text in all other corpora compared to SIMVECS.

6 Semi-Supervised learning with conversational text

In this section, we evaluate the efficacy of SIMVECS vector representations in semi-supervised learning tasks. One of the main requirements in language understanding services is to improve their performance during operation using active learning techniques. This is achieved by adapting to different user perspectives in different domains. Moreover, it is hard in many cases to perform clustering efficiently without taking the user perspective into consideration. For instance, Fig. 2 shows an example of two groups of utterances that can be clustered at different levels of granularity based on the user's needs. In many such cases, the same set of utterances can be clustered in different ways according to different

Corpora	Combined	AskUbuntu	Chatbot	WebApplications	Average
Spherical Kmeans	61%	63%	61%	64%	62%
LDA	46%	59%	61%	35%	50%
LDA+Kmeans	55%	67%	62%	54%	59%
Seq-Auto-encoder+Kmeans	65%	64%	62%	57%	62%
Fast-Text+Kmeans	53%	56%	**94%**	49%	63%
SIMVECS+Kmeans	**71%**	**83%**	61%	**76%**	**73%**

Table 3: Purity scores for SIMVECS vs several baselines

criteria. Moreover, it is not known which criteria is to be used by the clustering algorithm without user feedback. Therefore, semi-supervised learning is typically used in directing SIMVECS to the correct context-sensitive clustering. In this section, we evaluate the ability of SIMVECS in a semi-supervised learning mode, compared to several baselines.

In semi-supervised clustering, few data points are labeled and used as constraints to the clustering technique. These constraints are of the form **"Must-Link"** and **"Cannot-Link"** for pairs of data points. A "Must-Link" constraint arises when the user indicates that two utterances belong to the same intent, while a "Cannot-Link" constraint is when to the user indicates two utterances as belonging to different intents. We propose a simple algorithm to refine SIMVECS vectors based on the provided constraints: Whenever a "Must-Link" constraint is provided for a pair of points, we collapse their 2 vectors into one vector in the space. This is achieved by taking the max value of each entry of the corresponding indexes. Thus, the resulting vector is closer to neighbors of both points, shrinking the distances between neighbors of both utterances.Moreover, because each feature (dimension) in the space is a representative utterance, after the collapsing of two utterances, we perform PCA to come up with the new dimensions after the user is done with the labeling.

On the other hand, whenever a "Cannot-Link" constraint is provided, the similarity score between the two utterances is set to zero in the corresponding entry in SimMatrix. This increases the distance between the two points and transitively, between the neighbors of these two points.
To evaluate the effieciency of the resulting vectore, we vary the amount of user-labeled data points and estimate the corresponding clustering purity scores. As shown in Fig. 3, semi-supervised learning can significantly improve the performance of the clustering algorithm. We compare the performance of SIMVECS to BOW, Seq-Auto-encoder, and Fast-text. We use constrained K-means (COP-Kmeans) (Wagstaff et al., 2001) as the semi-supervised learner for SIMVECS as well as all baseline techniques. We see that SIMVECS achieves its maximum gain against other techniques when the proportion of labeled data is small. This is very useful in our problem as it reduces the labeling effort required from the user side to assist the clustering algorithm. The results show improvements over all three baselines, while the difference between the approaches shrinks with more labeled data points as expected. Also we notice that with few labeled data points (from 10% to 30%), Seq-Auto-encoder and Fast-text representations are performing better than BOW. However, with more labeled data ($\geq 50\%$) BOW starts to perform better. The reason is that with more labeled data points, the training examples (constraints) start to cover most of the expressions that can be used for a particular intent.

7 Supervised learning with conversational text

Supervised learning is critical to language understanding services in order to identify both intents and entities in new utterances coming in the stream. We compare the performance of intent classifiers when SIMVECS is used against BOW, Seq-auto-encoders, and Fast-text vector representations. For all corpora, a linear SVM classifier is trained per intent in one-vs-all fashion. For each intent, the utterances that belong to that intent represent the positive class, whereas all other utterances represent the negative class. We apply 5-fold cross validation and calculate the average F1-Score across all runs. Fig. 4 shows the improvement in F1-Score with both variants over baseline

Get Weather	
Is Warm	**Weather Update**
1 Will it get warmer, in Holy Cross Wilderness	4 What's the weather in Waretown, Lebanon
2 Will it be warmer, in five years, in Slemp, Kansas	5 What's the weather like in Saint Regis Falls, ND
3 Will it be warm, in Powersville, Guam, 23 hours from now	6 Tell me the weather forecast for Carmichaels, , Gambia, at one am

Figure 2: Example of clustering with different levels of granularity. Without user feedback, it is not clear whether the clustering algorithm should put all the utterances in the same cluster (Get Weather) or split them into two separate clusters (Is Worm & Weather Update).

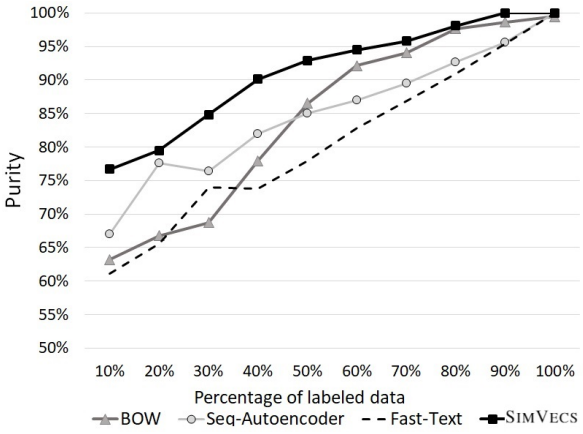

Figure 3: Improvement on clusters Purity using SIMVECS with COP-Kmeans vs several baselines

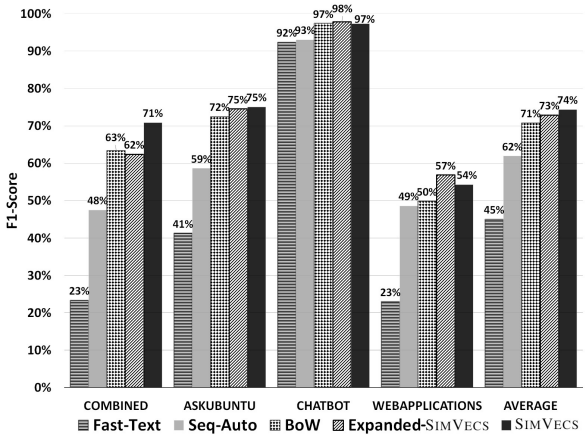

Figure 4: Improvement on classification accuracy using SIMVECS vs several baselines.

techniques. An improvement of 29% is observed over Fast-text representation, whereas an improvement of 13% over Seq-Auto-encoder representation is observed. SIMVECS is only slightly better than BOW (3% improvement on average). Additionally, we introduce a variant of SIMVECS that doesn't use the automatically tuned weights shown in Eq. 4. Instead, we concatenate all 6 similarity sub-metrics with all other utterances into one vector and use the resulting vector as the utterance representation (called Expanded-SIMVECS). Notice that this approach generates vector representations of size 6X compared to SIMVECS. We notice that using SIMVECS with our pre-trained weights is achieving better results than Expanded-SIMVECS across all corpora. The reason is that as Expanded-SIMVECS increases the number of dimensions, it also increases the sparsity of the space and hence requires more training data, which is known as "the curse of dimensionality" (Poggio et al., 2017). We also notice that the performance gain is proportional to the number of intents in the corpus. The peak gains of 8%, 25%, and 48% over BOW, Seq-Auto-encoder, and Fast-text respectively are observed with "Combined" corpora (which combines the 15 intents in all three corpora).

8 Background

In this section, we give the formulas used to estimate our evaluation metrics: Purity (Manning et al.) and F1-Score (Sasaki et al., 2007).

8.1 Purity:

is evaluated by the given formula:

$$Purity = \frac{1}{N}\sum_{i=1}^{k} Max_j |C_i \cap t_j| \quad (6)$$

Where N is the number of data points, k is the number of clusters, C_i is a generated cluster, and t_j is the intent which represents the majority in C_i.

8.2 F1-Score:

is evaluated by the given formula:

$$F_1 = 2.\frac{P * R}{P + R} \quad (7)$$

Where P is the precision, and R is the recall.

9 Discussion

One of the practical challenges in implementing SIMVECS can be the size of the SimMatrix, particularly when the number of utterances grows very large. Currently, this is not be a critical issue for current LU services as they tend to limit the number of utterances per application to a few thousands. But the maximium number of supported utterances is expected to grow in the future. For example, IBM Watson currently limits the number of utterances to 25,000 per workspace (application) whereas Microsoft LUIS limits the number of utterances to 15,000 per application. One approach to improve the scalability of SIMVECS is by constraining further the number of dimensions of the vector space thus reducing the memory requirements for storing SimMatrix. For reducing the computational time, multi-threading can be used to calculate similarity scores between different pairs of utterances concurrently and hence speedup the matrix construction process.

10 Conclusion

This paper introduces SIMVECS, a similarity-based vector representation technique designed to overcome prior work limitations in the field of conversational AI. We discussed the main challenges in vector representation for conversational AI applications and how SIMVECS overcomes these challenges. Through evaluation on different corpora and for different learning tasks, we showed the efficacy of vector representations generated by SIMVECS over several baselines.

References

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008a. The evaluation of sentence similarity measures. In *International Conference on data warehousing and knowledge discovery*, pages 305–316. Springer.

Palakorn Achananuparp, Xiaohua Hu, Xiaohua Zhou, and Xiaodan Zhang. 2008b. Utilizing sentence similarity and question type similarity to response to similar questions in knowledge-sharing community. In *Proceedings of QAWeb 2008 Workshop, Beijing, China*, volume 214.

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer.

Florian Beil, Martin Ester, and Xiaowei Xu. 2002. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Martin Boyanov, Ivan Koychev, Preslav Nakov, Alessandro Moschitti, and Giovanni Da San Martino. 2017. Building chatbots from forum data: Model selection using question answering metrics. *arXiv preprint arXiv:1710.00689*.

Daniel Braun, Adrian Hernandez-Mendez, Florian Matthes, and Manfred Langen. 2017. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185.

Christian Buchta, Martin Kober, Ingo Feinerer, and Kurt Hornik. 2012. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22.

Mengen Chen, Xiaoming Jin, and Dou Shen. 2011. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):35.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science*, 315(5814):972–976.

Frauke Friedrichs and Christian Igel. 2005. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

HK Lam, SH Ling, Frank HF Leung, and Peter Kwong-Shun Tam. 2001. Tuning of the structure and parameters of neural network using an improved genetic algorithm. In *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, volume 1, pages 25–30. IEEE.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Yuhua Li, David McLean, Zuhair A Bandar, Keeley Crockett, et al. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge & Data Engineering*, (8):1138–1150.

Ashraf Mahgoub, Paul Wood, Sachandhan Ganesh, Subrata Mitra, Wolfgang Gerlach, Travis Harrison, Folker Meyer, Ananth Grama, Saurabh Bagchi, and Somali Chaterji. 2017. Rafiki: a middleware for parameter tuning of nosql datastores for dynamic metagenomics workloads. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, pages 28–40. ACM.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze. Introduction to information retrieval? cambridge university press 2008. *Ch*, 20:405–416.

Paul Mcnamee and James Mayfield. 2004. Character n-gram tokenization for european language text retrieval. *Information retrieval*, 7(1-2):73–97.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Melanie Mitchell. 1996. An introduction to genetic algorithms mit press. *Cambridge, Massachusetts. London, England*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. 2017. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519.

Yutaka Sasaki et al. 2007. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5.

Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. 2001. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Mohamed A Zahran, Ahmed Magooda, Ashraf Y Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. 2015. Word representations in vector space and their applications for arabic. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 430–443. Springer.