# Using Paraphrasing and Memory-Augmented Models to Combat Data Sparsity in Question Interpretation with a Virtual Patient Dialogue System

**Lifeng Jin,[1] David King,[1] Amad Hussein,[2] Michael White,[1]** and **Douglas Danforth[3]**
[1]Department of Linguistics, [2]Department of Computer Science and Engineering,
[3]Department of Family Medicine
The Ohio State University, Columbus, OH, USA
{jin, king, mwhite}@ling.osu.edu
amadh881@gmail.com, doug.danforth@osumc.edu

## Abstract

When interpreting questions in a virtual patient dialogue system, one must inevitably tackle the challenge of a long tail of relatively infrequently asked questions. To make progress on this challenge, we investigate the use of paraphrasing for data augmentation and neural memory-based classification, finding that the two methods work best in combination. In particular, we find that the neural memory-based approach not only outperforms a straight CNN classifier on low frequency questions, but also takes better advantage of the augmented data created by paraphrasing, together yielding a nearly 10% absolute improvement in accuracy on the least frequently asked questions.

## 1 Introduction

To develop skills such as taking a patient history and developing a differential diagnosis, medical students interact with actors who play the part of a patient with a specific medical history and pathology, known as Standardized Patients (SPs). Although SPs remain the standard way to test medical students on such skills, SPs are expensive and can behave inconsistently from student to student. A virtual patient dialogue system aims to overcome these issues as well as provide a means of supplying automated feedback on the quality of the medical student's interaction with the patient (see Figure 1).

In previous work, Danforth et al. (2009, 2013); Maicher et al. (2017) used a hand-crafted pattern-matching system called ChatScript together with a 3D avatar in order to collect chatted dialogues and provide useful student feedback (Danforth et al., 2016). ChatScript matches input text using hand-written patterns and outputs a scripted response for each dialogue turn. With sufficient pattern-writing skill and effort, pattern matching with ChatScript can achieve relatively high accuracy, but it is unable to easily leverage increasing amounts of training data, somewhat brittle regarding misspellings, and can be difficult to maintain as new questions and patterns are added.

To address these issues, Jin et al. (2017) developed an ensemble of word- and character-based convolutional neural networks (CNNs) for question identification in the system that attained 79% accuracy, comparable to the hand-crafted ChatScript patterns. Moreover, they found that since the CNN ensemble's error profile was very different from the pattern-based approach, combining the two systems yielded a nearly 10% boost in system accuracy and an error reduction of 47% in comparison to using ChatScript alone. Perhaps not surprisingly, the CNN-based classifier outperformed the pattern-matching system on frequently asked questions, but on the least frequently asked questions—where data sparsity was an issue—the CNN performed much worse, only achieving 46.5% accuracy on the quintile of questions asked least often.

In this paper, we aim to combat this data sparsity issue by investigating (1) whether paraphrasing can be used to create novel synthetic training items, examining in particular lexical substitution from several resources (Miller, 1995; Le and Mikolov, 2014; Ganitkevitch et al., 2013; Cocos and Callison-Burch, 2016) and neural MT for back-translation (Mallinson et al., 2017); and (2) whether neural memory-based approaches developed for one-shot learning (Kaiser et al., 2017) perform better on low-frequency questions. We find that the two methods work best in combination, as the neural memory-based approach not only outperforms the straight CNN classifier on low frequency questions, but also takes better advantage of the augmented data created by paraphrasing. Together, the two methods yield nearly

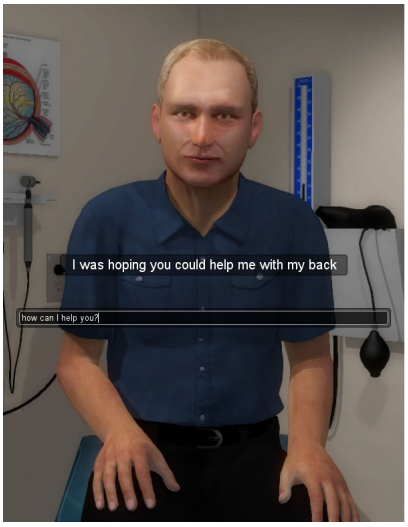



Figure 1: Virtual Patient Dialogue System

a 10% absolute improvement in accuracy on the quintile of least frequently asked questions.

## 2 Related Work

Question identification is a task that can be approached in at least two ways. One way is to treat it as a multiclass classification problem (e.g., using logistic regression), which can take advantage of class-specific features but tends to require a substantial amount of training data for each class. Formally, letting $q$ be the candidate question, $Y$ be a set of question classes and $\phi$ a feature extractor, we seek to find the most likely label $\hat{y}$:

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} \frac{e^{\phi(q,y)}}{\sum_{y' \in Y} e^{\phi(q,y')}}.$$

Alternatively, a pairwise setup can be used. For example, for each class a binary classification decision can be made as to whether a given question represents a paraphrase of a member of the class, choosing the highest confidence match. More generally, let $q_i^y \in L^y$ be the $i$-th question variant for label $y$ (where the question variants are the paraphrases of the label appearing in the training data); given some similarity metric $\sigma$, we seek to find the label $\hat{y}$ with the most similar question variant $q_i^{\hat{y}}$ in the set $L^{\hat{y}}$ to the candidate question $q$:

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}} \max_{q_i^y \in L^y} \sigma(q, q_i^y)$$

Early work on question answering (Ravichandran et al., 2003) found that treating the task as a maximum entropy re-ranking problem outperformed using the same system as a multiclass classifier. By contrast, DeVault et al. (2011) observed that maximum entropy multiclass classifiers performed well with simple n-gram features when each class had a sufficient number of training examples. Jaffe et al. (2015) explored a log-linear pairwise ranking model for question identification in a virtual patient dialogue system and found it outperformed a multiclass baseline along the lines of DeVault et al. (2011). However, Jaffe et al. used a much smaller dataset with only about 915 user turns, less than one-fourth as many as in the current dataset. For this larger dataset, a straightforward logistic regression multiclass classifier outperforms a pairwise ranking model.

In general it appears reasonable to expect that the comparative effectiveness of multiclass vs. pairwise approaches depends on the amount of training data, and that pairwise ranking methods have potential advantages for cross-domain and one-shot learning tasks (Vinyals et al., 2016; Kaiser et al., 2017) where data is sparse or nonexistent. Notably, in the closely related task of short-answer scoring, Sakaguchi et al. (2015) found that pairwise methods could be effectively combined with regression-based approaches to improve performance in sparse-data cases.

Other work involving dialogue utterance classification has traditionally required a large amount of data. For example, Suendermann-Oeft et al. (2009) acquired 500,000 dialogues with over 2 million utterances, observing that statistical systems outperform rule-based ones as the amount of data increases. Crowdsourcing for collecting additional dialogues (Ramanarayanan et al., 2017) could alleviate data sparsity problems for rare categories by providing additional training examples, but this technique is limited to more general domains that do not require special training/skills. In the current medical domain, workers on common crowdsourcing platforms are unlikely to have the expertise required to take a patient's medical history in a natural way, so any data collected with this method would likely suffer quality issues and fail to generalize to real medical student dialogues. Rossen and Lok (2012) have developed an approach for collecting dialogue data for virtual patient systems, but their approach does not directly address the issue that even as the number of dialogues collected increases, there can remain a long

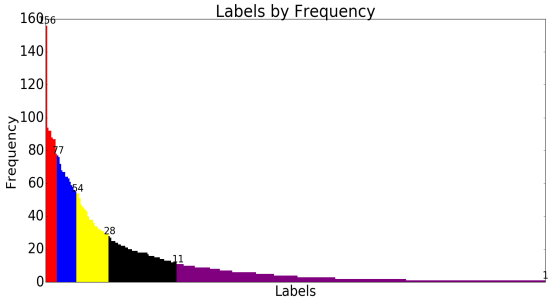



Figure 2: Label frequency distribution is extremely long-tailed, with few frequent labels and many infrequent labels. Values are shown above quintile boundaries.

tail of relevant but infrequently asked questions.

As an alternative to crowdsourcing, we pursue paraphrasing for data augmentation in this paper, focusing on the simplest methods to employ, namely lexical substitution and neural back-translation (see Section 5). The idea is to augment the observed question instances for questions with infrequent labels in the dataset with automatically generated paraphrases, with the aim of making such questions easier to recognize using machine-learned models. In future work, we plan to explore more complex paraphrasing methods, including syntactic paraphrasing (Duan et al., 2016) and inducing paraphrase templates from aligned paraphrases (Fader et al., 2013).

## 3 Data Imbalance

Our dataset currently consists of 4330 question-answer pairs from 94 dialogues between first year medical students and the virtual patient. After classifying an asked question as having a certain label, the virtual patient replies with the canned response for that label, as illustrated in Table 1. Unfortunately, the labels do not have a uniform distribution with regards to the number of variants each label has (that is, the number of question instances for that label in the dataset). In fact, most of the labels are underrepresented.

On average, each question label has 12 variants, but 8 labels account for nearly 20% of the data, while 256 labels account for the bottom 20% (Figure 2). We define a rare label to be any label that is in that set of 256 infrequent labels. Supplementing the data to account for this imbalance is the primary focus of our work.

## 4 Memory-Augmented CNN Classifier

Because of the data sparsity issue, we cast the problem of sentence classification for infrequent labels as a problem of few-shot learning. In particular, we use Kaiser et al.'s (2017) memory module together with a CNN encoder (Kim, 2014; Jin et al., 2017) as our main model, the memory-augmented CNN classifier (MA-CNN). Our aim is to take advantage of the MA-CNN's one-shot learning capability to mitigate the issue of data sparsity and also to make better use of data augmentation to achieve better performance.

### 4.1 The CNN encoder

The CNN encoder follows Kim (2014) and Jin et al. (2017). We briefly summarize the architecture here and direct interested readers to these two papers for implementation details. There are four layers in the encoder: an embedding layer, a convolution layer, a max-pooling layer and a linear layer. Let $\mathbf{x}_i \in \mathbb{R}^k$ be a $k$-dimensional embedding for the $i$-th element of the sentence $s$. We concatenate all of the element embeddings to get $\mathbf{S} \in \mathbb{R}^{|s| \times k}$ as the representation of the whole sentence.

The convolution layer may have many kernels, which are defined as weight matrices $\mathbf{w}_j \in \mathbb{R}^{hk}$, where $h$ is the width of the kernel. They slide across the sentence representation and then pass through a nonlinearity to produce a feature map $\mathbf{c}_j \in \mathbb{R}^{|s|-h+1}$. Then the max-pooling layer uses max-over-time pooling (Collobert et al., 2011) on the feature maps to ensure fixed-dimensional outputs.

Finally, we concatenate all the outputs from all the kernels into a single vector $\mathbf{o}$, multiply it with the weight matrix $\mathbf{W}_l$ and apply $p2$-normalization to it as the final fully-connected neural network layer for the CNN encoder:

$$\mathbf{e} = \frac{\mathbf{o} \cdot \mathbf{W}_l + \mathbf{b}_l}{\|\mathbf{o} \cdot \mathbf{W}_l + \mathbf{b}_l\|} \quad (1)$$

Here $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and the bias term for the final layer, respectively.

### 4.2 The memory module

We follow Kaiser et al. (2017) for implementation of our memory module. The memory module is a tuple of three matrices $\mathbf{K}$, $\mathbf{V}$ and $\mathbf{A}$, which stores

| Student question | Label detected | Canned response |
|---|---|---|
| hello mr. wilkins | hello mr | hello doctor. i am so glad to see you. |
| can you tell me a little about your issue | <None> | i'm sorry, i don't understand that question. would you restate it? |
| what brings you in today | what brings you in today | i was hoping you could help me with my back pain, it really hurts! it has been awful. |

Table 1: Sample interactions between a first year medical student and the virtual patient. The virtual patient's task is to accurately detect the kind of question the medical student is asking and then reply with the appropriate canned response.

one key, one label and one age of one memory entry in each corresponding row. A key is an encoded presentation of a training item, a label is the class identifier that the key belongs to, and the age is the number of memory updates that have taken place since the key was inserted or updated. To use the memory, a normalized query item $\mathbf{q}$ is multiplied by the key matrix

$$\mathbf{s}^\top = \mathbf{q} \cdot \mathbf{K} \tag{2}$$

to yield a vector of cosine similarities $\mathbf{s}$ between the query and every entry in the memory. The prediction made by the memory is then $\hat{v} = \mathbf{V}[\hat{n}]$, where $\hat{n} = \text{argmax}(\mathbf{s})$ and $\hat{v}$ is the predicted class label.

The memory operations include insert, update and erase, and loss calculation of the memory depends on the memory operations, therefore we briefly summarize them here. Let $\hat{n}$ be the row index in $\mathbf{s}$ with the highest similarity score such that $\mathbf{V}[\hat{n}]$ is the true label of the query, $\tilde{n}$ be the row index of the entry with the highest similarity score that has a different label from the true label, and $v$ be the true label. When $\mathbf{s}[\hat{n}] > \mathbf{s}[\tilde{n}]$, the memory loss is a margin loss between the similarity scores at $\hat{n}$ and at $\tilde{n}$ with some margin $\alpha$:

$$\text{loss} = [\mathbf{s}[\tilde{n}] - \mathbf{s}[\hat{n}] + \alpha]_+ \tag{3}$$

In this case, the memory entry at $\hat{n}$ will be updated by replacing it with the normalized average of itself and the query:

$$\mathbf{K}[\hat{n}] \leftarrow \frac{\mathbf{q} + \mathbf{K}[\hat{n}]}{\|\mathbf{q} + \mathbf{K}[\hat{n}]\|} \tag{4}$$

When $\mathbf{s}[\hat{n}] < \mathbf{s}[\tilde{n}]$, the memory loss is:

$$\text{loss} = [\mathbf{s}[\hat{n}] - \mathbf{s}[\tilde{n}] + \alpha]_+ \tag{5}$$

In this case, a new entry is inserted at a previously empty row $n'$:

$$\mathbf{K}[n'] \leftarrow \mathbf{q} \quad \mathbf{V}[n'] \leftarrow v \tag{6}$$

In both cases, the entry in $\mathbf{A}$ at the update or insert site will be replaced by 0, and all the other entries in $\mathbf{A}$ will add 1. When the memory is full, a new insertion will take place where $A[n']$ is the biggest.

Finally, if there is no entry in $\mathbf{K}$ that has the true label $v$, the insert operation is carried out without any loss calculation. The erase operation is to reset all three matrices to empty, which is used at the end of a training episode.

### 4.3 Episodic training and evaluation

We train our memory-augmented CNN classifier using a novel episodic training scheme based on the episodic training scheme used in one-shot learning (Vinyals et al., 2016; Kaiser et al., 2017). The main difference is that in one-shot learning, most tasks offer a balanced dataset with many classes but small numbers of instances per class. In our scenario, the dataset is imbalanced, and some classes may have a large number of instances. Moreover, in evaluation, there are no unseen classes in our case. We modify the episodic training scheme to accommodate these differences.

#### Episodic training

In training, we define an episode to be a complete $k$-shot learning trial with gradient updates. At the beginning of each episode, a batch of $|C| \times (k+1)$ samples, where $|C|$ is the number of classes, is sampled from the training data. The first sample of each class is then encoded and inserted into the memory with no loss calculated, which we call loading the memory. From the second sample on,

the encoder encodes each sample, and the memory calculates its loss according to its prediction. After all classes have had one sample to complete this process, the encoder is updated by the gradients calculated with the memory loss. The memory is then updated according to the operations corresponding to its predictions of the seen samples in each shot. When all $k$ shots have been processed, the memory is completely erased ready for the next episode (though naturally the updates to the encoder remain in effect).

It is easy to see that this process involves oversampling, which is a known technique for rebalancing imbalanced datasets. Because each class must have $k + 1$ samples for each episode, the minority classes have to be oversampled. However, experiments show that oversampling itself does not lead to better performance.

**Episodic evaluation**

In evaluation, we define a support set to be a batch of $|C| \times k$ samples from the training data. For a given test set, we first load the memory, then compare each test item to all the entries in the memory in order to generate the memory prediction for the test item based on the most similar memory entry. This forms the model's 1-shot predictions. Then we update the memory with the second sample for each class and redo the prediction step. We now have the model predictions with 2 shots. We continue to follow this routine until predictions from all $k$ shots have been collected.

Because there is some randomness in how a support set is sampled from the data, we use multiple support sets in evaluation. Since some of the classes have a large number of instances, each randomly sampled support set tends to be sufficiently different from other support sets that using multiple support sets becomes analogous to ensembling different models.

Finally, letting $p$ be the number of support sets, we have $k \times p$ predicted labels for each item in the test set. We use majority voting across all the predicted labels to get the final model prediction. This capitalizes on the ensembled support sets and reduces the variance of the model predictions.

## 5 Data Augmentation

Since previous work (Jin et al., 2017) showed that the majority of labels in our dataset have 11 variants or fewer, we explore using lexical substitution (McCarthy and Navigli, 2009) and neural machine translation (NMT) back-translation (Mallinson et al., 2017) for data augmentation. The main difference in our use of lexical substitution and previous works' is that our setup is unsupervised, as we have no gold test set for determining acceptable paraphrases. Similarly for the NMT system, we do not know which outputs are acceptable. To mediate this, we employ the use of both human and automatic filtering of the generated paraphrases with the end-goal of facilitating question label identification for infrequent labels.

### 5.1 Paraphrase generation

We exploit advances in lexical substitution and NMT to automatically produce paraphrases. We also combine these approaches to determine their collective effectiveness in our downstream label identification task.

**Lexical substitution**

Lexical substitution has often been held up as a exemplary task for paraphrase generation. In its simplest form, one must simply replace a given word with an appropriate paraphrase, i.e. one that retains most of the original sentence's meaning. As an example, in the question *have you ever been seriously ill?*, *seriously* could be replaced with *severely*, and we would consider this to be an appropriate substitution. However, if we instead substituted *solemnly* for the same word, we would not accept this as the meaning would have deviated too far.

For generating paraphrases, we employ three resources: WordNet (Miller, 1995), Word2Vec (Le and Mikolov, 2014), and paraphrase clusters from Cocos and Callison-Burch (2016). To evaluate these resources, we took the mean average precision (MAP) of a given resource's ability to produce a lexical substitution which matched a word that already existed in another variant for the same label. That is, if the label *how has the pain affected your work?* had only two variants, *has the injury made your job difficult?* and *is it hard for you to do your job?*, and a resource successfully produces the swap of *hard* → *difficult* (producing the sentence *is it difficult for you to do your job?*), this would positively affect a resource's MAP score. We only performed this evaluation on labels with 30 or more variants as this form of evaluation disproportionately penalizes labels with fewer variants.

These preliminary experiments indicated that

pooling candidates from all three resources performed better than any given one alone did. We also found that in the case of multiple word senses (e.g. *bug* meaning an insect, an illness, or a flaw in a program), simply picking the first sense produced a higher MAP score than a variety of other selection algorithms. This is not surprising since, in the case of WordNet, the first synset is the most frequently used sense of a given word. For Cocos and Callison-Burch's semantic clusters, these were ordered by a given cluster's average mutual paraphrase score as annotated in the Paraphrase Database (Ganitkevitch et al., 2013). Although our domain is medical, the dialogues are patient directed, less technical, and more colloquial, allowing us to use such a simple selection method for word sense disambiguation.

For augmenting the data in a way that would help the most sparse labels, we focused our lexical substitution task on labels with less that 11 variants. After pooling all the lexical substitution candidates from each resource, we ranked the substitutions by subtracting the original sentence's $n$-gram log probability from its paraphrase's.[1] We then extracted the top 100 scoring paraphrases for our initial unfiltered data set.

#### Neural machine translation

We additionally use Neural Machine Translation (NMT) to generate paraphrases by pivoting between languages. In multiple back-translation, a method developed in Mallinson et al. (2017), we take a given English source sentence and generate $n$-best translations into a pivot language. This is the forward step. For each pivot translation we generate an $m$-best list of translations back into English. Thus this backward step yields $n \times m$ paraphrases for a given source sentence, where each paraphrase within this final set has a weight based on which of the original $n$ translations it came from in the forward step and its ranking among the $m$ translations in the back step. Any duplicates within this final set are collapsed and their weights are combined before the set is ranked according to weight. This method favors translations which come from high quality sources (high-ranking translations in the lists $n$ and $m$) as well as translations which occur multiple times.

In our work we translated each given source sentence into 10-best forward translations and 10-best back translations before finally collapsing and ranking the 100 paraphrases. We used a model from Sennrich et al. (2016) and chose German as our pivot language given the quality of the translations and paraphrases we observed.[2]

### 5.2 Filtering

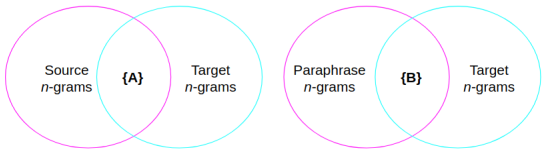
Since both the lexical substitution and NMT methods generate helpful and unhelpful paraphrases, we needed a way to select useful paraphrases. Although a typical next step might be to manually filter each system's output by hand, we were unsure if expensive human filtering would produce any gain in downstream performance. To explore this question, we experimented with a fully automatic *pseudo*-oracle.

The pseudo-oracle is an automatic filter which we designed to look at a particular test item in a cross-validation setup and select the paraphrases whose $n$-gram recall with that test item was higher than the original source sentence's, as illustrated in Figure 3. In using this initial step of filtering, we are able to isolate the paraphrases which are most likely to be helpful for classifying question labels. In preliminary experiments using logistic regression, we tested the performance of the pseudo-oracle selection process on the downstream classification task, where we found that the pseudo-oracle was able to facilitate classifying question labels, whereas using all the outputs from the lexical substitution and NMT paraphrase generations systems (without filtering) led to a drop in performance.

Thus, to lessen the expense of human filtering, we used the pseudo-oracle as an automated first step, under the assumption that the selected paraphrases would mostly be kept as well using manual filtering. Next, using the same Gigaword trained language model from Section 5.1, we ranked the lexical substitution and NMT outputs. From these ranked lists, we extracted the highest scoring subsets such that each paraphrase not only had a high log probability, but also contributed a unique $n$-gram (i.e., if two paraphrases contributed the same new $n$-gram, only the highest scoring paraphrase was selected). This diversity-enhancing filtering reduced the size of the dataset

[1] We used a 5-gram language model with back off, trained on the Gigaword (Parker et al., 2011).

[2] We found that the pretrained model for German produced the best back-translations when compared to other pretrained models. In future work, we plan to train our own models across various pivot languages to produce an increased variety of paraphrases.




$$\frac{|\{A\}|}{|\{Target\}|} < \frac{|\{B\}|}{|\{Target\}|}$$

Figure 3: A graphical representation of the pseudo-oracle selection process. For a given test item (here *Target*), the *n*-gram overlap with the paraphrase must be greater than the overlap with the source sentence that paraphrase was derived from.

to around 20% of the original raw lexical substitution output and 2.5% of the raw NMT output, greatly lessening human annotation costs.

Since we instructed the annotators (a subset of the authors) to only select useful paraphrases which contributed novel *n*-grams not present in any other variant, their task was necessarily different from the pseudo-oracle's. Annotators required 16 hours per annotator to manually filter the data. We found that the annotators selected paraphrases which might not necessarily help the downstream task in a cross-validation setup, but which could be expected to help with completely unseen data. For this reason, we chose to combine the pre-selected paraphrases chosen by the pseudo-oracle together with the human-filtered paraphrases in our evaluation.

## 6 Experiments

We use the best model in Jin et al. (2017), namely a stacked convolutional neural network (Stacked-CNN), together with the model proposed in this work (MA-CNN) in all of the experiments. Our task is to accurately predict a question's label based solely on the typed input from the medical student. With improved accuracy, the virtual patient will be able to more coherently answer the students' questions.

We shuffle the gold dataset first and use 10-fold cross-validation to evaluate our data augmentation process. We specifically focus our analysis on rare labels since that is also where we concentrate our data augmentation efforts. The model we propose here is targeted at improving performance for the rare labels, therefore we are interested in how the model performs on them. Paraphrases are not added to test sets, and paraphrases derived from those test items are filtered from training. Finally, we compute significance using the McNemar test (McNemar, 1947).

### 6.1 Hyperparameters

We mostly follow Jin et al. (2017) in setting the hyperparameters of the CNN encoder in MA-CNN. We only use word-based features in the encoder. Following Jin et al. (2017), we set the number of kernels of the encoder of MA-CNN to be 300. We use kernels of widths 3 to 5 for the CNN encoder. All non-linearities in the models are rectified linear units Nair and Hinton (2010). We use Adadelta (Zeiler, 2012) as the optimizer for the whole MA-CNN, and use the recommended values for its hyperparameters ($\rho = 0.9, \epsilon = 1 \times 10^{-6}, learning\_rate = 1.0$). We initialize the embeddings with Word2Vec but allow them to be tuned by the system (Mikolov et al., 2013).

For episodic training, we set the number of shots to be 10. For the episodic evaluation, we use 5 support sets. For each support set, we also do 10-shot evaluation. Therefore for each test item, there are 50 predictions in total. We combine all predictions with majority voting, weighted by the similarity score of each prediction.

### 6.2 MA-CNN on rare labels

We first train our model MA-CNN and the stacked CNN model from Jin et al. (2017) using just the original VP dataset and explore how the model architecture affects rare label accuracy. Table 2 shows the test accuracy for both models. MA-CNN performs very well on the rare labels. The performance difference between the stacked CNN model and MA-CNN is highly significant, which shows that the pairwise-classification approach paired with episodic training is really powerful on the items which belong to labels with few training instances. We can also see that MA-CNN does not perform as well as the CNN ensemble on all labels, which is consistent with the previous observation that non-pairwise classifiers work better when training data is large. It is worth noting though that the stacked CNN ensemble consists of 10 CNNs that take in word- and character-based features as their inputs, meanwhile the encoder of the MA-CNN is just a single word-based CNN. This further illustrates how a pairwise system which is designed specifically for dealing with classes with few training instances can help im-

| System | Full Acc | Rare Acc |
|---|---|---|
| StackedCNN | 79.02 | 46.54 |
| MA-CNN | 75.22 | **51.78***** |

Table 2: Test results for the stacked CNN ensemble (Jin et al., 2017) and the memory-augmented CNN classifier (MA-CNN) without any generated paraphrases. The difference of performance on the rare items is highly significant ($p = 9.5 \times 10^{-5}$, *McNemar's test*).

| System | Full Acc | Rare Acc |
|---|---|---|
| StackedCNN | 78.45 | 53.04 |
| MA-CNN | 75.33 | **56.14***** |

Table 3: Test results for the stacked CNN ensemble and the memory-augmented CNN classifier (MA-CNN) with the manually filtered paraphrases. The gain brought by the adding the automatically generated paraphrases into training data for MA-CNN is highly significant ($p = 1.6 \times 10^{-4}$, *McNemar's test*).

prove performance on those classes by using nearest neighbor comparison and episodic training inspired by one-shot learning.

### 6.3 Generated paraphrases as training data

We further explore the effect on model performance of using the generated paraphrases along with the gold training data in training. We use the manually filtered dataset with both paraphrasing methods, and train both the stacked CNN ensemble and MA-CNN with it plus the gold set. Table 3 shows the results on the test set. First, we can see that both models benefit in terms of rare label accuracy by using the augmented dataset. The difference between MA-CNN trained with only the gold dataset and the augmented dataset is highly significant, showing that the generated paraphrases are of high quality and help MA-CNN to achieve even better performance on the rare labels. It is interesting to note that for full accuracy, performance of both models does not significantly change, showing that the paraphrases are of high enough quality to not be harmful to the frequent labels.

### 6.4 Effects of data augmentation

Table 4 shows the effect of using pseudo-oracle and manually filtered data on rare labels. We find that the MA-CNN is able to use the data augmentation in a way that directly benefits the rare labels. Specifically, the MA-CNN benefits from the human filtered data, indicating that it benefits from information provided to it that raw $n$-gram overlap

| System | Rare Acc |
|---|---|
| Pseudo-oracle | 54.87 |
| Manual | **56.14** |

Table 4: Test results for the memory-augmented CNN classifier (MA-CNN) with different filtering techniques.

| Paraphrases | Rare Acc |
|---|---|
| No paraphrases | 51.78 |
| Lexical substitution | 53.16 |
| Neural Machine Translation | 55.22 |
| Both | **56.14** |

Table 5: Test results for the memory-augmented CNN classifier (MA-CNN) with different subsets of the manual filtered paraphrases generated using different paraphrase methods.

does not capture. At the same time, however, filtering using the pseudo-oracle evidently provides a reasonable approximation of what improvements in accuracy can be obtained with human filtering of the generated paraphrases.

### 6.5 Quality of generated paraphrases

We also want to see how the performance on rare labels is connected to the method with which the paraphrases are generated. We use the individual subsets each of which is generated by a single method to augment the training data. Table 5 shows how these methods compete against each other. Surprisingly, simple lexical substitution is already good at providing information that is helpful to MA-CNN, but the neural machine back translation is an even better method at providing paraphrases that have positive impact on rare label accuracy. We inspect the paraphrases generated by both methods and find that paraphrases from back translation are generally more diverse in phrasal structure and contain more novel words than those generated with lexical substitution. The combined dataset gives further improvement, showing that lexical substitution and neural machine translation are at least partially complementary to each other as generation methods.

### 6.6 Combining the stacked CNN and the MA-CNN

Given the fact that the MA-CNN performs very well on rare labels, but not so well on all labels, it is interesting to see if a combined system with the stacked CNN and MA-CNN can provide

| System | Full Acc | Rare Acc |
|---|---|---|
| StackedCNN | 79.02 | 46.54 |
| MA-CNN | 75.33 | **56.14** |
| Combiner | **79.86***** | 50.98 |

Table 6: Test results for the combiner as well as the two combined subsystems: the stacked CNN ensemble trained with gold and the memory-augmented CNN classifier trained with gold and generated paraphrases. The gain on full accuracy is highly significant ($p = 1.9 \times 10^{-9}$, *McNemar's test*).

a further performance increase. We here choose a relatively simple logistic regression model as our model combiner, though a more sophisticated model could be used in principle. Using 1-5 grams of words and stemmed words as well as 2-5 grams of characters, we trained the model to predict the rarity of a label for a question, i.e. if a candidate question belongs to a rare label or not. This rarity predictor gets 94.2% accuracy on all labels, and 78.1% accuracy on rare labels. Note that the majority baseline for all labels is 80%, but for rare labels it is 20%. This rarity predictor serves as our combiner; that is, we use the combiner to choose whose result to trust between the two classification systems. If the combiner predicts that an item belongs to a rare label, we choose the prediction from the MA-CNN; if the combiner instead predicts it belongs to a frequent label, we choose the prediction for it from the stacked CNN. This is done with 10-fold cross validation, just like how the classifiers were trained above.

The stacked CNN model we use here is the one trained with only gold training data, which is the model with the best accuracy on all labels. We use the MA-CNN model trained with both gold and generated data. With the combiner, we get 50.98% accuracy on rare labels, and 79.86% accuracy on all labels, as shown in Table 6. The result indicates that the two systems are complementary to each other, and simple combination is already effective in providing a significant performance boost. Although the accuracy on rare labels is not as high as the MA-CNN by itself, it is higher than the stacked CNN model by 5 points, and all of these points are translated into an accuracy increase on all labels that is close to 1 point.

## 7 Conclusion

In this paper, we have investigated the use of paraphrasing for data augmentation and neural memory-based classification in order to tackle the challenge of a long tail of relatively infrequently asked questions in a virtual patient dialogue system. We find that both lexical substitution and neural back-translation yield paraphrases of observed questions that improve system performance on rare labels once the generated paraphrases are manually filtered down to ones taken to be useful, with neural back-translation contributing more to gains in accuracy than lexical substitution. We also find that neural memory-based classification with a novel method of episodic training outperforms a straight CNN classifier on low frequency questions and takes better advantage of the generated paraphrases, together yielding a nearly 10% absolute improvement in accuracy on the least frequently asked questions. Finally, using a simple logistic regression model to combine the predictions of the straight CNN and memory-based classifier, we find that the combined system performs better on all labels, and the gain is from more accurate predictions of rare labels. We expect these gains to yield increased user engagement and ultimately better learning outcomes. In future work, we plan to investigate using the memory-based classifier for fully automatic paraphrase filtering as well as more advanced methods of paraphrasing, including deep generative paraphrasing, syntactic paraphrasing and using aligned paraphrases to induce paraphrase templates. More powerful models may also be explored to better combine the models.

## Acknowledgments

Thanks to Kellen Maicher for creating the virtual environment and to Evan Jaffe, Eric Fosler-Lussier and William Schuler for feedback and discussion. This project was supported by funding from the Department of Health and Human Services Health Resources and Services Administration (HRSA D56HP020687), the National Board of Medical Examiners Edward J. Stemmler Education Research Fund (NBME 1112-064), and the National Science Foundation (NSF IIS 1618336). The project does not necessarily reflect NBME policy, and NBME support provides no official endorsement. We thank Ohio Supercomputer Center (1987) for computation support.

## References


Anne Cocos and Chris Callison-Burch. 2016. Clustering paraphrases by word sense. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1463–1472.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Douglas Danforth, A. Price, Kellen Maicher, D. Post, Beth Liston, Daniel Clinchot, Cynthia Ledford, D. Way, and Holly Cronau. 2013. Can virtual standardized patients be used to assess communication skills in medical students. In *Proceedings of the 17th Annual IAMSE Meeting, St. Andrews, Scotland*.

Douglas Danforth, Mike Procter, Richard Chen, Mary Johnson, and Robert Heller. 2009. Development of virtual patient simulations for medical education. *Journal For Virtual Worlds Research*, 2(2).

Douglas Danforth, Laura Zimmerman, Kellen Maicher, Holly Cronau, Cynthia Ledford, D. Post, Allison Macerollo, D. Way, and Beth Liston. 2016. Virtual standardized patients can accurately assess information gathering skills in medical students. In *Proceedings of the American Association of Medical Colleges*, Seattle, WA.

David DeVault, Anton Leuski, and Kenji Sagae. 2011. An evaluation of alternative strategies for implementing dialogue policies using statistical classification and rules. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1341–1345.

Manjuan Duan, Ethan Hill, and Michael White. 2016. Generating disambiguating paraphrases for structurally ambiguous sentences. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 160–170, Berlin, Germany. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

Evan Jaffe, Michael White, William Schuler, Eric Fosler-Lussier, Alex Rosenfeld, and Douglas Danforth. 2015. Interpreting questions with a log-linear ranking model in a virtual patient dialogue system. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 86–96.

Lifeng Jin, Michael White, Evan Jaffe, Laura Zimmerman, and Douglas Danforth. 2017. Combining CNNs and Pattern Matching for Question Interpretation in a Virtual Patient Dialogue System. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 11–21.

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to Remember Rare Events. In *Proceedings of the International Conference on Learning Representations*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

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

Kellen Maicher, Douglas Danforth, A. Price, Laura Zimmerman, B. Wilcox, Beth Liston, Holly Cronau, Laurie Belknap, Cynthia Ledford, D. Way, D. Post, Allison Macerollo, and Milisa Rizer. 2017. Developing a conversational virtual standardized patient to enable students to practice history taking skills. *Simulation in Healthcare*, 12(2):124–131.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 881–893.

Diana McCarthy and Roberto Navigli. 2009. The english lexical substitution task. *Language resources and evaluation*, 43(2):139–159.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations*, pages 1–12.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, 3, pages 807–814.

The Ohio Supercomputer Center. 1987. Ohio Supercomputer Center.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword. *Linguistic Data Consortium*.

Vikram Ramanarayanan, David Suendermann-Oeft, Hillary Molloy, Eugene Tsuprun, Patrick Lange, and Keelan Evanini. 2017. Crowdsourcing multimodal dialog interactions: Lessons learned from the HALEF case. In *Proceedings of the AAAI-17 Workshop on Crowdsourcing, Deep Learning, and Artificial Intelligence Agents*, pages 423–431.

Deepak Ravichandran, Eduard Hovy, and Franz Josef Och. 2003. Statistical qa-classifier vs. re-ranker: What's the difference? In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 69–75. Association for Computational Linguistics.

Brent Rossen and Benjamin Lok. 2012. A crowdsourcing method to develop virtual human conversational agents. *International Journal of HCS*, 70(4):301–319.

Keisuke Sakaguchi, Michael Heilman, and Nitin Madnani. 2015. Effective feature integration for automated short answer scoring. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1049–1054. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

David Suendermann-Oeft, Keelan Evanini, Jackson Liscombe, Phillip Hunter, Krishna Dayanidhi, and Roberto Pieraccini. 2009. From rule-based to statistical grammars: Continuous improvement of large-scale spoken dialog systems. pages 4713–4716.

Oriol Vinyals, Charles Blundell, Timothy Lillcrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. In *Proceedings of Neural Information Processing Systems*, pages 817–825.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR*.