DSTC9 task 2

Divy Bramhecha, Michael Heck, Milica Gasic
[†] DSML, HHU

Abstract. Attempted solution for the second part of the DSTC9 track 1.

Introduction

"Most virtual assistants can help users to book a hotel, a restaurant or movie tickets, they fall short of answering potential follow-up questions users may have, for example: how to park vehicles; whether they are allowed to bring pets or children to the reserved place; or what the cancellation policy is. No API/DB entry is usually available to handle such requests. On the other hand, relevant domain knowledge is already available on web pages in the form of descriptions, FAQs and customer reviews for many of these out-of-scope scenarios." (Kim et al., 2020)

The above problem is part of the The Ninth Dialog System Technology Challenge (DSTC9) as proposed in (Kim et al., 2020). The challenge track addresses task-oriented conversations grounded on fine-grained domain/entity-level knowledge related to given dialogue contexts. In addition, the track includes evaluation of participant model submissions on generalization to new, unseen domains in the training data set, as well as generalization to unseen modalities (i.e., moving from written to spoken conversations).

According to the challenge, the pipeline is divided into 3 parts:

(1) Identify the out of ontology turn. (Knowledge-seeking turn detection)
(2) Successfully find the correct knowledge snippet from the entire knowledge base (Knowledge Selection).
(3) Generate an appropriate response, continuing the dialog smoothly. (Response Generation)

This work focuses on the (2) Knowledge Selection task.

[*]s1885517@ed.ac.uk

1. Problem definition

Assuming that there exists $\mathbf{f_1}(\mathbf{U_t}|\mathbf{K})$, where $\mathbf{U_t} = \{\mathbf{u_1}, .., \mathbf{u_{t-1}}, \mathbf{u_t}\}$ is the dialog context and $\mathbf{K} = \{\mathbf{k_1}, .., \mathbf{k_n}\}$ is the knowledge, is used to determine a user turn as knowledge-seeking. Once the function $\mathbf{f_1}$ determines a given turn $\mathbf{t}$ as a knowledge-seeking turn we can move forward with the Knowledge Selection task to sort out the relevant knowledge snippets. This task takes each pair of $\mathbf{U_t}$ and $\mathbf{k_i} \in \mathbf{K}$ and predicts whether they are relevant or not as follows:

$$f_2(U_t, k_i) = \begin{cases} 1 & \text{if } k_i \in K \text{ is relevant to } U_t, \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

Also, note in the database provided as part of the challenge, a given turn $\mathbf{t}$ has exactly 1 relevant knowledge snippet and so the f_2 can be written as:

$$f_2(U_t, K) = k_i \tag{1.2}$$

where $i \in$ relevant knowledge.

2. An information retrieval task!

2.1. Baseline model. The baseline model presented in the challenge employed BERT-Based model. They trained a binary classification model which takes concatenation of the utterances in U_t and the knowledge k_i as an input instance. Using the final layer output at [CLS] token position to obtain a probability s_i that k_i is relevant to the given dialog context U_i. The model was fine tuned using binary cross-entropy loss as L:

$$L = -\sum_{i \in I_{pos}} \log s_i - \sum_{i \in I_{neg}} \log 1 - s_i \tag{2.1}$$

where "pos" and "neg" refer to the relevant and irrelevant knowledge snippet respectively. They found 5 negative candidates for each positive candidate worked the best. (Kim et al., 2020)

2.2. Our model. The Knowledge Selection task can be treated as an information retrieval task where the dialog (between the user and the system) is a query used to rank the entire knowledge database based on their relevance. An important aspect of ranking algorithms is that they need to be fast if they are ever meant to be deployed. For instance, the baseline model takes 11 hours for prediction on the validation set with a NVIDIA Tesla V100. Moreover, it is also crucial that the model can rank the knowledge database quickly in order to pick negative samples effectively. For the above reasons we need a model to perform large scale semantic similarity, and Sentence-Bert (Reimers and Gurevych, 2019) achieves this.

For us the idea is a function $\mathbf{S_{sentence\_vector}}(\text{input\_text}) = \tilde{\mathbf{v}}$ such that

$$f_2(U_t, K) = \arg\min_{k_i} \{\|S(U_t) - S(k_i)\| | k_i \in K\} \tag{2.2}$$

f_2 returns the most relevant knowledge snippet.

In BERT $\mathbf{S_{sentence\_vector}}$, the pooling strategy is to use the representation from the CLS-token. Reimers and Gurevych, 2019, in S-BERT investigate various pooling strategies to conclude that mean-pooling on BERT outputs results in the best pooling strategy. Their

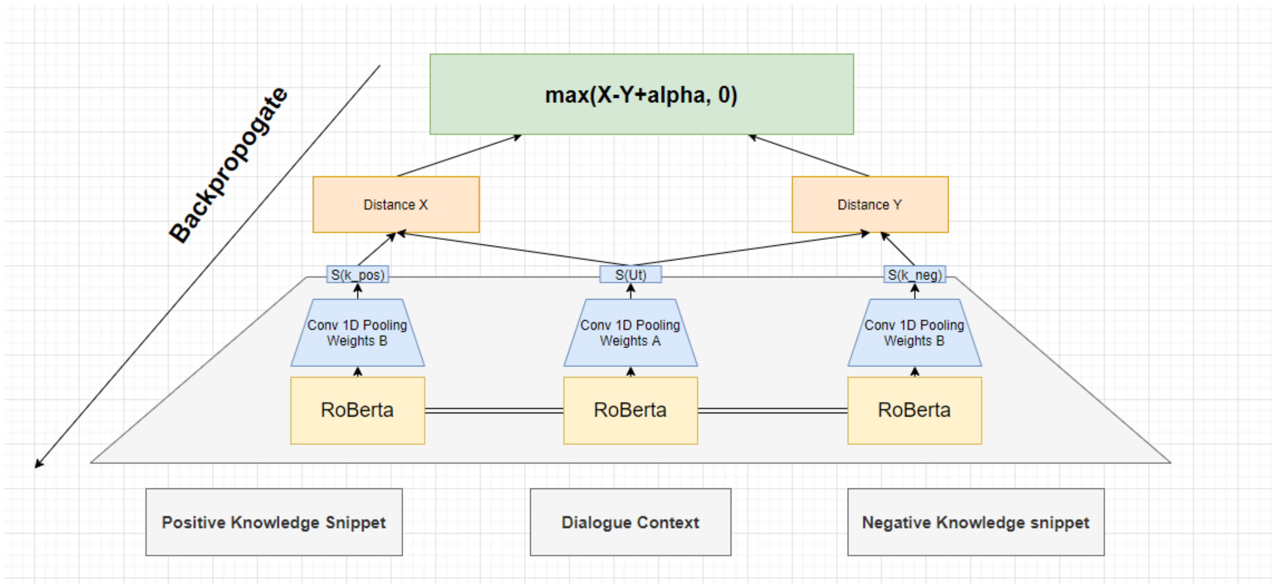
max(X-Y+alpha, 0)
Backpropogate
Distance X
Distance Y
S(k_pos)
S(Ut)
S(k_neg)
Conv 1D Pooling Weights B
Conv 1D Pooling Weights A
Conv 1D Pooling Weights B
RoBerta
RoBerta
RoBerta
Positive Knowledge Snippet
Dialogue Context
Negative Knowledge snippet

Figure 1. Simplified model architecture during training

experiments agree with the values we obtained on DSTC9 data set as well. We further investigate pooling strategies as in our case encoding a dialog context vs encoding a knowledge snippet must have some differences, like the amount of context present. We do this by choosing two separate pooling functions, one for dialog context and another for knowledge snippets. The pooling function we choose is a 1d convolution layer on top of the RoBERTa outputs. This 1d convolution layer acts like a weighted sum over the sequence output of the final encoder layer. There are 2 set of weights trained for 1d convolution layer, first for dialog context and second for knowledge snippet. A note, we conducted experiments with both BERT and RoBERTa, however, RoBERTa consistently outperforms BERT based models.

In order to fine-tune the RoBERTa model we pick the best performing strategy from SBERT (Reimers and Gurevych, 2019) which is to create Siamese and triplet networks (Schroff et al., 2015) to update the weights such that the produced sentence embeddings are semantically meaningful and can be compared with distance measure. Figure 1 captures the model structure.

The **triplet objective** function for our use is:

$$\max(\|S(U_t) - S(k_{pos})\| - \|S(U_t) - S(k_{neg})\| + \epsilon, 0) \tag{2.3}$$

Where $S(U_t)$ is sentence embedding for the dialog context, $S(k_{pos})$ is sentence embedding for relevant(positive) knowledge snippet, and $S(k_{neg})$ for negative knowledge snippet. Margin ϵ ensures that $S(k_{pos})$ is at least ϵ closer to $S(U_t)$ than $S(k_{neg})$ is.

3. Negative sampling

The negative samples we choose for training any model for the Knowledge Selection task is very crucial!

3.1. **Baseline sampling used.** In the baseline model they find that using 5 negative samples for every 1 positive sample worked the best. They had 3 sampling strategies, "All" where they randomly sample 5 knowledge snippets from the entire database. "Oracle" where they randomly sample 5 knowledge snippets from the correct entity ID. "Mix" where they combine the above two strategies. "Mix" was their default negative sampling strategy.

3.2. **Distance based sampling strategies.** One of the major benefits of having a model that can represent the knowledge snippets in vector form with fast compute time is that it allows us to pick negative samples by constantly comparing the $S(U_t)$ with the $S(k_{neg})$. There were various strategies we experimented with, the "mix" sampling from baseline, hard sampling, semi hard sampling (Schroff et al., 2015), Distance weighted sampling (Wu et al., 2018).

Hard sampling: In this method we choose the k_{neg} such that

$$\arg\min_{k_i} \{\|S(U_t) - S(k_i)\| | k_i \in K_{neg}\} \tag{3.1}$$

This method works quite badly, as the high difficulty of the negative samples proves to be impossible for the model to learn the differences and results in the collapsed models.

Semi-hard sampling: In this method we choose the k_{neg} such that

$$\arg\min_{k_i} \{\|S(U_t) - S(k_{pos})\| - \|S(U_t) - S(k_i)\| + \alpha | k_i \in K_{neg}\} \tag{3.2}$$

This method works quite well in practise however, it offers a very small increase in performance to "mix" sampling. That means it probably can be improved further.

3.3. **Distance weighted sampling.** To find a better sampling method we must investigate the "mix" sampling method. Recall that our embeddings are typically constrained to the n-dimensional unit sphere S^{n-1} for large $n \geq 128$. Consider the situation where the points are uniformly distributed on the sphere (Wu et al., 2018). In this case, the distribution of pairwise distances follows:

$$q(d) \propto d^{n-2}[1 - \frac{1}{4}d^2]^{\frac{n-3}{2}} \tag{3.3}$$

For higher dimensions $q(d)$ approaches $\mathcal{N}(\sqrt{2}, \frac{1}{2n})$ In other words, if negative examples are scattered uniformly, and we sample them randomly we are likely to obtain examples that are $\sqrt{2}$-away. For thresholds less than $\sqrt{2}$, this induces no loss, and thus no progress for learning. Learned embeddings follow a very similar distribution, and thus the same reasoning applies. (Wu et al., 2018) for these reasons developed distance based sampling which is adapted for our use case as shown below:

$$Pr(K_{neg} = k_i | U_t) \propto \min(\lambda, q^{-1}(\|S(U_t) - S(k_i)\|)) \tag{3.4}$$

The q^{-1} is Distance weighted sampling offers a wide range of examples during training, and thus steadily produces informative negative examples while controlling the variance.

4. Pre-training task

After doing error analysis on models up to this point we realised that the model fails to recognise the difference between similar meaning phrases, specially phrases like "Senior discount" vs "children discount" or "cost of WiFi" vs "is WiFi Available". More importantly, it wasn't affecting just the R@1 score but also R@5 score which was surprising. Hence, we decided to fine tune Roberta on MRPC-dataset. The fine tuning for RoBERTa was done in the same manner as in Devlin et al., 2019. As a part of MRPC challenge, the model learns the equivalence relationship between two sentences. The dataset's diversity and the similarity of the task to ours was the key reason why we chose it as a pretraining task.

5. Experiments and results

Note: A correct knowledge snippet k_i is represented by three components: domain name, entity ID and document ID. Training the model to identify all three works quite bad in practise, and we'll discuss this aspect more in detail later. All the experiments reported below are performed assuming the correct domain, and entity ID. Therefore the task is to find the correct the document ID.

DSTC9 validation set task2 (given domain name, entity ID)			
Model	R@1	MRR@5	R@5
Baseline	93.6%	NaN	NaN
SBERT mean pool + mix sample	91.5%	93.2%	97.4%
SRoBERTa mean pool + mix sample	92.6%	94.5%	97.6%
SRoBERTa mean pool + Semi-Hard sampling	93.2%	NaN	NaN
SRoBERTa mean pool + distance weighted sampling	96.3 %	97.2%	98.6%
SRoBERTa mean pool + distance weighted sampling + MRPC pretrained	97.1 %	97.9%	99.5%
SRoBERTa Conv1d pooling + distance weighted sampling + MRPC pretrained	**98.3%**	**99.2%**	**99.7%**

As a part of the challenge, we expect unseen domains during test time, so the design decisions also considered that. We took 1 domain out of the training and tested on all of the domains for R@1. The model used for this was the best performing model from previous experiment.

Domain independence experiment R@1 values					
Not trained on	Total	Hotel	Restaurant	Train	Taxi
Hotel	95.4%	**88.3%**	99.3%	94.2%	99.7%
Restaurant	95.3%	94.4%	**90.4%**	97.6%	99.7%
Train	92.6%	92.5%	97.2%	**87.3%**	93.5%
Taxi	97.2%	95.3%	98.4%	95.8%	**99.2%**
Nothing	98.3%	99.3%	97.6%	97.1%	99.7%

However, when the model is not provided with the correct domain and entity id, the performance drops very heavily. This is not surprising as the model has no explicit supervision in identifying various names in order to differentiate between k_i.

DSTC9 Validation set Task2 (end to end)			
Model	R@1	MRR@5	R@5
Baseline	67.2%	78.4%	92.9%
SRoBERTa Conv1d pooling + distance weighted sampling + MRPC pretrained	**11.2%**	24.6%	42.3%

6. Entity Selection Model

Due to results above, we further divide the knowledge selection task into 3 parts.

(1) Out of ontology domain detection
(2) Entity ID detection
(3) Document ID detection

The Document ID detection task is solved exactly as described in previous section. Next we tackle the Entity Id detection task. Note: Entity Id detection essentially refers to finding the correct name (eg: hotel name) that the question is referring to.

One way to address the problem of extracting the entity is to perform dialog state tracking with objectives coming from both in-ontology and out of ontology dialogue turns. In this way one can utilise more training data and produce a single model for both of these tasks. More importantly, one can consider value independent dialogue state trackers such as TripPy for better generalization capabilities. While this makes sense in practice, in this particular dataset the out-of-ontology questions were artificially inserted in an existing data set. This meant that the name that the user referred to most recently is not necessarily the one for which an out-of-ontology request has been issued. This makes sense to us humans, as more questions can pop up in our head at later stage in the conversation, however as the span based dialog state trackers have never been trained with such a dataset they fail to perform well. Moreover, training these models require precise spans labels which the DSTC9 dataset doesn't provide. Lastly, models trained for span prediction don't provide any confidence scores, hence ranking a database becomes a bigger challenge. To address these issues we propose another model for finding the appropriate name, i.e. its entity id.

6.1. **Sequence Tagging.**

Input to RoBerta consists of tokens. Consider a simple training task of identifying all hotel names in a sequence given that these input tokens are also labeled as hotel name (1) or not a hotel name (0). Then we can use the sequence output of RoBerta, which consists of feature vectors for each input token as an input to a sigmoid layer and use binary cross-entropy loss function for training. Methods similar to this have been a popular choice to do some sort of span prediction on input sequences, however it has few important drawbacks.

Due to binary cross-entropy as the loss function, each token is either classified as a 1 (hotel name) or 0 (not a hotel name) as in figure 2. In the following sentence:

how far is the hotel marriott?

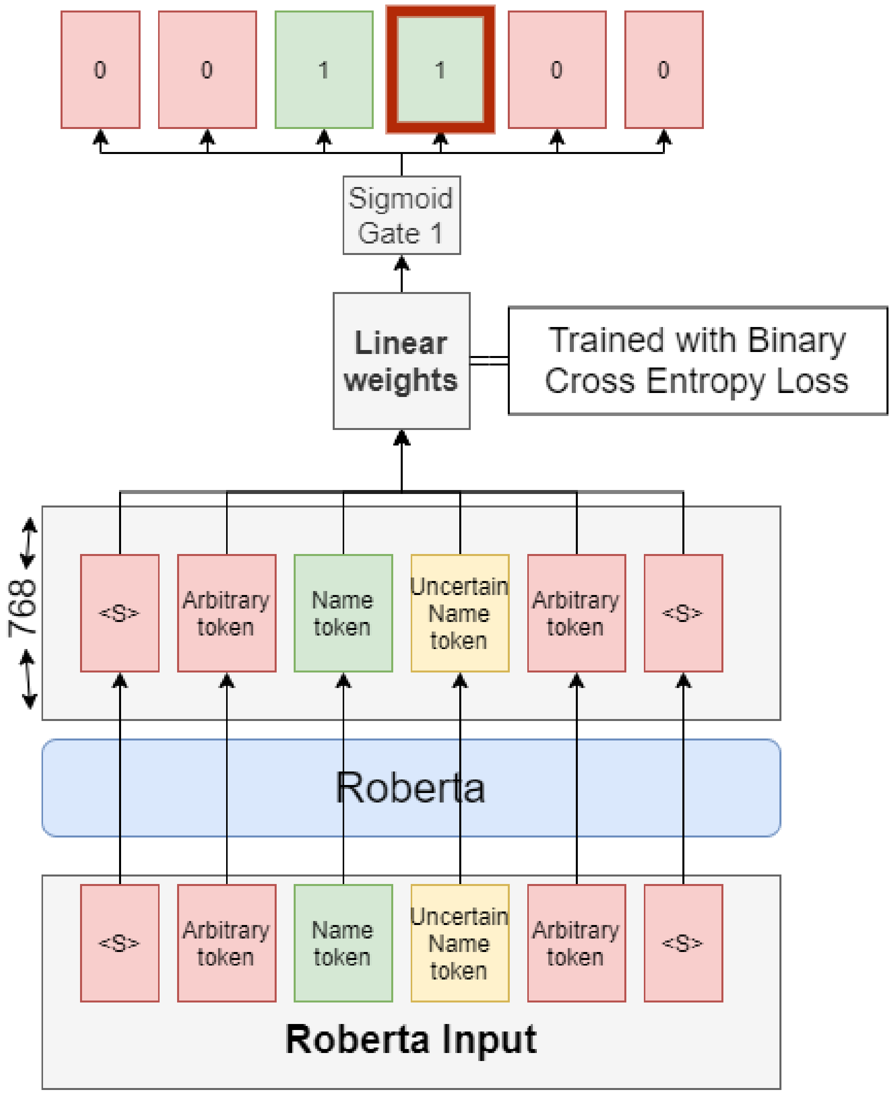

Figure 2. Sequence Tagging. Confidently classifying an uncertain token.

Most models trained in this fashion will classify "marriott" as 1. However, it's tricky to classify the word "hotel". Some might argue that it's part of the name, some others that it is not, but most of us will be uncertain. However, the output of this model will always be either a 1 or a 0 depending on the labels in the training dataset. This problem becomes much more apparent when the model is trying to identify unseen names and tokens which it often ends up either misclassifying, or only partly identifying. Moreover, this also shows how sensitive it is to the labels in the training dataset, hence annotating such datasets becomes non-trivial.

One of the methods to tackle the problem is to use Label smoothing cross entropy loss function instead, which replaces one hot targets of cross entropy with smooth target distribution (Szegedy et al., 2016). However, this method too suffers the same problem where the model is explicitly given a target value of an uncertain event.

6.2. **Database entry.** Moreover, our task is to find out the correct entity_id and not the correct name tokens. A traditional approach has been to do span prediction on user text,

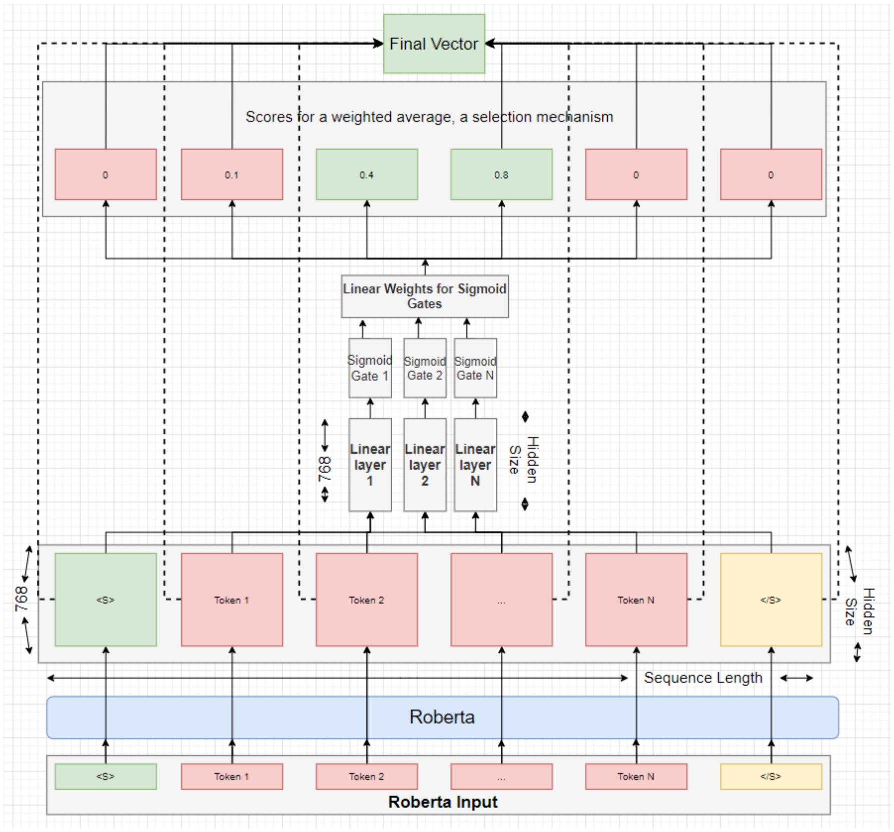

Figure 3. simplified pooling mechanism architecture. The goal is that the model weighs tokens based on how important they might be in the comparison to make. I.e. in this case, all the tokens referring to names have a high value, and tokens specifically referred to in the user's question (positive entity id) have the highest values!

extract the key words relating to the query and do some text processing, like minimum edit distance, to find the relevant name in the database. However, in practise we see that users often don't precisely mention the names, and sometimes the span prediction mechanism itself makes a mistake. These errors propagate quickly.

6.3. **Our model.** Hence, we propose a model that provides confidence scores in its prediction of the token labels without explicitly training on them. Instead of reverting back to the discrete space of words when these tokens have been identified, it generates a vector for the entire utterance that only represents a specific relevant information like names. It still relies on the same Siamese network archtiture as shown in figure 1, however the conv1d pooling is replaced by the various linear layers as shown in figure 3.

Here, the Positive knowledge snippet is replaced by the "positive name", i.e. the out of ontology name being referred to, and the negative knowledge snippet is replaced by "negative name" which refers to any other name in the database. Similar to figure 2, the sequence output undergoes a sigmoid layer, however as there is no cross-entropy loss function here the outputs lie between 0 and 1. These are treated as confidence scores as seen in figure 3.

6.4. **Training and pre-training.** The process of creating vector in Figure 3, is the same for the positive name, negative name and the dialog. As there is no explicit training on span labels, randomly initializing the weights for the linear layers leads to a collapsed model.

We solve this problem by adding a pre-training step, where each of the linear layers are trained identical to the sequence tagging model. After completion of this step our model is identical to that of a sequence tagging model, hence suffers the same problems. However, now we can continue training with triplet loss function where there is no explicit reference to span labels. This allows the model to automatically smooth out it's confidence scores on each word, and also allows it the ability to use tokens which it would previously have incorrectly identified. Most importantly, in the pre-training step we don't require very accurate span labels, hence training on dataset like DSTC9 becomes possible.

For the entity prediction part of the challenge, two linear layers were pretrained. One to find all the names present in the input, and another to specifically find the out of ontology request name. This was followed by training in the Siamese architecture fashion with triplet loss function. During prediction these dialogues and database entries can be quickly converted to vectors representation which are then compared to find out the most similar ones.

7. Experiments and Results

Note: The experiment reported below assumes we already know the correct domain for the out of ontology question.

DSTC9 Validation set Task2, given domain find entity id			
Model	R@1	R@2	R@3
Baseline	?%	?%	?%
Our model	**98.1%**	**99.1%**	**99.2%**

As a part of the challenge, we expect unseen domains during test time, so the design decisions also considered that. We took 1 domain out of the training and tested on all of the domains for R@1. The model used for this was the best performing model from previous experiment.

Domain independence experiment R@1 values	
Not trained on	Total
Hotel	96.1%
Restaurant	94.7%

8. Out of ontology domain detection

Lastly, we must find the domain of the out of ontology requests.

RoBERTa was finetuned using a standard binary cross entropy loss function. It took the dialog sequence along with the domain description as input, and the label was 1 if it was the correct domain, 0 otherwise. Examples of the domain description are "Is the question about hotel domain?", "Is the question about train domain?". The exactness of these questions is arbitrary as a lot of variants seem to work the same in practise.

9. Experiments and Results

Note: The experiment reported below assumes perfect input from task 1, i.e. out ontology request detection. Recall represents that the correct domain is predicted as 1. Accuracy represents the correct domain is predicted as 1 and incorrect domains are predicted 0.

DSTC9 Validation set Task2, find the out of ontology requests' domain		
Model	Accuracy	Recall
Baseline	?%	?%
Our model	**99.1%**	**99.8%**

As a part of the challenge, we expect unseen domains during test time, so the design decisions also considered that. We took 1 domain out of the training and tested on all of the domains for accuracy/recall. The model used for this was the best performing model from previous experiment.

Domain independence experiment accuracy%/recall% values					
Not trained on	Total	Hotel	Taxi	Train	Restaurant
Hotel	89.4/98.4	**65.6/94.9**	99.2/99.2	99.3/99.6	94.2/99.7
Restaurant	89.9/98.3	92.5/99.9	98.4/98.4	97.2/98.5	**72.4/96.3**
Train	91.8/94.9	99.7/99.7	98.7/99.2	**78.0/86.5**	99.3/99.3
Taxi	93.1/98.6	99.4/99.6	**55.6/93.9**	98.9/99.0	99.6/99.7

Note, that the accuracy drops quite significantly on unseen domains this is mainly due to what binary cross entropy function does to the model. However, recall is more important for this task as it just acts as a small filter for entity detection model.

10. End to end performance results

DSTC9 Validation set Task2 (end to end)			
Model	R@1	MRR@5	R@5
Baseline	67.2%	78.4%	92.9%
SRoBERTa Conv1d pooling + distance weighted sampling + MRPC pretrained	11.2%	24.6%	42.3%
All models combined	**95.4%**	?%	?%

11. Implementation details specifics used in test set of the challenge.

11.1. **Task2.1 Out of ontology request domain prediction.** As we expect a new domain in the test set, we try to maximize recall instead of accuracy. This is done by halting the training early using a heuristic, roughly between 1-2 epochs of training. This heuristic, measures the saturation of the model with respect to the validation set. I.e. how many predictions of 1 are made with for a given domain as input, for a ratio of accuracy. I.e. From all the models with >95% accuracy, the one with the highest recall is selected.

11.2. **Task2.2 Entity ID detection.**

11.2.1. *Representing entities.* An entity for the training/test set is represented as (insert appropriate data for given entity id in place of DOMAIN_NAME, CITY_NAME, ENTITY_NAME):

"The domain is DOMAIN_NAME and the city it belongs to is CITY_NAME and the entity name is ENTITY_NAME"

11.2.2. *Sequence tagging pre-training.* DSTC9 dataset is based off Multiwoz2.1 dataset, which contains span labels. Hence, we annotated the DSTC9 dataset with span labels from Multiwoz, indicating the name. Similarly, as we know the out of ontology request's entity's name, we can use that to align it with one of the span labels (using minimum edit distance). Note, this dataset might introduce a lot of errors, however as we only need it as pre-training step to roughly initialize our weights, it does a good enough job. It's trained for 1/3rd of an epoch in the exact manner as a sequence tagging model.

11.2.3. *How negative samples are picked.* The implementation is based off the theory discussed above. The conclusion of theory discussed above is simply that instead of picking from a certain fixed difficulty range like hard-sampling or semi-hard sampling, we should sample more uniformly, have a more varied training data. The distance generator was an inverse Gaussian function with mu=3. The generated value was then used to scan through the entities list, to find which entity had a distance value most similar to the distance generated by the inverse Gaussian function.

11.2.4. *Updating the vector representation for the entities/ knowledge.* Computing the vector representation for each entity/knowledge after each learning step would be too computationally expensive, hence we decide to update them after every epoch.

11.2.5. *Ensemble for task 2.2.* New domain implies new types of entity names, which present various challenges. It being a competition, we want to have the highest score possible while being the rules of the competition. This implies, as the solution doesn't have to be very efficient, or deployed online yet, we can use an ensemble of models.

If the model for task2.2 is trained for 7 epochs, the best performance on the validation set is at the end of 7 epochs. However, if best performance is 98.1%, the model remains at an accuracy of over 97.5% just after 3 epochs. Moreover, if the same model for task2.2 undergoes the same training procedure but doesn't train on a particular domain (domain Independence experiment) it's best performance is somewhere between 3-6 epochs, rather than at the end of 7 epochs. It's quite difficult to overfit huge transformer based models like RoBERTa to the training set by just finetuning it, however, it clearly starts to overfit to the style of text/ style of vocabulary. The test we were to expect not only a new domain, but also some dialogs were speech conversation transcriptions. Hence, we wouldn't know at which point in the training is most suitable for the test set, i.e. is it the model after 3 epochs, or after 3.3 epochs, or after 4.6 epochs etc. Moreover, the model does seem to learn/ pay attention to different aspects after every epoch. This is seen when we investigate the distances between the encoded sentence vectors.

Hence, we decided to choose all the model checkpoints from 3rd-6th epoch, each contributing its confidence score which are then used to rank the entity names according to their relevance.

11.3. **Future work and limitations.** Limitations:

(1) While comparing to the database the assumption is made that a maximum of one knowledge snippet is relevant. (as it uses a similarity measure to rank the database.) This can be solved by converting the distance scores to 0 or 1, by adding another layer to the loss function.
(2) Requires pretraining on span labels, which require some sort of intuition and feature engineering. Potential solutions include, pretraining each sigmoid gate for different generic tasks like various pos tags.
(3) Weighing the importance of each linear layer's weights is non-trivial and training it is difficult. Potential solutions include, comparison with the specific task required.
(4) Computationally inefficient. Requires encoding the the input through RoBerta (which itself is inefficent) 3 times in a single run during training.

Potential Future work:

(1) The above model could be expanded to build a complete dialog state tracker which makes predictions based on confidence scores, provides span labels, and produces vectors for database reference/ ranking.
(2) The Siamese network architecture is nice workaround the binary cross entropy loss problem where the model can continue training without explicit labels. I.e. expanding it to bigger but poorly/un labelled data sets. Or expanding research on confidence scores, etc. Maybe useful for Carel.
(3) A single model can be used to produce contextualized vectors depending on the information to captured instantly. I.e. normally a contextualized vector captures the meaning of the input sequence. However, mapping that to a single vector is non trivial. Eg: For a sequence: "I want to stay at ewing house and then climb the Arthur seat. Do you think its a good idea?" Depending on the context of the task, one might want to represent this sentence only with some relevant parts like either, act of "climbing arthur seat" or the notion of good idea, or all of it. This can be possible by filtering out the tokens with the specialized sigmoid layers.

References

Coope, S., Farghly, T., Gerz, D., Vulić, I., & Henderson, M. (2020). Span-convert: Few-shot span extraction for dialog with pretrained conversational representations.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Heck, M., van Niekerk, C., Lubis, N., Geishauser, C., Lin, H.-C., Moresi, M., & Gašić, M. (2020). Trippy: A triple copy strategy for value independent neural dialog state tracking.

Kim, S., Eric, M., Gopalakrishnan, K., Hedayatnia, B., Liu, Y., & Hakkani-Tur, D. (2020). Beyond domain apis: Task-oriented conversational modeling with unstructured knowledge access.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298682

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Wu, C.-Y., Manmatha, R., Smola, A. J., & Krähenbühl, P. (2018). Sampling matters in deep embedding learning.