A Repository of Conversational Datasets

github.com/PolyAI-LDN/conversational-datasets

Matthew Henderson, Paweł Budzianowski, Íñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulić, and Tsung-Hsien Wen

matt@poly-ai.com
PolyAI Limited, London, UK.

Abstract

Progress in Machine Learning is often driven by the availability of large datasets, and consistent evaluation metrics for comparing modeling approaches. To this end, we present a repository of conversational datasets consisting of hundreds of millions of examples, and a standardised evaluation procedure for conversational response selection models using 1-of-100 accuracy. The repository contains scripts that allow researchers to reproduce the standard datasets, or to adapt the pre-processing and data filtering steps to their needs. We introduce and evaluate several competitive baselines for conversational response selection, whose implementations are shared in the repository, as well as a neural encoder model that is trained on the entire training set.

1 Introduction

Dialogue systems, sometimes referred to as conversational systems or conversational agents, are useful in a wide array of applications. They are used to assist users in accomplishing well-defined tasks such as finding and/or booking flights and restaurants (Hemphill et al., 1990; Williams, 2012; El Asri et al., 2017), or to provide tourist information (Henderson et al., 2014c; Budzianowski et al., 2018). They have found applications in entertainment (Fraser et al., 2018), language learning (Raux et al., 2003; Chen et al., 2017), and healthcare (Laranjo et al., 2018; Fadhil and Schiavo, 2019). Conversational systems can also be used to aid in customer service1 or to provide the foundation for intelligent virtual assistants such as Amazon Alexa, Google Assistant, or Apple Siri.

Modern approaches to constructing dialogue systems are almost exclusively data-driven, supported by modular or end-to-end machine learning frameworks (Young, 2010; Vinyals and Le, 2015; Wen et al., 2015, 2017a,b; Mrkšić and Vulić, 2018; Ramadan et al., 2018; Li et al., 2018, inter alia). The research community, as in any machine learning field, benefits from large datasets and standardised evaluation metrics for tracking and comparing different models. However, collecting data to train data-driven dialogue systems has proven notoriously difficult. First, system designers must construct an ontology to define the constrained set of actions and conversations that the system can support (Henderson et al., 2014a,c; Mrkšić et al., 2015). Furthermore, task-oriented dialogue data must be labeled with highly domain-specific dialogue annotations (El Asri et al., 2017; Budzianowski et al., 2018). Because of this, such annotated dialogue datasets remain scarce, and limited in both their size and in the number of domains they cover. For instance, the recently published MultiWOZ dataset (Budzianowski et al., 2018) contains a total of 115,424 dialogue turns scattered over 7 target domains. Other standard task-based datasets are typically single-domain and smaller by several orders of magnitude: DSTC2 (Henderson et al., 2014b) contains 23,354 turns, Frames (El Asri et al., 2017) comprises 19,986 turns, and M2M (Shah et al., 2018) spans 14,796 turns.

An alternative solution is to leverage larger conversational datasets available online. Such datasets provide natural conversational structure, that is, the inherent context-to-response relationship which is vital for dialogue modeling. In this work, we present a public repository of three large and diverse conversational datasets containing hundreds of millions of conversation examples. Compared to the most popular conversational datasets used in prior work, such as length-restricted Twitter conversations (Ritter et al., 2010) or very technical domain-restricted technical chats from the Ubuntu corpus (Lowe et al., 2015, 2017; Gunasekara et al., 2019), conversations from the three conversational datasets available in the repository are more nat-
ural and diverse. What is more, the datasets are large: for instance, after preprocessing around 3.7B comments from Reddit available in 256M conversational threads, we obtain 727M valid context-response pairs. Similarly, the number of valid pairs in the OpenSubtitles dataset is 316 million. To put these numbers into perspective, the frequently used Ubuntu corpus v2.0 comprises around 4M dialogue turns. Furthermore, our Reddit corpus includes 2 more years of data and so is substantially larger than the previous Reddit dataset of Al-Rfou et al. (2016), which spans around 2.1B comments and 133M conversational threads, and is not publicly available.

Besides the repository of large datasets, another key contribution of this work is the common evaluation framework. We propose applying consistent data filtering and preprocessing to public datasets, and a simple evaluation metric for response selection, which will facilitate direct comparisons between models from different research groups.

These large conversational datasets may support modeling across a large spectrum of natural conversational domains. Similar to the recent work on language model pretraining for diverse NLP applications (Howard and Ruder, 2018; Devlin et al., 2018; Lample and Conneau, 2019), we believe that these datasets can be used in future work to pretrain large general-domain conversational models that are then fine-tuned towards specific tasks using much smaller amounts of task-specific conversational data. We hope that the presented repository, containing a set of strong baseline models and standardised modes of evaluation, will provide means and guidance to the development of next-generation conversational systems.

The repository is available at github.com/PolyAI-LDN/conversational-datasets.

2 Conversational Dataset Format

Datasets are stored as Tensorflow record files containing serialized Tensorflow example protocol buffers (Abadi et al., 2015). The training set is stored as one collection of Tensorflow record files, and the test set as another. Examples are shuffled randomly (and not necessarily reproducibly) within the Tensorflow record files. Each example is deterministically assigned to either the train or test set using a key feature, such as the conversation thread ID in Reddit, guaranteeing that the same split is created whenever the dataset is generated. By default the train set consists of 90% of the total data, and the test set the remaining 10%.

```plaintext
context/1 Hello, how are you?
context/0 I am fine. And you?
context Great. What do you think of the weather?
response It doesn’t feel like February.
```

Figure 1: An illustrative Tensorflow example in a conversational dataset, consisting of a conversational context and an appropriate response. Each string is stored as a bytes feature using its UTF-8 encoding.

Each Tensorflow example contains a conversational context and a response that goes with that context, see e.g. figure 1. Explicitly, each example contains a number of string features:

- A **context** feature, the most recent text in the conversational context.
- A **response** feature, text that is in direct response to the context.
- A number of extra context features, **context/0**, **context/1** etc., going back in reverse order so that **context/i** always refers to the \(i\)th most recent extra context, so that no padding needs to be done, and datasets with different numbers of extra contexts can be mixed.
- Depending on the dataset, there may be some extra features also included in each example. For instance, in Reddit the author of the context and response are identified using additional features.

3 Datasets

Rather than providing the raw processed data, we provide scripts and instructions to the users to generate the data themselves. This allows for viewing and potentially manipulating the pre-processing and filtering steps. The repository contains instructions for generating datasets with standard parameters split deterministically into train and test portions. These allow for defining reproducible evaluations in research papers. Section 5 presents benchmark results on these standard datasets for a variety of conversational response selection models.

Dataset creation scripts are written using Apache Beam and Google Cloud Dataflow (Akidau et al., 2015), which parallelizes the work across many machines. Using the default quotas, the Reddit script
Table 1: Summary of the datasets included in the public repository. The Reddit data is taken from January 2015 to December 2018, and the OpenSubtitles data from 2018.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Built from</th>
<th>Training size</th>
<th>Testing size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reddit</td>
<td>3.7 billion comments in threaded conversations</td>
<td>654,396,778</td>
<td>72,616,937</td>
</tr>
<tr>
<td>OpenSubtitles</td>
<td>over 400 million lines from movie and television subtitles (also available in other languages)</td>
<td>283,651,561</td>
<td>33,240,156</td>
</tr>
<tr>
<td>AmazonQA</td>
<td>over 3.6 million question-response pairs in the context of Amazon products</td>
<td>3,316,905</td>
<td>373,007</td>
</tr>
</tbody>
</table>

3.1 Reddit

Reddit is an American social news aggregation website, where users can post links, and take part in discussions on these posts. Reddit is extremely diverse (Schrading et al., 2015; Al-Rfou et al., 2016): there are more than 300,000 sub-forums (i.e., subreddits) covering various topics of discussion. These threaded discussions, available in a public BigQuery database, provide a large corpus of conversational contexts paired with appropriate responses. Reddit data has been used to create conversational response selection data by Al-Rfou et al. (2016); Cer et al. (2018); Yang et al. (2018). We share code that allows generating datasets from the Reddit data in a reproducible manner: with consistent filtering, processing, and train/test splitting. We also generate data using two more years of data than the previous work, 3.7 billion comments rather than 2.1 billion, giving a final dataset with 176 million more examples.

Reddit conversations are threaded. Each post may have multiple top-level comments, and every comment may have multiple children comments written in response. In processing, each Reddit thread is used to generate a set of examples. Each response comment generates an example, where the context is the linear path of comments that the comment is in response to.

Examples may be filtered according to the contents of the context and response features. The example is filtered if either feature has more than 128 characters, or fewer than 9 characters, or if its text is set to [deleted] or [removed]. Full details of the filtering are available in the code, and configurable through command-line flags.

Further back contexts, from the comment’s parent’s parent etc., are stored as extra context features. Their texts are trimmed to be at most 128 characters in length, without splitting words apart. This helps to bound the size of an individual example.

The train/test split is deterministic based on the thread ID. As long as all the input to the script is held constant (the input tables, filtering thresholds etc.), the resulting datasets should be identical.

The data from 2015 to 2018 inclusive consists of 3,680,746,776 comments, in 256,095,216 threads. In total, 727,013,715 Tensorflow examples are created from this data.

3.2 OpenSubtitles

OpenSubtitles is a growing online collection of subtitles for movies and television shows available in multiple languages. As a starting point, we use the corpus collected by Lison and Tiedemann (2016), originally intended for statistical machine translation. This corpus is regenerated every year, in 62 different languages.

Consecutive lines in the subtitle data are used to create conversational examples. There is no guarantee that different lines correspond to different speakers, or that consecutive lines belong to the same scene, or even the same show. The data nevertheless contains a lot of interesting examples for modelling the mapping from conversational contexts to responses.

Short and long lines are filtered, and some text is filtered such as character names and auditory description text. The English 2018 data consists
of 441,450,449 lines, and generates 316,891,717 examples. The data is split into chunks of 100,000 lines, and each chunk is used either for the train set or the test set.

3.3 AmazonQA

This dataset is based on a corpus extracted by Wan and McAuley (2016); McAuley and Yang (2016), who scraped questions and answers from Amazon product pages. This provides a corpus of question-answer pairs in the e-commerce domain. Some questions may have multiple answers, so one example is generated for each possible answer.

Examples with very short or long questions or responses are filtered from the data, resulting in a total of 3,689,912 examples. The train/test split is computed deterministically using the product ID.

4 Response Selection Task

The conversational datasets included in this repository facilitate the training and evaluation of a variety of models for natural language tasks. For instance, the datasets are suitable for training generative models of conversational response (Serban et al., 2016; Ritter et al., 2011; Vinyals and Le, 2015; Sordoni et al., 2015; Shang et al., 2015; Kannan et al., 2016), as well as discriminative methods of conversational response selection (Lowe et al., 2015; Inaba and Takahashi, 2016; Yu et al., 2016; Henderson et al., 2017).

The task of conversational response selection is to identify a correct response to a given conversational context from a pool of candidates, as illustrated in figure 2. Such models are typically evaluated using Recall@k, a typical metric in information retrieval literature. This measures how often the correct response is identified as one of the top k ranked responses (Lowe et al., 2015; Inaba and Takahashi, 2016; Yu et al., 2016; Al-Rfou et al., 2016; Henderson et al., 2017; Lowe et al., 2017; Wu et al., 2017; Cer et al., 2018; Chaudhuri et al., 2018; Du and Black, 2018; Kumar et al., 2018; Liu et al., 2018; Yang et al., 2018; Zhou et al., 2018; Gunasekara et al., 2019; Tao et al., 2019). Models trained to select responses can be used to drive dialogue systems, question-answering systems, and response suggestion systems. The task of response selection provides a powerful signal for learning implicit semantic representations useful for many downstream tasks in natural language understanding (Cer et al., 2018; Yang et al., 2018).

The Recall@k metric allows for direct comparison between models. Direct comparisons are much more difficult for generative models, which are typically evaluated using perplexity scores or using human judgement. Perplexity scores are dependent on normalization, tokenization, and choice of vocabulary, while human judgement is expensive and time consuming.

When evaluating conversational response selection models on these datasets, we propose a Recall@k metric termed 1-of-100 accuracy. This is Recall@1 using 99 responses sampled from the test dataset as negatives. This 1-of-100 accuracy metric has been used in previous studies: (Al-Rfou et al., 2016; Henderson et al., 2017; Cer et al., 2018; Kumar et al., 2018; Yang et al., 2018; Gunasekara et al., 2019). While there is no guarantee that the 99 randomly selected negatives will all be bad responses, the metric nevertheless provides a simple summary of model performance that has been shown to correlate with user-driven quality metrics (Henderson et al., 2017). For efficient computation of this metric, batches of 100 (context, response) pairs can be processed such that the other 99 elements in the batch serve as the negative examples.

Sections 4.1 and 4.2 present baseline methods of conversational response selection that are implemented in the repository. These baselines are intended to run quickly using a subset of the training data, to give some idea of performance and characteristics of each dataset. Section 4.3 describes a more competitive neural encoder model that is trained on the entire training set.
4.1 Keyword-based Methods

The keyword-based baselines use keyword similarity metrics to rank responses given a context. These are typical baselines for information retrieval tasks. The TF-IDF method computes inverse document frequency statistics on the training set, and scores responses using their tf-idf cosine similarity to the context (Manning et al., 2008).

The BM25 method builds on top of the tf-idf similarity, applying an adjustment to the term weights (Robertson and Zaragoza, 2009).

4.2 Vector-based Methods

The vector-based methods use publicly available neural net embedding models to embed contexts and responses into a vector space. We include the following five embedding models in the evaluation, all of which are available on Tensorflow Hub:

- **USE** the Universal Sentence Encoder from Cer et al. (2018)
- **USE-LARGE** a larger version of the Universal Sentence Encoder
- **ELMO** the Embeddings from Language Models approach from Peters et al. (2018).
- **BERT-SMALL** the deep bidirectional transformer model of Devlin et al. (2018).
- **BERT-LARGE** a larger deep bidirectional transformer model.

There are two vector-based baseline methods, one for each of the above models. The SIM method ranks responses according to their cosine similarity with the context vector. This method relies on pretrained models and does not use the training set at all.

The MAP method learns a linear mapping on top of the response vector. The final score of a response with vector \( y \) given a context with vector \( x \) is the cosine similarity \( \langle \cdot, \cdot \rangle \) of the context vector with the mapped response vector:

\[
\langle x, (W + \alpha I) \cdot y \rangle
\]

where \( W, \alpha \) are learned parameters and \( I \) is the identity matrix. This allows learning an arbitrary linear mapping on the context side, while the residual connection gated by \( \alpha \) makes it easy for the model to interpolate with the SIM baseline. Vectors are L2-normalized before being fed to the MAP method, so that the method is invariant to scaling.

The combination of the three embedding models with the two vector-based methods results in the following six baseline methods: USE-SIM, USE-MAP, USE-LARGE-SIM, USE-LARGE-MAP, ELMO-SIM, and ELMO-MAP.

4.3 Encoder Model

We also train and evaluate a neural encoder model that maps the context and response through separate sub-networks to a shared vector space, where the final score is a dot-product between a vector representing the context and a vector representing the response as per Henderson et al. (2017); Cer et al. (2018); Kumar et al. (2018); Yang et al. (2018). This model is referred to as POLYAI-ENCODER in the evaluation.

Full details of the neural structure are given in Henderson et al. (2019). To summarize, the context and response are both separately passed through sub-networks that:

1. split the text into unigram and bigram features
2. convert unigrams and bigrams to numeric IDs using a vocabulary of known features in conjunction with a hashing strategy for unseen features
3. separately embed the unigrams and bigrams using large embedding matrices
4. separately apply self-attention then reduction over the sequence dimension to the unigram and bigram embeddings
5. combine the unigram and bigram representations, then pass them through several dense hidden layers
6. L2-normalize the final hidden layer to obtain the final vector representation

Both sub-networks are trained jointly using the dot-product loss of Henderson et al. (2017), with label smoothing and a learned scaling factor.

5 Evaluation

All the methods discussed in section 4 are evaluated on the three standard datasets from section 3,
and the results are presented in table 2. In this evaluation, all methods use only the (immediate) context feature to score the responses, and do not use other features such as the extra contexts.

The keyword-based TF-IDF and BM25 are broadly competitive with the vector-based methods, and are particularly strong for AmazonQA, possibly because rare words such as the product name are informative in this domain. Learning a mapping with the MAP method gives a consistent boost in performance over the SIM method, showing the importance of learning the mapping from context to response versus simply relying on similarity. This approach would benefit from more data and a more powerful mapping network, but we have constrained the baselines so that they run quickly on a single computer. The Universal Sentence Encoder model outperforms ELMo in all cases.

The POLYAI-ENCODER model significantly outperforms all of the baseline methods. This is not surprising, as it is trained on the entire training set using multiple GPUs for several hours. We welcome other research groups to share their results, and we will be growing the table of results in the repository.

### Table 2: 1-of-100 accuracy results for keyword-based baselines, vector-based baselines, and the encoder model for each of the three standard datasets. The latest evaluation results are maintained in the repository. Results are computed on a random subset of 50,000 examples from the test set (500 batches of 100).

<table>
<thead>
<tr>
<th>Model</th>
<th>Reddit</th>
<th>OpenSubtitles</th>
<th>AmazonQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF</td>
<td>26.7</td>
<td>10.9</td>
<td>51.8</td>
</tr>
<tr>
<td>BM25</td>
<td>27.6</td>
<td>10.9</td>
<td>52.3</td>
</tr>
<tr>
<td>USE-SIM</td>
<td>36.6</td>
<td>13.6</td>
<td>47.6</td>
</tr>
<tr>
<td>USE-MAP</td>
<td>40.8</td>
<td>15.8</td>
<td>54.4</td>
</tr>
<tr>
<td>USE-LARGE-SIM</td>
<td>41.4</td>
<td>14.9</td>
<td>51.3</td>
</tr>
<tr>
<td>USE-LARGE-MAP</td>
<td>47.7</td>
<td>18.0</td>
<td>61.9</td>
</tr>
<tr>
<td>ELMO-SIM</td>
<td>12.5</td>
<td>9.5</td>
<td>16.0</td>
</tr>
<tr>
<td>ELMO-MAP</td>
<td>19.3</td>
<td>12.3</td>
<td>33.0</td>
</tr>
<tr>
<td>BERT-SMALL-SIM</td>
<td>17.1</td>
<td>13.8</td>
<td>27.8</td>
</tr>
<tr>
<td>BERT-SMALL-MAP</td>
<td>24.5</td>
<td>17.5</td>
<td>45.8</td>
</tr>
<tr>
<td>BERT-LARGE-SIM</td>
<td>14.8</td>
<td>12.2</td>
<td>25.9</td>
</tr>
<tr>
<td>BERT-LARGE-MAP</td>
<td>24.0</td>
<td>16.8</td>
<td>44.1</td>
</tr>
<tr>
<td>POLYAI-ENCODER</td>
<td><strong>61.3</strong></td>
<td><strong>30.6</strong></td>
<td><strong>84.2</strong></td>
</tr>
</tbody>
</table>

6 Conclusion

This paper has introduced a repository of conversational datasets, providing hundreds of millions examples for training and evaluating conversational response selection systems under a standard evaluation framework. Future work will involve introducing more datasets in this format, more competitive baselines, and more benchmark results. We welcome contributions from other research groups in all of these directions.

### References


Tyler Akidau, Robert Bradshaw, Craig Chambers,


Steve Young. 2010. Still talking to machines (cognitively speaking). In Proceedings of INTERSPEECH.


A Appendix

Reddit

context/2 Could someone there post a summary of the insightful moments.
context/1 Basically L2L is the new deep learning.
context/0 What’s “L2L” mean?
context “Learning to learn”, using deep learning to design the architecture of another deep network: https://arxiv.org/abs/1606.04474
response using deep learning with SGD to design the learning algorithms of another deep network *
context_author goodside
response_author NetOrBrain
subreddit MachineLearning
thread_id 5h6yvl

OpenSubtitles

context/9 So what are we waiting for?
context/8 Nothing, it...
context/7 It’s just if...
context/6 If we’ve underestimated the size of the artifact’s data stream...
context/5 We’ll fry the ship’s CPU and we’ll all spend the rest of our lives stranded in the Temporal Zone.
context/4 The ship’s CPU has a name.
context/3 Sorry, Gideon.
context/2 Can we at least talk about this before you connect...
context/1 Gideon?
context/0 You still there?
response Artificial intelligences cannot, by definition, be killed, Dr. Palmer.

file_id lines-emk

AmazonQA

context i live in singapore so i would like to know what is the plug cos we use those 3 pin type
response it’s a 2 pin U.S. plug, but you can probably get an adapter, very good hair dryer!

product_id B003XNYHWS

Figure 3: Examples from the three datasets. Each example is a mapping from feature names to string features. Features with a star * are used to compute the deterministic train/test split.