Constructing Datasets from Dialogue Data

Ondřej Sotolář, Jaromír Plhák, Michaela Lebedíková, Michał Tkaczyk, and David Šmahel

Faculty of Informatics, Masaryk University,
Botanická 68a, 602 00, Brno, Czech Republic
{xsotolar, xplhak, x450458, x245062, davs}@fi.muni.cz

Abstract. We present methods for transforming raw dialogue data into a dataset suitable for processing with statistical NLP models. We reveal the potential pitfalls for processing this type of data, such as ensuring the representatives of the sample, the generalization ability of models, and the definition of the local context of the utterances. We use novel methods to solve these problems and demonstrate their effectiveness on an utterance classification problem. As a result, this paper provides guidelines for generating valuable datasets from dialogue data.

Keywords: Dialogue Dataset, Dataset Split, Online Conversations

1 Introduction

Online communication allows for the synchronous exchange of text, images, voice, and videos between two or more people [6]. It is realized using native applications such as Messenger, WhatsApp, and Discord, through social networks such as Facebook or Twitter, dedicated internet forums, and online discussions in general. Such communication is often studied in dialogue systems, which is concerned with designing agents (commonly called chatbots) that are capable of participating in the discourse [7]. In the case of dialogue systems, we have information available about the dialogue, such as the intents, topics, and dialog flow, because the agent actively shapes the discourse. In this work, we consider the general case, where we do not have this type of special information, and we work only with the text content of the dialogues and their metadata.

Our motivation for examining a sample of online dialogues is the discovery of phenomena of interest. Doing so with conventional and manual methods is inefficient for two reasons: a sample of a significant size includes a large quantity of unstructured text, and the occurrence of the researched phenomena might be low [14]. While this goal is similar to intent detection, as known in dialogue systems, using the existing methods is usually impossible. In the general case, we lack information on the structure of the dialogues, which the conversational agent controls. Information on how to approach Natural Language Processing (NLP) tasks in the general case is scattered among small, often unrelated tasks (such as Dialogue Act Recognition [11,10], Argument Mining [12,5,2,9], Short

A. Horák, P. Rychlý, A. Rambousek (eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2022, pp. 131–139, 2022.

Text Classification [8,4], or Emotion Detection [1]). This makes it difficult to research methodologies for novel tasks.

This paper aims to present a practical methodology for processing raw dialogue data. We provide guidelines with examples, diagrams, and algorithms for the complete process of generating datasets from dialogue data. In Section 2, we present a practical methodology for storing and retrieving the dialogues. Section 3 presents an algorithm for constructing training examples with a local context. Finally, in Section 4, we propose a novel algorithm for k-fold data splitting.

2 Data Structure for Storing and Retrieving Dialogues

Dialogues are composed of temporal sequences of *utterances* as shown in Table 1. Utterances are typically short text fragments, complete sentences, or short sequences thereof. They are uniquely identified by their timestamp within the dialogue. Metadata associated with individual utterances can be, for example, the author, a reference to previous utterances (identifying a response), and others. Many applications allow to use multi-media in the dialogues; however, we omit them here.

Furthermore, in this work, we consider the data to be annotated with class labels at the utterance level. The labels identify either syntactic or semantic phenomena per their definition. Whether the dialogue context influences the utterance labels is a design decision. In the following, we encode the *none* class with index 0 and others with $\{1..N\}$.

Table 1: Excerpt from a dialogue from the dataset from [13] (translated to English) showing time, participant name, the utterance, and the phenomenon label in each row.

Time	Author	Utterance	Class
01:44:07	John	I'll finish the Math task tomorrow	none
01:44:13	John	Like, I really have to do it	none
01:44:28	Tim	The math task looks easy to me	Emotional Support
01:44:49	Tim	You have 6 hours to deadline, chill	Emotional Support
01:46:34	John	But I'm really tired after the day	none
01:47:51	Tim	I'm having some tea and I'm super	none

To efficiently store and retrieve dialogues, we propose a hierarchical data structure that reflects the relationships between the dialog components. At the top level, we can, in many cases, identify an owner, first author, or originator of the dialogue. In Instant Messaging (IM), it is a user; in forums and social media, this would be the original poster (the topic creator). The second level of the structure is composed of *threads* that group dialogues together. The set of utterance authors in a thread is unique. Because threads might be long-running, we suggest a third level of the structure. There, we delimit individual

conversations with a time constraint to help separate different topics. We argue that in long-running threads, after a certain pause, the topic is more likely to change. We suggest finding the threshold for separating the conversations experimentally. For example, with IM conversations, we have used 1-hour long pauses to delimit them. The resulting structure is shown in Figure 1.

For practical reasons, we map this structure to a table in a relational database shown in Figure 2. In Figure 3, we show an example PostgreSQL query for retrieving dialogue data. It is structured into conversations, where each row of the result contains one conversation with its utterances and labels in ordered lists. Any other metadata can be retrieved similarly.

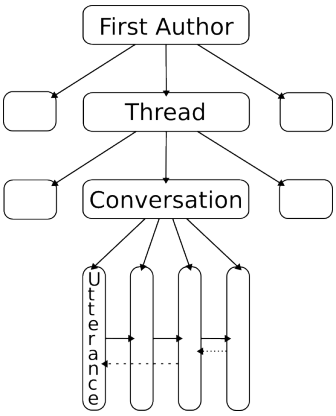


Fig. 1: Data structure for dialogues segmented by the first author, threads, time-delimited conversations, and utterances.

column_name	type
uid_first_author	integer
thread_id	integer
conversation_id	integer
line_num	integer
time	timestamp
uid_author	integer
reaction_to	integer
label	integer
text	varchar

Fig. 2: Relational database table for structure shown in Figure 1.

```
SELECT
  ARRAY_AGG(d.line_text::TEXT ORDER BY line_num asc) AS texts,
  ARRAY_AGG(d.label::INTEGER ORDER BY line_num asc) AS labels
FROM dialogues s
WHERE d.label = 0 OR d.label = 1
GROUP BY d.uid_first_author, d.thread_id, d.conversation_id;
```

Fig. 3: Example of a SQL query (PostgreSQL) to retrieve the time-delimited conversations.

3 Constructing Training Examples with Local Context

To construct training examples, we could use individual utterances. However, previous research [3,10,4,9] has shown that including the context of the dialogue can improve the solutions for many different tasks. In this work, where we consider utterance-level labels, we also use the concept of *local context*. The local context of a target utterance is defined as a window of the neighboring utterances. The size and position of the context window is a design decision. It is called local context because the window size usually covers only a few utterances, and its purpose is to help the model to capture local dependencies. This contrasts with other types of context, such as the whole dialogue or the language style of a particular user across many dialogues. Such types of context often span a large amount of text, which has to be condensed due to the practical limits of sequential models used in NLP. Conversely, local context can usually be used in its original text form.

We present an algorithm for delimiting the local context of a target utterance in Algorithm 1. We use the sliding window concept. We found that individual utterances differ significantly in their character length; thus, we do not define the threshold with an utterance count but with a character count instead. In our algorithm, the context window includes an arbitrary number of neighboring utterances as long as the sum of their character lengths does not exceed the given limit. The limit is soft: if the character limit is reached within the bounds of an utterance, it is appended as a whole, thus exceeding the limit. The algorithm constitutes a complete solution to generate the training dataset if iterated over selected conversations (or whole threads).

Algorithm 1: Algorithm for constructing training examples.

function rows_to_examples

Input : *rows*: list of utterance tuples (text, label $\in \{0..N\}$, metadata) ordered by time (conversation or whole thread),
ctx: character length of context,
sep: utterance separator,
sep_{target}: separator of target utterance,
has_small_pieces: stop if window reaches start & contains all 0s.

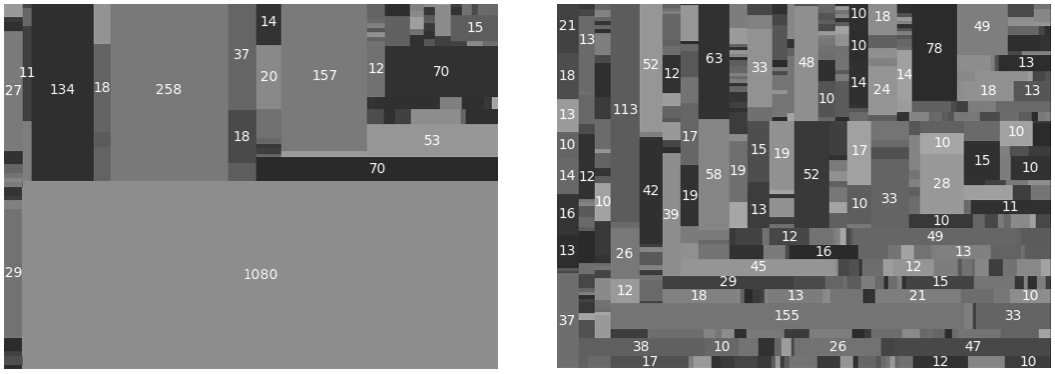
Output : *examples*: set of examples as tuples (text, label, metadata)

$examples \leftarrow \emptyset$;
for t, l *in rows* **do**
  if $len(t) < 2$ **then** `# Case when only one utterance in dialogue`
    `# Compose function concatenates the utterance data in the window (uses separators for text).`
    $examples.add(compose\_example(t, l, sep, sep_{target}))$;
    continue
  $target_i \leftarrow len(t) - 1$;
  while $target_i > 0$ **do** `# Case when 2 and more utterances`
    $last_i \leftarrow target_i$;
    $char_{cnt} \leftarrow 0$;
    while $last_i > 0$ & $char_{cnt} < ctx$ **do**
      $last_i \mathrel{-}= 1$;
      $char_{cnt} \mathrel{+}= len(t[last_i])$;
      $examples.add($
        $compose\_example(t[last_i : target_i + 1], l[last_i : target_i + 1], sep, sep_{target}))$;
      if has_small_pieces & $last_i = 0$ & $sum(l[0 : target_i]) = 0$ **then**
        break;
      $target_i \mathrel{-}= 1$;
return *examples*;

4 Selecting Representative Samples

The performance of predictive models is measured with a testing sample that is unseen during training. The measurement reliability depends on the validity of this sample, which should be representative of the actual data, to determine whether the model overfits the training sample. With dialogues, this concerns the style of individual authors and also the topics of the conversations, which might have a distinct vocabulary. We argue that splitting such data naively into the training and testing samples may positively bias the performance measurement, thus not reflecting the model's true generalization ability. Imagine a user who authors many utterances on a single topic with distinct keywords. Consider a classification task: if we split this data between the train and test samples, we risk overfitting the model on the keywords, then also successfully classifying the examples in the test set. This would result in a model with a good measured performance but a poor generalization ability because it would likely fail on data from other users.

To avoid this issue, we first suggest analyzing the data with regard to the contributions of individual utterance authors. In Figure 4, we show different samples of a dataset from [13], each annotated with a different class. We would assume that sample a) *is not* representative and sample b) *is* representative of real-world data based on the distribution of the contributions of different authors of the utterances.



a) b)

Fig. 4: The content contributions in samples of the IM dataset from [13], each labeled with a particular class. One rectangle represents a person. The relative size of the rectangle and the number within each rectangle represent the number of utterances they authored.

Second, we suggest using k-fold cross-validation. To use it, we need to split the data in a stratified manner for a given k. However, standard splitting algorithms are unsuitable for dialogues because we need specific criteria to define the splits. Ideally, we would have examples with disjoint sets of authors in each split. This is not always necessary; thus, we need a parameter for setting the maximal overlap of the example's authorship between the splits (max_{c_rank}).

Furthermore, following the idea of stratified sampling, the splitting algorithm should keep the same size of the splits and also the same class ratio.

In Algorithms 2, 3, and 4, we present the *dialog_k_fold* algorithm, which splits the data into k different groups per the specified criteria. If such a split exists, the algorithm will return the number of splits given by the k parameter. Otherwise, it will return the maximum possible splits for the criteria plus the set of remaining examples. We formalize the condition for proving its effectiveness in Lemma 1.

Algorithm 2: Algorithm for modified k-fold split for dialogue data.

function dialog_k_fold

Input : E: set of examples,
k: desired number of splits,
max_{c_rank}: threshold for maximum number of author overlap

Output : *splits*: list of 1..k data splits
remainder: examples excluded from the split groups

$T \leftarrow group\_by\_author\_tuples(E)$;
$G \leftarrow (g_1, ..., g_k)$;
$R \leftarrow \emptyset$;
while $\exists t \in T : \neg \exists g \in G : t \in g$ **do**
  $t \leftarrow t$ with $min(t.c\_rank)$, **if** tied **then** use $max(t.size)$;
  $g \leftarrow best\_group(G, t, max_{c\_rank}, \frac{size(E)}{k}, label\_ratio(E))$;
  $g.add(t)$;
$R \leftarrow T \quad \{t : t \in g, g \in G\}$;
return G, R;

Lemma 1. *Given a dataset $\mathcal{D}$, a model performance measurement $\mathcal{M}(train, test)$, let $(d_1, d_2, d_3) \leftarrow dialog\_k\_fold(\mathcal{D}, k = 3)$ create three splits, where the overlap between utterance authors is minimal. Set $\mathcal{D}_{holdout} \leftarrow d_3$ aside as a holdout set and merge the rest $\mathcal{D}_{new} \leftarrow d_1 \cup d_2$. Let:*

$$\mathcal{S}_{rand} = (r_1, r_2) \leftarrow random\_split(\mathcal{D}_{new}, k = 2), \tag{1}$$

$$\mathcal{S}_{dkf} = (f_1, f_2) \leftarrow dialog\_k\_fold(\mathcal{D}_{new}, k = 2). \tag{2}$$

Then, the (cross-validated) difference between $\mathcal{M}(r_x, r_y)$ and $\mathcal{M}(r_x, \mathcal{D}_{holdout})$ should be greater than if we use $\mathcal{S}_{dkf}$ for training. The following condition should hold up to additional cross-validation, i.e. for all permutations of (d_1, d_2, d_3):

$$avg|\mathcal{M}(r_x, r_y) - \mathcal{M}(r_x, \mathcal{D}_{holdout})| > avg|\mathcal{M}(f_x, f_y) - \mathcal{M}(f_x, \mathcal{D}_{holdout})|, \tag{3}$$

where:

$$x, y \in \{1, 2\}, x \neq y.$$

Proof. We prove Lemma 1 experimentally using the dataset sample b) presented in Figure 4, which shows the utterance author distribution. The results presented in Figure 5 demonstrate that condition (3) of Lemma 1 holds.

Algorithm 3: Algorithm for grouping examples by author tuples. Additionally, it computes author overlap and ranks each group by its severity.

function group_by_author_tuples

Input : $Examples(thread\_id, labels, lines\_per\_author)$: set of examples
Output : T: set of grouped examples with aggregate attributes
$T \leftarrow$ group *examples* by author tuples with aggregates:
    $size \leftarrow count(labels)$,
    $ratio \leftarrow$ ratio of labels in the group,
    $authors \leftarrow$ indexed list with sum of each author's utterance count.
for $t \in T$ **do** `# Calculate utterance authors overlap into an n-hot vector`
    for $a \in t.authors$ **do**
        for $g_i \in T, g_i \neq t$ **do**
            $t.conflicts[i]$ += **if** $a \in g_i.authors$ **then** $t.count(a)$ **else** 0;

`# Finally, we rank the groups in T. The rank values can have duplicates.`
foreach $t \in T$: $t.c\_rank \leftarrow$ order of t in T ordered by $sum(t.conflicts)$;
return T;

Algorithm 4: Algorithm for selecting the split group to add the given author group.

function best_group

Input : G: split groups,
    t: group of examples (grouped by author tuple),
    max_{c_rank}: threshold for maximal author overlap,
    $size_{desired}$: desired group size: ideally $\frac{size(dataset)}{k}$,
    $class\_ratio_{desired}$: desired class ratio: ideally same as original dataset
Output : g: chosen split group
$g_m \leftarrow$ group with minimum conflicts with t;
if $g_m.conflicts(t) > max_{c\_rank}$ **then**
    return $\emptyset$;
$g_n \leftarrow$ group with maximum $|g_n.size - size_{desired}|$ with t;
$g_o \leftarrow$ group with maximum $|g_n.class\_ratio - class\_ratio_{desired}|$ with t;
$g \leftarrow$ select from $\{g_m, g_n, g_o\}$ with most votes, **if** tied **then** take g_m;
return g

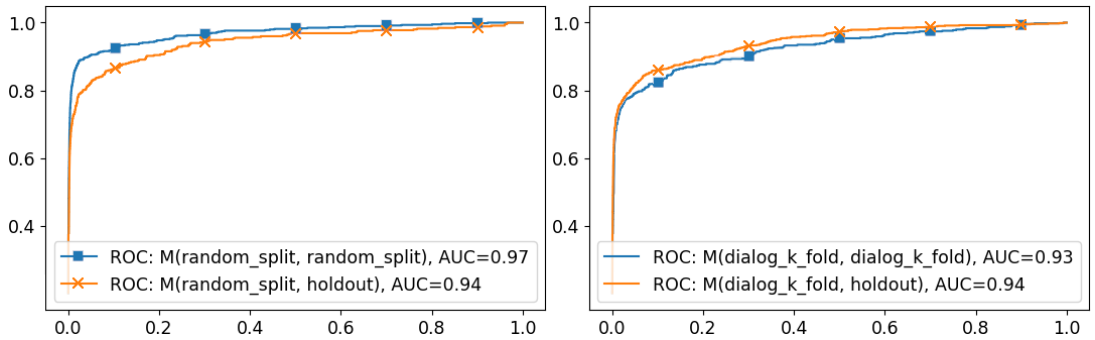


Fig. 5: Model performance comparison using a performance metric $\mathcal{M}$ (see Lemma 1) and a holdout set $\mathcal{D}_{holdout}$, where utterances are authored by a disjoint set of authors than in $\mathcal{D}_{new}$. In a), $\mathcal{D}_{new}$ is randomly split into (r_x, r_y). The model trained and tested on (r_x, r_y) shows artificially higher measured performance than the more reliable measured performance of the same model tested on $\mathcal{D}_{holdout}$. In b), where $\mathcal{D}_{new}$ is split using *dialog_k_fold*, we can see the significantly closer performance, proving the effectiveness of our algorithm.

5 Discussion and Limitations

We have experimentally proven that *dialog_k_fold* effectively improves the measured performance reliability. However, we have not given formal proof of the algorithm correctness or time complexity. We leave this for future work. Furthermore, the function *best_group* defined in Algorithm 4 leads to a multi-criteria optimization problem, which in our current implementation, we have solved with a rule-based, heuristic approach. We suggest finding an optimal solution in further work. The function also needs to calculate the authorship overlap of each utterance with each other in $\mathcal{O}(n^2)$ time which is expensive for large datasets.

6 Conclusion

We have presented practical methods for structuring, storing, and retrieving dialogue data. We have also presented an algorithm for constructing training examples from such data. Furthermore, we presented a novel k-fold algorithm for the stratified splitting of datasets of dialogue data. We have demonstrated that using our *dialog_k_fold* algorithm improves the reliability of performance measurements when compared to naive splitting methods.

Acknowledgements. This work has received funding from the Czech Science Foundation, project no. 19-27828X.

References

1. Acheampong, F.A., Wenyu, C., Nunoo-Mensah, H.: Text-based emotion detection: Advances, challenges, and opportunities. Engineering Reports (7), e12189 (2020)
2. Aker, A., Sliwa, A., Ma, Y., Lui, R., Borad, N., Ziyaei, S., Ghobadi, M.: What works and what does not: Classifier and feature analysis for argument mining. In: Proceedings of the 4th Workshop on Argument Mining. pp. 91–96 (2017)
3. Chatterjee, A., Narahari, K.N., Joshi, M., Agrawal, P.: Semeval-2019 task 3: Emocontext contextual emotion detection in text. In: Proceedings of the 13th international workshop on semantic evaluation. pp. 39–48 (2019)
4. Chen, J., Hu, Y., Liu, J., Xiao, Y., Jiang, H.: Deep short text classification with knowledge powered attention. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6252–6259 (2019)
5. Habernal, I., Gurevych, I.: Argumentation mining in user-generated web discourse. Computational Linguistics (1), 125–179 (2017)
6. Huang, H., Leung, L.: Instant messaging addiction among teenagers in china: Shyness, alienation, and academic performance decrement. CyberPsychology & Behavior (6), 675–679 (2009)
7. Jurafsky, D., Martin, J.H.: Chapter 24: Chatbots and dialogue systems in speech and language processing. vol. 3. US: Prentice Hall (2014)
8. Lee, J.Y., Dernoncourt, F.: Sequential short-text classification with recurrent and convolutional neural networks. arXiv preprint arXiv:1603.03827 (2016)
9. Lugini, L., Litman, D.: Contextual argument component classification for class discussions. arXiv e-prints pp. arXiv–2102 (2021)
10. Martínek, J., Cerisara, C., Král, P., Lenc, L.: Cross-lingual approaches for task-specific dialogue act recognition. In: IFIP International Conference on Artificial Intelligence Applications and Innovations. pp. 232–242. Springer (2021)
11. Martínek, J., Král, P., Lenc, L., Cerisara, C.: Multi-lingual dialogue act recognition with deep learning methods. arXiv preprint arXiv:1904.05606 (2019)
12. Persing, I., Ng, V.: End-to-end argumentation mining in student essays. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1384–1394 (2016)
13. Sotolář, O., Plhák, J., Šmahel, D.: Towards personal data anonymization for social messaging. In: International Conference on Text, Speech, and Dialogue. pp. 281–292. Springer (2021)
14. Sotolář, O., Plhák, J., Tkaczyk, M., Lebedíková, M., Šmahel, D.: Detecting online risks and supportive interaction in instant messenger conversations using czech transformers. RASLAN 2021 Recent Advances in Slavonic Natural Language Processing p. 19 (2021)