Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference "Dialogue 2023"

Moscow, June 14–17, 2023

Who is answering to whom? Modeling reply-to relationships in Russian asynchronous chats

Igor Buyanov
FRC CSC RAS
Moscow, Russia
buyanov.igor.o@yandex.ru

Darya Yaskova
MTS AI
Moscow, Russia
dary.yaskova@gmail.com

Ilia Sochenkov
FRC CSC RAS
Moscow, Russia
sochenkov@isa.ru

Abstract

The study highlights the asynchronous nature of modern group chats and related problems such as retrieving relevant information on the asked question and understanding reply-to relationships. In this work, we formalize the reply recovery task as a building block toward solving described problems. Using simple heuristics, we try to apply the result reply recovery model to a thread reconstruction problem. As a result, we show that modern pre-trained models such as BERT show great results on the task of reply recovery compared to more simple models, though it cannot be applied to thread reconstruction with just simple heuristics. In addition, experiments have shown that model performance depends on the chat domain. We open-sourced a model that can automatically predict which message the particular reply responds to and provide a representative Russian dataset that we built from Telegram chats of different domains. We also provide a test set for a thread reconstruction task. [1]

Keywords: Russian language, asynchronous chats, group discussion, thread reconstruction, reply recovery, BERT

DOI: 10.28995/2075-7182-2023-22-1052-1060

Кто кому отвечает? Моделирование взаимоотношений сообщений в асинхронных чатах на русском языке

Игорь Буянов
ФИЦ ИУ РАН
Москва, Россия
buyanov.igor.o@yandex.ru

Дарья Яськова
MTS AI
Москва, Россия
dary.yaskova@gmail.com

Илья Соченков
ФИЦ ИУ РАН
Москва, Россия
sochenkov@isa.ru

Аннотация

В исследовании поднимается тема асинхронной природы современных групповых чатов и связанных этим проблем, таких как получение соответствующей информации и понимание того, кто кому отвечает. В этой работе мы формализуем задачу восстановления ответов как базовый блок в решении описанных проблем. Используя простые эвристики, мы стараемся применить полученную модель восстановления ответа к проблеме реконструкции тредов сообщений. В результате мы показали, что современные предварительно обученные модели, такие как BERT, показывают отличные результаты на задаче восстановления ответов по сравнению с более простыми моделями. Тем не менее тесты показали, что использование модели с простыми эвристиками не дают хороших результатов на задаче реконструкции тредов. Кроме того, эксперименты показали, что производительность модели зависит от домена чата. Мы опубликовали модель и набор данных на русском языке, который мы создали из чатов Telegram из разными доменов, для задачи восстановления ответов. Мы также опубликовали тестовый набор для задачи реконструкции тредов.

Ключевые слова: русский язык, асинхронный чат, обсуждения в группе, реконструкция тредов, восстановление ответов, BERT

[1] https://github.com/Astromis/research/tree/master/reply_recovery

1 Introduction

The spread and availability of the Internet allow people to be in touch with each other, regardless of their location. Via Web, one can ask for help about a particular problem or discuss any topic with other people. To make this process more ordered, people start to form communities and online forums that dedicate to some declared theme.

With further development of web technologies and, in particular, smartphones, several quick message applications gained popularity such as Telegram and WhatsApp. The fact that smartphones are always near hand coupled with a live time regime of receiving and sending messages allows for speeding up online conversations. Although at first these applications were intended to be for personal conversations, the group chat feature was also introduced, thus opening doors for group discussions.

However, the speed of these chats that allows the users to quickly ask and get help also is a drawback as useful information is flushed away. Another issue that information seekers could encounter is difficulty to track messages that relate to a topic of interest because in such chats users can discuss several topics simultaneously. Both of these drawbacks lead to a tangle, e.g. a situation when a group chat newcomer asks about a topic that was already discussed. Usually, the only thing he gets is a response about this fact, but not at least a message from which the topic begins. On the other hand, if you want to read messages that are in one dialogue, you have to swipe up unrelated messages. Although some chat applications such as Telegram have a feature to show messages that are connected via replies, users may just not use the "reply to" function that breaks any sense of this feature.

From the scientific perspective, the study of online chats can help us to pursue our understanding of discourse and dialogue phenomenon, as these chats are a rich source of how people are communicating with each other using text and other modalities. The fact that these chats can have several topic discussions simultaneously with intertwined participants makes it a great challenge to automatically analyze them, while humans easily keep track of discussed topics they follow. The investigation of these chats, for example, can help us to develop dialog agents that can actively operate in group conversations rather than in personal ones.

In this work, we investigate the simple relation between messages in these chats, specifically, what message is replying to another. We will refer to this task as reply recovery. We present a model that can automatically predict such relations and try to generalize it to the more complex task of reconstructing separate threads in chats. Our contribution can be summarized as follows:

1. We formalize a task of reply recovery, and provide a representative Russian dataset that we built from Telegram chats of different domains. We also study several methods to solve this task.
2. We create a small benchmark from chats where all "reply to" relations were annotated. We propose a greedy wrapper upon reply recovery model to test its performance in a thread reconstruction setting.

2 Related work

The first work that defined the problem of thread structure reconstruction is (Wang et al., 2008). They used a dataset constructed from forum conversations. The authors proposed a simple unsupervised method that relies on a graph-based text representation. The graph is constructed in a way that all messages are connected to all previous messages, with edge weight calculated as TF-IDF between message texts. Having a complete edge matrix, authors apply a threshold to filter out weak connections. The authors also propose some penalizing strategies based on data observations.

This work started productive research in this direction. The supervised methods appeared that works with emails(Dehghani et al., 2013) and blog comments(Balali et al., 2013). In the work (Louis and Cohen, 2015) authors pointed out a topic as an aspect of thread structure that represents which theme is discussed. They segment messages into several topics and model their treelike structure with different types of context-free grammar.

In the work (Nguyen et al., 2017) the authors adapt a well-known coherence model based on the entity grid in a way that can operate in asynchronous conversation. They pointed out that traditional coherence models can't be applied to this task as they assume a chronological, synchronous flow. On the top of the entity grid, they apply a convolutional network trained with a pairwise ranking loss on choosing the valid

thread tree.
In the paper (Guo et al., 2018) authors for the first time considered the task of predicting "reply-to" relations. They provide two LSTM-based models. The first model use just messaged words, while the second one operates on a sentence level, using the previous model as a core. In addition to forum data, they use data from quick messages systems, in particular WeChat, which is similar to WhatsApp.

In the most recent work (Ji et al., 2021) authors propose a complex solution that tries to capture latent factors such as topic consistency and discourse dependency. To do that, they combine two modules. The first one jointly learns latent topics and discourse, while the second one makes actual predictions about relations.

3 Method

In this work, we consider the task of determining whether one message can be a reply to another. We call this task reply recovery. On top of that, we test simple methods for thread reconstruction. We describe them in separate chapters.

3.1 Reply recovery task

We will use notation and definitions from (Guo et al., 2018). We are given a group chat corpus which is an ordered list of messages $M = \{m_1, m_2, ..., m_N\}$ where N is the total number of messages. The messages come from different users that can participate in several conversations at once. That leads to a situation when messages relating to a particular conversation come asynchronously, so it becomes unclear to say what a particular message responds to. We say $m_i \prec m_j$ if m_j has a "reply to" relation with m_i for $\forall m_i, m_j \in M, m_i \neq m_j$. The task is a binary classification, the objective of which is to predict whether the pair of messages $m_i, m_j, j > i$ has a "reply to" relation.

3.2 Thread reconstruction

Let's assume that messages M contain I threads that we define as a subset of messages $T \in M$ that are related to each other by meaning and thus form a coherent dialogue between users on some topic. The messages from different threads are intertwined as they come asynchronously. So the task of thread reconstruction is to divide the message list M into I threads. We hypothesize that by having a complete map of "reply to" relations, we can get these I threads.

4 Dataset collection

We found out that real chats can be used as a natural source of the data for this task, as people do use the "reply to" feature, explicitly declaring to whom they answer. That could be used as positive labeling. On the other side, it might be thought that the message with its reply should be read coherently. The usual way to make a sequence of text incoherent is to replace some parts with a random sample. We do the same, combining messages randomly.
As a message source, we use a publicly available list of opened Telegram chats. Using the Telegram API, we gather messages from these chats. To ensure the dataset diversity, we manually picked chats with different topics: two chats of women who recently become mothers and who lived in Bali, a chat about football, two chats on IT topic, one house tenant chat, and two teen chats about gaming and suicidal game "Siniy kit" ("Blue whale"). We choose these chats randomly from what we had scraped, except the suicidal game chat, as it is very hard to collect data dedicated to a particular topic. Another criterion was a high ratio of "reply to" messages, although the resulting message counts come with a high disbalance.

4.1 Chat data analysis and preparation

As we have our data, we do a basic analysis to fulfill our interests. First of all, we check the offset between messages and their replies (Fig. 1, left). It can be seen that an offset equal to one (the right after the reply message) is the most common case, and on the other hand, the offset with a distance of 14 is almost diminished. A similar distribution is presented in paper (Guo et al., 2018).
We also examine the number of trees and chains that can be formed from existing "reply to" relations

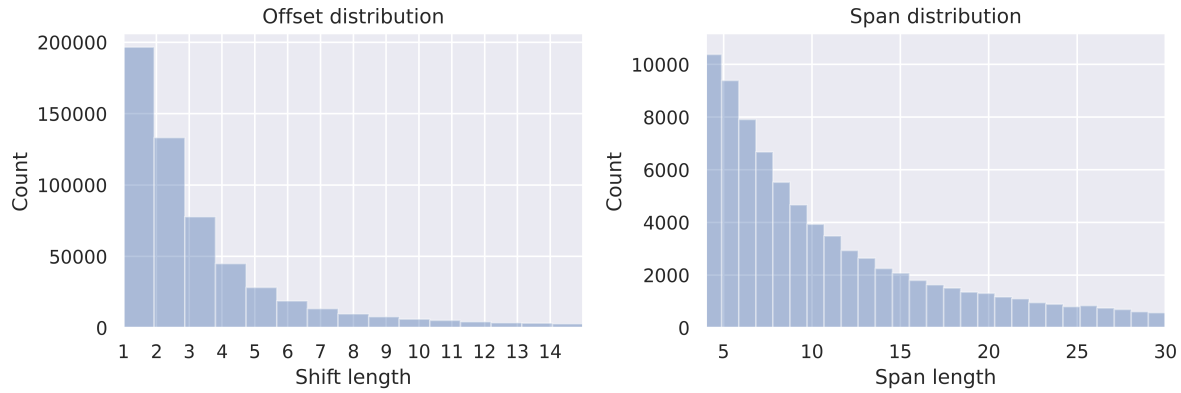

Figure 1: Distributions of message offsets (left) and spans (right) in chats

and related distributions. In the whole dataset, we find 173064 chains and 62067 trees. It's not surprising the superior number of chains have only two nodes, whereas trees have three nodes. At last, we examine the distribution of the distance between the start node and the end node in threads, which we refer to as a span distribution (Fig. 1, right). We exclude threads with less than three nodes as a trivial variant. From this distribution, we can roughly say that most threads lie between 30 messages.

To construct our reply-to dataset, we collect all pairs of messages and their replies. Next, we filter pairs where a message or reply length is less than three symbols and more than 1000. We also filter out pairs that contain more than one Latin symbol pair, as we will use a pretrained model that is not multilingual. We assume these pairs to be positive. As we mentioned above, to make negative pairs, we randomly combine messages with the restriction that they must be from the same chat. We hypothesize that such pairs would be harder than if we pick them from different domains. In total, we have 894997 pairs that we divide in train and evaluation set in proportion 89 and 10 percent. We leave 1 percent for a test set.

4.2 Test set annotation for reply recovery

During the analysis, we figure out that not all explicitly marked message pairs are coherent, and surprisingly not all randomly combined messages are incoherent. To be sure of the good quality of the test set, we decided to annotate it with crowdworkers at the Yandex.Toloka platform that is widely used in post-Soviet country space. For example, it was used to create a large-scale dataset of crowdsourced audio transcriptions named CrowdSpeech (Pavlichenko et al., 2021)

Each task consists of two texts representing the message and its potential reply. The crowdworkers were asked a question if a second text can be a reply to the first or not. The project setup included a restriction of using only the top 10 percent best-rated tolokers and annotation overlap equals to three. The negative and positive examples are annotated separately. While we are aware that such a setting could make an annotator biased, we didn't figure out how to preserve identification information on the platform to be able to map examples with a preannotation. By the end of the annotation process, we got inter-annotator agreement by Krippendorff's alpha (Krippendorff, 2011) for negative being equal to 0.182 and for positive being equal to 0.280. We think that such a low score relates to the noisy nature of the crowdsource, as the task was pretty simple. In addition, we didn't perform an annotator training procedure. Actually, we rely on the Dawid-Skene model (Dawid and Skene, 1979) as a method of obtaining true labels from noisy crowd labels. After completing a pilot start, the investigation of the model confidence result distribution shows that the sufficient part of examples has a very high confidence score, although the distribution has a notable tail. We establish a threshold of 80 percent, above which we consider the example annotation to be reliable. Another finding is that there are some negative examples out of a positive set. Some of them have a strong confidence that's of high interest. A similar picture is observed with the negative set.

Having all annotations completed, we manually validate messages that were supposed to be negative but were annotated as positive and vice versa. We also examine messages that have weak confidence. We can highlight the reason why that occurs: foreign language, domain misunderstanding, and, in particular, meme phrases, uninformative messages, and, at last, simple mistakes.
In the end, we got 4693 positive and 3997 negative messages with balanced domain distribution, although one category has slightly more messages due to data management mistakes.

4.3 Test set annotation for thread reconstruction

As we intended to use our model to thread reconstruction, we create a small test that consists of chats from the above-mentioned topics. We randomly pick 10 slices from each topic with 100 messages in range. We ensure that slices do not intersect and that all messages contain text. As the task is much harder than just deciding whether two texts are related and, to the best of our investigation, the technical restriction of the platform, we hire two annotators to manually label all connections in each chat using the Label Studio platform (Tkachenko et al., 2020). The main criterion of messages being connected is coherence between them. We conduct a workshop where annotators were instructed to keep in mind that related messages should have a meaning if we would discard all other messages, and also that related messages have common words. It's important to note that unlike in some other works, we assume that the response has only one head. However, some messages have a clear sense of addressing many chat participants. These messages were asked to mark as self-connected. It contradicts the work (Guo et al., 2018) where messages are self-connected if they have no replies. In our schema, such messages just didn't have annotation.
Another notable phenomenon is several messages followed by one another from one user. The annotators were instructed to connect these messages, except when some message replies to a message outside the monologue. In the end, we got 89 annotated dialog slices with 100 messages in each sample. It has to be noticed that the random manual inspection of the dialogues shows that the annotation is quite noisy.

5 Models description

5.1 Reply recovery

As a starting point, we use **Logistic regression** on top of concatenated vectors of texts within the pair. The vector representation is a term-document matrix. The hypothesis behind the use of such a simple model is the fact that related messages often reuse words that represent a subject or object. This is the ground of entity grid representation(Barzilay and Lapata, 2008).
Following the (Guo et al., 2018) we train **LSTM** model on word level, but we augment it with self-attention mechanism (Vaswani et al., 2017). Given the model of concatenated text pairs, we expect that the attention mechanism allows the model to learn better discourse dependencies. We exclude the hierarchical sentence level variant from those words, as it didn't give notable gain.
The defined task aligns closely with the next sentence prediction (NSP) loss that is used to train the BERT model (Devlin et al., 2019). We can assume that messages are often about one sentence, though we assume that integrating more than one sentence is an applicable strategy.

We use **Conversational RuBERT**[2] (ConvBERT) that is fine-tuned on social media texts RuBERT (Kuratov and Arkhipov, 2019). It's important that it has a vocabulary based on this data. In turn, we fine-tuned the Conversational RuBERT on our data with NSP loss with 3 epochs and 1e-5 learning rate.
Thinking about how the model will be used in a thread reconstitution task and the fact that two related messages should be coherent, we try to use **Siamese architecture** (Neculoiu et al., 2016) with different bases. As a reminder, the Siamese network architecture consists of two identical basic networks with shared parameters. In the usual setting, they consume two objects and the main goal is to distinguish the difference between them. Usually, it trains with a contrastive or a triplet loss. In our work, we use LSTM and our fine-tuned ConvBERT as the basic network. The latter architecture is also known as SentenceBERT or SBERT (Reimers and Gurevych, 2019). Instead of the above-mentioned losses, we use CrossEntropyLoss as our task is a classification.

[2] `http://docs.deeppavlov.ai/en/master/features/models/bert.html`

As can be seen in the next section, ConvBERT shows the best result but due to our approach to thread reconstruction and as a matter of fact that BERT-like models are computationally expensive, we also experiment with a distilling knowledge from ConvBERT into LSTM by the teacher and student paradigm (Tang et al., 2019). This would allow us to disentangle embeddings for two messages and reduce computations. The question is whether a quality loss would be affordable.

5.2 A note about tokenization

In social media, people tend to reduce some words and make abbreviations of frequent phrases to save typing time. This phenomenon can be seen in this work (Buyanov and Sochenkov, 2022) where authors study the language of Twitter posts of persons with suicidal tendencies. Another thing is that people try to mimic speech methods of emotional expression, thus they multiply vowels ("whaaaat?"). In particular, Russian speakers express laughter like "axaxax" ("ahahah"). Due to a combination of different sequence lengths, sequence permutation errors, and mistyping (sometimes intended) the vocabulary of laughing can be very large, and the normalization of this vocab is a tough task. All described facts lead to an enormous vocabulary size, where many entities have a low frequency. So we think that using the BERT tokenizer, which works on a subword level, with Logistic Regression and LSTM models could be beneficial in terms of vocab efficiency.

In our experiments, we compare two ways of tokenization. In first one is a using tokenizer that does not use subword tokenization. For that purpose, we use a tokenizer from NLTK and will refer to it as **simple tokenizer**. In a second way, we use the **BERT tokenizer** and will refer to it as its name.

5.3 Thread reconstruction method

As a baseline, we consider a dummy heuristic that connects the present message with the previous one. Following our assumption, we need to reconstruct all “reply to” relations that would reveal threads. Having a model that can predict the relation between two messages, we would apply this model to all pairs of messages inside a chat of length N. Although, with our models, it is prohibitively expensive as we need to proceed a $O(N^2)$ pairs. Likely, from the data analysis we know that the biggest part of the pairs have a distance of less than 15 messages, so we can restrict the search space to this count reducing the count of pairs to $O(15 * (n - 15))$.

It is worth noting that we would recompute the embeddings of almost every text 15 times. While using the Siamese network, we can escape this computational overhead by precomputing embeddings of all texts with the basic network. Having a matrix, we then can just use classification head on pairs of rows, which is much cheaper to compute. Unfortunately, to compute BERT in the NSP regime we have to form a string with a special format, so here we can’t escape these expenses.

We organize the predictions in the $N \times N$ probability matrix, from which we must derive a valid adjacency matrix. We experiment with two simple heuristics. The first is to keep the nearest predicted reply w.r.t. current message position. For example, if the 7th message 2nd, 5th, and 6th messages were predicted as replies, then we chose the 6th. We will call this *greedy binary*. The second heuristic is to choose the message as a reply that has the highest probability score among all possible ones. We will call it *greedy probas*. Moreover, we can variate how many messages before the current position we will consider. Having the processed matrix, we can derive threads as a collection of connected graph components. We also can use this matrix to compare it with human annotation.

6 Results

6.1 Reply recovery

As we have a standard classification task, we report precision, recall, and F1 score. In Table 1 the result on the entire dataset is presented. We can see that ConvBERT shows the best result overall models. The simple linear regression fails to discover any useful relations to tackle the problem. Another observation is that Siamese architectures perform worse and for the LSTM the performance drop is very significant compared to SentenceBERT. We also see that using the BERT tokenizer for the LSTM model is beneficial compared to simple token dictionary tokenization. As for distillation, we see that model fails to learn

knowledge from BERT. We leave the study of why to further research, but it's probably a good idea to focus on DistllBERT (Sanh et al., 2019) as a base block for SentenceBERT.

An important observation is that the model performance depends on a chat domain. In Table 2 we see the models variate significantly. For example, the ConvBERT model ranges up to 10 score points. We think that one of the factors is a chat topic and goal that influences symbol length. The chats where participants ask questions or discuss complex living situations tend to have more message symbol length, like *sling38* where participants are young mothers. Conversely, game chats are about fun which is not required to write a lot. The Pearson correlation between average pair length and ConvBET results is 0.62 with a p-value of 0.076. Another factor is probably the lexicon specificity of chats. However, these factors do not cover all cases. Although, *cotedazuchat* is a chat of Russian emigrants in France where they discuss various topics, and it has medium average text length compared to others, it has the worst performance score.

Model name	**Precision**	**Recall**	**F1**
with BERT tokenizer			
LSTM	0.651	0.719	0.628
Siames LSTM	0.668	0.491	0.539
Logistic regression	0.475	0.529	0.501
with simple tokenizer			
LSTM	0.602	0.606	0.566
Siames LSTM	0.507	0.537	0.505
Logistic regression	0.474	0.528	0.500
ConvBERT	**0.822**	**0.846**	**0.833**
DisilledLTSM	0.459	1.000	0.630
SentenceBERT	0.786	0.838	0.797

Table 1: Model scores overall reply recovery test set.

Chat name	Avg pair len	LSTM-ST	SLSTM-BT	LSTM-BT	SLSTM-ST	SBERT	ConvBERT
balichat_woman	140.948	0.695	0.688	0.659	0.580	0.857	**0.883**
borussia_chat	81.697	0.585	0.627	0.605	0.568	0.798	**0.821**
chat_suicidnikov	72.400	0.538	0.576	0.579	0.533	0.784	**0.826**
cotedazurchat	93.785	0.575	0.628	0.588	0.536	0.762	**0.793**
easypeasycodechat	94.804	0.702	0.519	0.555	0.278	0.859	**0.885**
openwrt_ru	101.410	0.556	0.620	0.566	0.527	0.788	**0.885**
orange_sosedi	151.584	0.625	0.647	0.658	0.533	0.817	**0.849**
sling38	174.404	0.686	0.655	0.668	0.512	0.842	**0.890**
terrariaphone	69.732	0.578	0.624	0.618	0.544	0.801	**0.841**

Table 2: Model scores for different chat domains. BT is the BERT tokenizer, ST is the simple tokenizer. The best score is in bold, and the worse one is underlined.

6.2 Thread reconstruction

As we can see from Table 3 none of the proposed heuristics can beat the baseline. Comparing two proposed heuristics, we could say that the selection of the highest probability generally performs worse than taking the first predicted message. Another note is that with increasingly considered messages, *greedy probas* performers significantly worse than *greedy binary*. The probable explanation is that *greedy binary* is aligned with the observation that in general, the reply is a previous message. In contrast, in *greedy probas* relies on predictions of not so excellent model. Based on these results, we see that simple heuristics are not enough to restore threads with the reply recovery model of the proposed quality. We hypothesize that the model for thread reconstruction should consider the context of the dialogue, or the

reply recovery model should have better performance.

Model name	Precision	Recall	F1
dummy	0.776	**0.814**	**0.793**
greedy_binary_15	0.740	0.749	0.743
greedy_binary_3	**0.799**	0.735	0.762
greedy_binary_7	0.753	0.747	0.749
greedy_probas_15	0.640	0.645	0.642
greedy_probas_3	0.768	0.710	0.735
greedy_probas_7	0.680	0.676	0.677

Table 3: Scores of methods on thread reconstruction test set

7 Conclusion and future work

In this work, we investigate the asynchronous chats in the Russian language. We show that modern pre-trained models show great results on the task of reply recovery compared to more simple models. We also experiment with a thread reconstruction task based on restored "reply to" relations with simple heuristics. The results show that these heuristics are not enough, despite the relatively good result of the model.

As a future work, we can highlight the investigation of pipelines that would produce a dataset of better quality, as we found that using replies only and random sampling is not guaranteed to have good labeling. Possibly weak supervision approach can be used to better filter out broken pairs. Speaking of thread reconstruction, our small benchmark is far from ideal, so it certainly can be improved. Also, using the reply recovery model, one can create a noisy train dataset on thread reconstruction. Such datasets can reduce the cost of annotation of large-scale data. From the model point of view, the graph neural networks can probably benefit from the natural graph structure of chat conversations.

Acknowledgements

We thank our annotators Anastasiya Dyachenko and Alexander Lugarev for their work for the thread reconstruction dataset. The research was funded by the Ministry of Science and Higher Education of the Russian Federation in accordance with agreement № 075-15-2020-907 of 16.11.2020. The grant was provided for state support for the creation and development of a World-class Scientific Center "Agro-technologies for Future".

References

Ali Balali, Heshaam Faili, Masoud Asadpour, and Mostafa Dehghani. 2013. A supervised approach for reconstructing thread structure in comments on blogs and online news agencies. *Computación y Sistemas*, 17.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34:1–34.

Igor Buyanov and Ilya Sochenkov. 2022. The dataset for presuicidal signals detection in text and its analysis. *Computational Linguistics and Intellectual Technologies*.

A. Philip Dawid and Allan Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of The Royal Statistical Society Series C-applied Statistics*, 28:20–28.

Mostafa Dehghani, Azadeh Shakery, Masoud Asadpour, and Arash Koushkestani. 2013. A learning approach for email conversation thread reconstruction. *Journal of Information Science*, 39:846 – 863.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Gaoyang Guo, Chaokun Wang, Jun Chen, and Pengcheng Ge. 2018. Who is answering to whom? finding "reply-to" relations in group chats with long short-term memory networks.

Lu Ji, Jing Li, Zhongyu Wei, Qi Zhang, and Xuanjing Huang. 2021. Who responded to whom: The joint effects of latent topics and discourse in conversation structure. *ArXiv*, abs/2104.08601.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

Yuri Kuratov and Mikhail Arkhipov. 2019. Adaptation of deep bidirectional multilingual transformers for russian language. *ArXiv*, abs/1905.07213.

Annie Louis and Shay B. Cohen. 2015. Conversation trees: A grammar model for topic structure in forums. // *Conference on Empirical Methods in Natural Language Processing*.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. // *Rep4NLP@ACL*.

Tien Dat Nguyen, Shafiq R. Joty, Basma El Amel Boussaha, and M. de Rijke. 2017. Thread reconstruction in conversational data using neural coherence models. *ArXiv*, abs/1707.07660.

Nikita Pavlichenko, Ivan Stelmakh, and Dmitry Ustalov. 2021. Crowdspeech and vox diy: Benchmark dataset for crowdsourced audio transcription. // J. Vanschoren and S. Yeung, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, abs/1908.10084.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. 2020. Label Studio: Data labeling software. Open source software available from https://github.com/heartexlabs/label-studio.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Yi-Chia Wang, Mahesh Joshi, William Cohen, and Carolyn Rosé. 2008. Recovering implicit thread structure in newsgroup style conversations. 01.