

Master Thesis

November 16, 2022

Detecting Related Stack Overflow Posts for Discord Conversations

Artemis Ioanna Kardara

of Athens, Greece (17-746-082)

supervised by

Prof. Dr. Thomas Fritz

Alexander Lill



University of
Zurich^{UZH}



HASEL

Detecting Related Stack Overflow Posts for Discord Conversations

Artemis Ioanna Kardara



University of
Zurich^{UZH}



Master Thesis

Author: Artemis Ioanna Kardara, artemisioanna.kardara@uzh.ch

Project period: May 2022 - November 2022

Human Aspects of Software Engineering Lab
Department of Informatics, University of Zurich

Acknowledgements

I want to thank Alexander Lill for his continuous support and guidance, as well as Prof. Dr. Thomas Fritz for the opportunity to work on this thesis under the HASEL research group. I am also grateful to my family and my partner for their endless help and support throughout the duration of my studies.

Abstract

Programming Question-Answer Communities are at the forefront of modern Developer Work. While existing research has historically focused on finding duplicate posts in Stack Overflow, there is currently little to no research for near-duplicate detection which focuses on Platforms like Discord. To address this gap, we consider existing state-of-the-art approaches in the field and additionally replicate and apply the most promising to the new domain. A Discord-Stackoverflow dataset in the Java domain is constructed and utilized, while for the evaluation both a classification and retrieval task are considered. The experimental results show that a direct transfer from an existing domain and model is feasible to an extent, while classification and retrieval in the new domain reach up to 77% *Precision*, 70% *F1-Score*, and 80% *Recall-Rate* depending on the length of the examined Input Sequence.

Zusammenfassung

Das Programmieren von Frage-Antwort-Communities ist an der Spitze der Liste von moderner Entwicklerarbeit. Während sich die bisherige Forschung auf das Auffinden von doppelten Beiträgen in Stack Overflow konzentriert hat, gibt es derzeit wenig bis gar keine Forschung zur Erkennung von Beinahe-Duplikaten, die sich auf Plattformen wie Discord konzentriert. Um diese Lücke zu schließen, betrachten wir bestehende State-of-the-Art-Ansätze in diesem Bereich und replizieren die vielversprechendsten und wenden sie auf den neuen Bereich an. Ein Discord-Stackoverflow-Datensatz im Bereich Java wird erstellt und verwendet, während für die Evaluierung sowohl eine Klassifizierungs- als auch eine Retrieval-Aufgabe betrachtet wird. Die experimentellen Ergebnisse zeigen, dass ein direkter Transfer von einem bestehenden Bereich und einem bestehenden Modell bis zu einem gewissen Grad möglich ist, während die Klassifizierung und das Retrieval im neuen Bereich bis zu 77% *Praezision*, 70% *F1-Score* und 80% *Recall-Rate* erreichen, abhängig von der Länge der untersuchten Input-Sequenz.

Contents

1	Introduction	1
1.1	Motivation	3
1.2	Research Questions	4
1.3	Background	4
1.3.1	Natural Language Processing	4
1.3.2	Natural Language Understanding	5
1.3.3	Semantic Relatedness Tasks	6
1.4	Thesis Outline	7
2	Related Work	9
2.1	Feature-Based Approaches	9
2.2	Deep Learning Approaches	10
2.2.1	Bi-Directional LSTM Models	11
2.3	Available Datasets	12
2.3.1	WikiQA	12
2.3.2	Quora Question Pairs	12
2.3.3	Communication Platform Datasets	14
3	Approach	15
3.1	ASIM: Attention-Based Sentence Pair Interaction Model	15
3.2	SRDM: Semantics and Relevance Duplicate Questions Detecting Model	15
3.3	Common Architecture Elements	17
3.4	Chosen Solution	19
3.5	Evaluation Metrics	20
4	Model Replication	23
4.1	Dataset Exploration	24
4.2	RQ1: Model Reproducibility	26
4.3	Preprocessing	26
4.3.1	Misspelled Words	26
4.4	Model Architecture	27
4.5	Model Training	28
4.6	Replication Results	28

5	Detecting Discord and Stack Overflow Pairs	33
5.1	Discord Dataset Construction	33
5.1.1	Conversation Extraction	33
5.1.2	Mining of Discord-SO Pairs	34
5.1.3	Annotation of Conversation Pairs	36
5.1.4	Data Cleaning	38
5.2	RQ2: Classification Task	39
5.2.1	Results	40
5.3	RQ3: Semantic Retrieval Task	44
5.3.1	Results	45
5.4	RQ4: Conversation Length Variability	46
5.4.1	Results	46
5.4.2	Retrieval Examples	47
6	Discussion	51
6.1	Limitations	51
6.2	Threats to Validity	52
6.3	Toward an Efficient, Real-Time Neural Retrieval Application	53
7	Conclusion and Future Work	55
7.1	Conclusion	55
7.2	Future Work	55
8	Acronyms	57
A	Implementation Details	61
A.1	Stack Exchange API	61
A.2	Discord Retrieval Runs	62
B	Discord-SO Conversation Examples	63

List of Figures

1.1	Number of Public Discord Servers by Tag.	2
1.2	Natural Language Processing (NLP) and Natural Language Understanding (NLU) task distinction via Stanford [Mac14].	5
3.1	The Attention-Based Sentence Pair Interaction Model (ASIM) Architecture as presented by [PWQ ⁺ 21].	16
3.2	The Semantics and Relevance Duplicate Questions Detecting Model (SRDM) Architecture as presented by [LLZY21].	17
4.1	Count of nodes per connection degree - Distribution of question occurrence in the dataset - x: Frequency of question - y: Question appearance counts	24
4.2	Example of Knowledge Unit (KU)s in the Stack Overflow corpus.	25
4.3	Confusion Matrices of Model Experiments.	30
5.1	Conversation statistics of Discord Community.	35
5.2	Conversation statistics of Discord dataset	37
5.3	Confusion Matrix of Discord-SO Dataset Classification	40
5.4	Example of the retrieval score calculation as defined by the Pytorch framework. . .	44
B.1	Discord Conversation #977541076548222976	64
B.2	Stack Overflow Question #3699141	65
B.3	Discord Conversation #979813430401835078	66
B.4	Stack Overflow Question #56728398	67
B.5	Discord Conversation #971797369874182194	68
B.6	Stack Overflow Question #33595941	69
B.7	Discord Conversation #963170643640213576	70
B.8	Stack Overflow Question #43130952	71
B.9	Discord Conversation #904644087200755723	72
B.10	Stack Overflow Question #223918	73
B.11	Discord Conversation #959725068391432212	74
B.12	Stack Overflow Question #71714015	75
B.13	Discord Conversation #966404222058893322	76
B.14	Stack Overflow Question #13102045	77
B.15	Stack Overflow Question #17443201	78
B.16	Stack Overflow Question #12273794	79
B.17	Stack Overflow Question #12544564	80
B.18	Stack Overflow Question #36096097	81
B.19	Stack Overflow Question #4185320	82
B.20	Stack Overflow Question #7487341	83
B.21	Stack Overflow Question #13056980	84
B.22	Stack Overflow Question #11328043	85
B.23	Stack Overflow Question #12888780	86

List of Tables

4.1	Knowledge Unit Dataset Statistics (Shirani et al.).	23
4.2	ASIM Replication results.	29
4.3	F1 score of ASIM and Model Variations.	31
5.1	Classification Evaluation of Datasets using Complete Conversations as Input and the ASIM-RV3 model	40
5.2	Examples of Discord-SO pairs, along with their assigned class(label) and the predicted class by ASIM-RV3	42
5.3	Examples of Discord-SO pairs that were misclassified by ASIM-RV3	43
5.4	Retrieval Recall Performance of the best performing model (ASIM-RV3) on the Discord-SO dataset and the Stack Overflow dataset.	45
5.5	Classification and Retrieval Evaluation of Different Conversation Lengths.	47
5.6	Example: Retrieval Results Using Short Conversation Query	49
5.7	Example: Mining Results Using Short Conversation Query	50
A.1	Retrieval Evaluations of Discord-SO pairs. For each run the Retrieval Dataset is enriched by Random Subsets of Stack Overflow Knowledge Units.	62

Introduction

Knowledge sharing via Question and Answer (Q&A) websites is quite popular amongst professionals and enthusiasts alike. In Software Engineering early such knowledge sharing was performed in digital bulletin boards [Dri16] and web forums where people held conversations in the form of posted messages and replies. In 2008 a new form of Q&A platform was created currently known as Stack Overflow (SO)¹. A combination of specific features made the SO website quite popular and motivated people to participate. It is easily accessible by anyone without the need to have an account to read and search for questions. The questions themselves can span a variety of different topics and cover any interest and knowledge level. Participation is encouraged by a voting feature where people upvote or downvote specific answers and users earn reputation points². In addition, digital badges are awarded and more privileges are granted with a higher reputation such as voting, commenting, and even editing other people's posts (mainly used to fix formatting and typos)³. This meticulous reputation and privileges system has made SO the most popular website related to programming questions.

Due to the website's widespread use, the existence of duplicate questions and their detection was observed and led to a highly interesting and researched topic, with the first publications dating back to 2015. This characteristic can be viewed as another of its differentiating factor, i.e the effort that is involved with the detection of duplicate questions and the suggestion system that shows related questions to the users (often already answered). It also helps provide centralized quality answers to common questions with the ability to have more than one answer in the conversation (answers are sorted based on the votes they received). SO has an established way of marking and providing its users with marked duplicate and related questions. When a question is identified as duplicate, it is closed and does not accept further answers. Moderators, users with a gold badge, and users with over 3000 reputation points can mark questions as duplicate and close them, an action that needs up to 2 additional votes to be executed. Each of the mentioned authorized users could potentially mark up to 50 duplicate questions per day. It can be therefore inferred that manually detecting duplicate questions requires a great deal of manpower and hours, and yet it inevitably allows for many more unidentified duplicate questions.

Naturally, it is quite common for duplicate question pairs to not get identified concurrently as the second question is being asked, rather it can be many days or even weeks later. Additionally, the conversations for each question provide a treasure of knowledge as users explain in detail answers and concepts, as well as link other relevant questions and thus effectively connecting concepts and methods. Therefore, the need for automated duplicate detection is apparent and the benefits to be gained are plenty.

¹<https://stackoverflow.com/>

²<https://stackoverflow.com/help/whats-reputation>

³<https://stackoverflow.com/help/privileges>

Since the launch of SO until now, a plethora of well-known Community Question Answering (CQA) sites has emerged, to facilitate communication and question-answering between people such as Quora, Yahoo! Answers, and countless forums, as well as communication platforms such as Slack, Discord, and Gitter. The more organically structured communication platforms like Discord, in particular, are frequently visited by developers to achieve arguably more direct communication, while the instant and informal conversations allow for a tighter formation of communities.

It should not come as a surprise that Discord's motto as a communication platform is "Your Place to Talk and Hang Out", and is one that effectively creates the space for community-driven communication. Its online spaces have multiple features that provide great flexibility, from public channels to private chats, to voice calls and video. Every individual can be a part of multiple communities at once and can choose to blend one in the background while actively taking part in another.

The platform has seen a great increase in traffic in recent years. In 2020, the platform's monthly active users "nearly doubled from the prior year to 100 million" and "went up an additional 40% in 2021 to 140 million" ⁴. In addition, according to Discord ⁵, there is a great number of public servers tagged with technology and development tags. To better illustrate the extent of public Discord communities, Figure 1.1 shows the number of public servers that are marked as development related through a variety of tags.

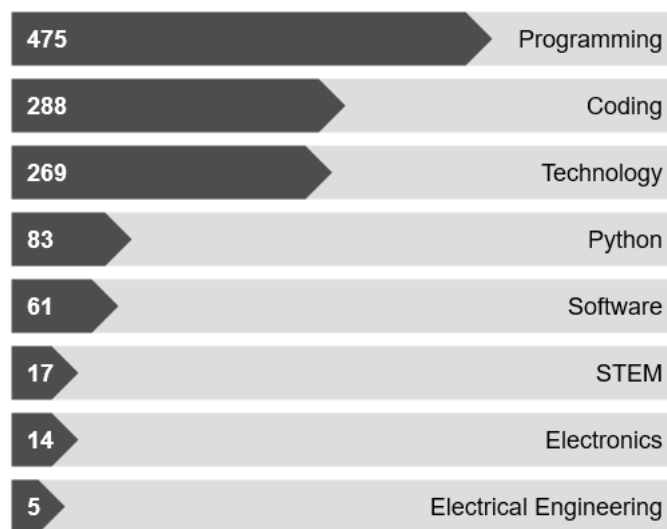


Figure 1.1: Number of Public Discord Servers by Tag.

However, the previously mentioned SO features have not been replicated in such platforms yet. Thus, it can be argued that their programming communities could greatly benefit from similar functionality. This holds especially true in the case of novice members in programming and question-answering communities who tend to ask guidance on fundamental principles and frequently asked questions that could be easily answered historically by an existing post. Simultaneously, such functionality could certainly save community hours and capacity for more intricate

⁴<https://www.thinkimpact.com/discord-statistics/#5-discord-usage>

⁵<https://discord.me/servers/>

and novel inquiries, without excluding such answers from a near duplicate detection either.

1.1 Motivation

Undeniably, CQA websites and platforms are indispensable tools for professional programmers and coding novices alike and there is nothing that suggests their usage subsiding in the near future. However, while there is a rich bibliography for websites like SO and Quora, there is little to no research available, to the best of our knowledge, for other communication platforms. Indeed, duplicate questions can frequently be observed in coding-related channels of communication platforms and team messaging applications like Discord, Slack, and Gitter. In some cases, duplicate answers are identified and handled manually by the moderators, or in most cases not at all. In smaller team channels that might not be a concern, but as a community grows so does the need for the laborious effort of a moderator. Therefore, automatic duplicate detection can be beneficial to decrease the required spent time by moderators, as they can simply review a detected duplication and take action if needed. In a more automated scenario, bots can also monitor the conversations and suggest the most relevant posts based on the questions being asked and the unfolding conversations. As a result, questions and issues are resolved faster and no question is left without an answer due to increased user traffic and surpassed capacity. In addition, smaller communities can benefit from the automatic suggestion of a handful of similar conversations that can be manually reviewed easily, considering that when collective domain expertise is lacking, responding to all incoming questions can become especially challenging. Lastly, in the case where suggestions are rated in terms of helpfulness, a reinforcement task scenario could realistically improve the results over time in new domains where the question knowledge base is not built yet.

The lack of functionality in these communities similar to the functionality observed in SO can be inherently explained by their difference in structure. Chatterjee et al. [CDP⁺19, p. 490], who explored Slack Q&A chats, convincingly state that “while chat communities share some commonalities with other developer communications, they also differ in intent and use. Overall, Q&A forum content is archival, while chat community content is transient. Q&A forums encourage longer, more in-depth questions that receive one or more well-thought-out answers. The question and its answers are subsequently read by many developers”. They continue by explaining that, conversely, developer chats are characterized by a typically more informal structure that is conversation based and involves “rapid exchanges of messages between two or more developers, where several clarifying questions and answers are often communicated in short bursts”. Therefore, chat-based communities contain shorter responses that are more often than not meshed with non-information-providing messages. Indeed, the inherent nature of chat conversations means that they should be considered short-lived, and in the case of Slack are often only available for a certain amount of time that is depending on the space quota of each community. Although it cannot be necessarily observed directly, similar questions are repeated and answered by members frequently, especially with time and as the community grows.

Therefore, the main goal of this thesis is to investigate if and how well, approaches that are commonly used for Q&A platforms and more specifically SO, can perform in informal developer conversations. To achieve that, the available approaches will be considered and the state-of-the-art approach will be replicated and applied to extracted Discord conversations. The outcome of this work is to automatically identify duplicate or near-duplicate questions of a Discord conversation paired to a SO post with the highest precision possible.

1.2 Research Questions

Given the above, we can define the research questions that this thesis will aim to answer.

RQ1: Can the results of current state-of-the-art approaches in SO duplicate detection be successfully reproduced?

To answer the question, available research will be taken into consideration, the most fitting approach will be selected and replicated. The successful reproducibility will be addressed using the same measures of performance as the original approach.

RQ2: How does the identified methodology perform on the new dataset and how do the results compare to the reproduced results?

In other words, can we perform knowledge transfer to the fairly new domain of Discord? How do the conversations occurring on the two platforms relate to each other in terms of their characteristics? To answer the question, a Discord dataset will be constructed, explored, and the structure defined to best facilitate the domain transfer task.

RQ3: Can the same methodology be used for semantic question retrieval?

The motivation for this task is the aspiration of aiding developer communities with their help queries. Thus, we seek to answer the question of whether the identified state-of-the-art (SoA) solution can be adapted and applied to a semantic retrieval task. And if the adaptation is possible, what is the observed performance?

RQ4: How is model performance affected when different parts of the conversation thread are considered as input?

This research question is motivated by the fact that conversations on communication platforms are characterized by a more informal structure that can scatter query information across multiple messages. Thus we seek to understand the effects that different parts of the conversation have on the duplicate and retrieval results.

1.3 Background

In this section, the theoretical background of foundational concepts needed to comprehend the work in this thesis is explained. Artificial Intelligence (AI) concepts such as NLP and NLU applications are covered.

1.3.1 Natural Language Processing

NLP uses machine learning to reveal the structure and meaning of the text. The origins of NLP can be found in Turing's paper "Computing Machinery and Intelligence" which proposed the famous Turing test. As part of the Turing test, there is a task that involves artificial interpretation and generation of natural language [TUR50]. NLP has evolved from **symbolic NLP** (1950s - early 1990s), to **statistical NLP** (1990s-2010s) to **neural NLP** (present) [GM86] [MS08] [Gol16].

NLP can be utilized on several high-level use cases. Some popular use cases identified by industry pioneers of AI and NLP IBM [Edu] and Google [goo] include text summarization, content and sentiment classification, and machine translation.

1.3.2 Natural Language Understanding

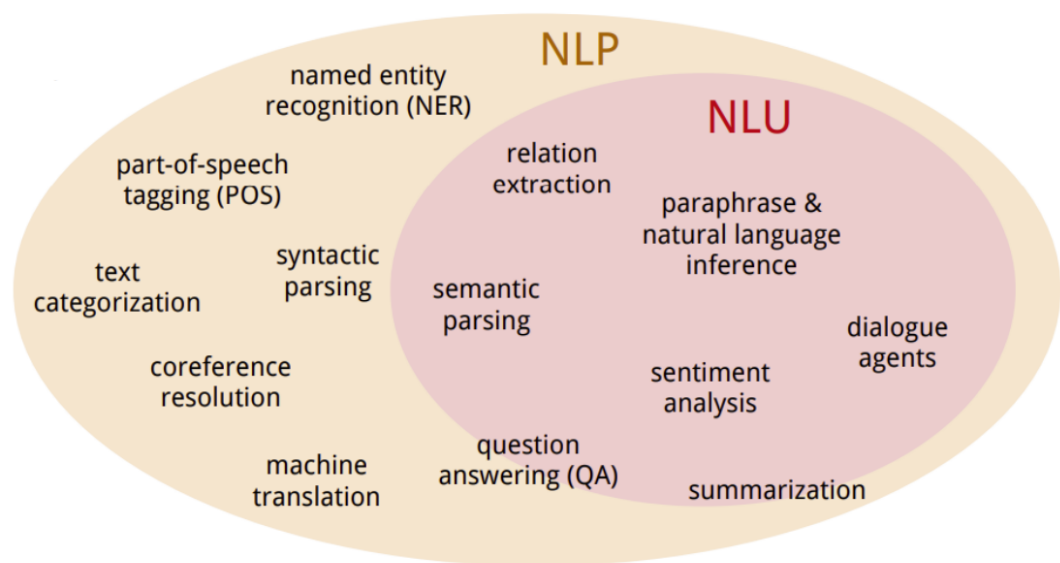


Figure 1.2: NLP and NLU task distinction via Stanford [Mac14].

As a subtopic of NLP, NLU uses syntactic and semantic analysis of text and speech to determine the meaning of a sentence under a specific context. As seen in the above Figure 1.2 of Subsection 1.3.2, NLU goes one step further than the literal natural language processing performed by NLP tasks and attempts to analyze the deeper meaning depending on the context of a sentence, conversation or question and answering dialogue.

The main focus of NLU is to use reading comprehension and enable machines to interpret the natural language, derive meaning, identify context, and draw insights.

Since NLU is a subtopic of NLP often times the separation of the two is unclear because most applied methodologies use both of them to achieve the desired outcome. An intuitive way to make a distinction between the two is that NLP can work with unstructured data and transform it into a more structured or desired format. NLU is not concerned with that part and focuses on subsequent tasks of deriving semantic meaning from the text under a different context. Certainly, it is their combination, that offers the greatest results and scientific value. [IBM] [KS21] [SCA]

The last important subtopic of NLP, is Natural Language Generation (NLG). It focuses on constructing text output in the format of human-understandable language and can involve the generation of speech through text-to-speech applications.

1.3.3 Semantic Relatedness Tasks

The purpose of semantic tasks is to perform an analysis of the structure of text, sentences, and word interactions, and based on the output of this analysis to discover the related contexts in text. Research provides a plethora of examples and definitions of tasks that aim to capture the specific considerations of each one. Below we present the most prevalent ones, that will be used as a basis for the definition of this thesis' description of work.

Duplicate and Near-Duplicate Classification

A fundamental approach when it comes to NLP and AI in general, is the classification task. In its generic form, it can involve tasks that classify text into different classes, for instance, express sentiment, subjectivity and objectivity, question type, and opinion polarity to name a few [CKS⁺17]. In the duplicate question scenario, we want to know whether two given questions are duplicates or not. This usually requires a binary classification task, which can output an assigned probability to the positive class, i.e a probability that two text sequences are duplicates or not. [TRR⁺21] [ZLXS15]

Paraphrase Mining

Duplicate phrase or paraphrase mining is the task of identifying all duplicate and near duplicate question pairs given a large corpus of questions. This task usually translates to unsupervised or transfer learning tasks, where knowledge from a studied domain is used in a new domain. [MFdlC⁺20]

Semantic Textual Similarity

The general goal of Semantic Relatedness and Semantic Textual Similarity (STS) tasks is to find how similar different sentences are in terms of meaning. The context of the task can be questions and phrases, but also longer text like paragraphs and whole documents. Usually, STS is used as a basis for other tasks. For instance, semantic search, passage ranking, clustering, and information retrieval for documents utilize STS. [JHC⁺20] [MTMR22]

Neural Information Retrieval

Information Retrieval of Duplicate Questions [MC17], [GFP⁺20], [ZRB⁺16] refers to retrieval based on techniques that have been used for search engines and classic retrieval tasks and, in more recent years, Information Retrieval (IR) based on deep neural networks that usually incorporate semantic search. In fact, the terms can usually be used interchangeably.

In general, though, the task definition aims to identify the most related questions that exist in the set, given a not previously seen question (or query) and a large data set of questions. The distinction between classic IR and Semantic Search is that the former is based on finding matches using lexical features and similarities, while the latter seeks to improve search by understanding the content and context of the search input.

Researchers of Sentence Bidirectional Encoder Representations from Transformers (SBERT) [Rei] note a further distinction in the definition of a semantic search task, which is the difference between Symmetric and Asymmetric Semantic Search. **Symmetric Search** assumes that the search query and the entries in the corpus have relatively the same lengths. For instance, the search is for single sentences or questions within the corpus. Conversely, **Asymmetric Search**

receives a relatively short query and seeks to retrieve paragraphs or whole documents. Depending on the task, neural retrieval requires a large amount of training data with a size starting from approximately 100k+ pairs.

1.4 Thesis Outline

This thesis will examine the approach of **duplicate question classification** which is the basis of the majority of previous work on SO posts. It aims to examine the reproducibility and transferability of the state-of-the-art approach that best fits the domain of Discord Conversations. Additionally, it will examine how the identified approach can be utilized for near-duplicate question retrieval.

The thesis report is structured as follows:

Chapter 2 details related work in SO duplicate question detection, such as Machine Learning approaches and Deep Learning Models.

Chapter 3 contends the state-of-the-art approaches and explains the selection criteria of the chosen model.

Chapter 4 presents the methodology and process of replicating the model, and the process of creating the Discord Dataset to evaluate the transferability.

Chapter 5 presents the evaluation metrics and the results of the research questions.

Chapter 6 discusses the results as well as the limitations and validity of the work.

Chapter 7 concludes this work with final considerations and identifies future work.

Appendix A provides additional implementation information.

Appendix B provides a more complete overview of examined conversations.

Related Work

Duplicate question detection in Developer Forums and SO has been researched extensively over the years. The explored approaches can be broadly divided into two main categories. The earlier work that focused more on **feature-based approaches**, paved the way and set the baselines for more recent work that utilizes vector spaces and **deep learning techniques** to improve the results and provide state-of-the-art solutions. This chapter provides an overview of the most noteworthy approaches and research findings.

2.1 Feature-Based Approaches

The first interesting machine learning approach was presented by Zhang et al [ZSLA17] that employed a feature-based approach to solve duplicate detection as a supervised binary classification problem. The features that were introduced, one of which is novel for the domain, include vector similarity which is the cosine similarity of a Doc2Vec representation of a post's title and body. Secondly, topical similarity performs topic modeling (Latent Dirichlet Allocation (LDA)) and measures the similarity of topical distributions between two data points. Lastly, a feature that uses the association score of a trained perceptron produces pairs of phrases that are seen frequently in established duplicate question pairs. For the classification, six common classifiers were used, of which random forest was the top performer. Notably, an interesting detail of this approach is the usage of questions from nine different data sets, one of which was the SO dataset, created from the 2016 data dump, and containing questions from six different programming languages.

Shortly after, Zhang et al. developed DupDetector as a two-stage “ranking-classification” problem to detect duplicate questions both in SO and Quora [ZSL⁺18]. The first step involves ranking the historical questions by relevance, for example by using topic modeling, to limit the search space and significantly speed up the second step, which is the supervised classification model. Similar to their previous approach, the classification involves features such as vector similarity, a relevance feature adapted from the first step, and an association feature that uses frequently found phrase pairs and each domain's duplicate questions. They took into consideration four question features namely the similarity scores between topics, titles, descriptions, and tags. The first calculation was the similarity between the topics of the two question pairs by employing topic modeling using the LDA algorithm. For the remaining features the similarity score between pairs of titles, tags, and descriptions respectively, was computed based on the similar words that they contain. Finally, the four similarity scores were consolidated to perform duplicate detection.

Silva et al. [SPM18] set out to reproduce the DupPredictor results of Zhang Y. et al., an approach quite similar to the previously described ones, which used mostly the same features with the goal of calculating their weighted similarity scores and rank them by relatedness [ZLXS15].

However, the reproduced work observed a decay in performance and the reported results were not able to be perfectly replicated. A suggested explanation was the difference in the corpus between replications because a more recent data dump of SO was used as a basis. An association between the increase of the corpus pairs and the decrease in performance was also observed, which establishes the need for robust methodologies that are unaffected by the corpus size.

2.2 Deep Learning Approaches

Advancements in NLU research guided the shift towards more deep learning approaches that are able to extract the semantic relationship between text, beyond the lexical similarity.

Xu et al. were the first to propose a multi-class approach to predicting semantic similarity between SO KUs [XYX⁺16]. To perform the word embedding training and the classification tasks, they utilized data from the 2016 Stack Exchange data dump and they compiled 6,400 pairs of KUs that have the tag "Java". The approach uses domain-specific Word2Vec embeddings following the continuous skip-gram model and trained with 100,000 KUs related to Java. The researchers then trained a Convolutional Neural Network (CNN) model for the prediction task using cosine loss to capture the semantic relations between the posts.

Shah et al. highlight the prevalent issue of the unavailability of labeled data, especially in cases of forums outside of Stack Exchange [SLM⁺18]. They address this issue by exploring Adversarial Domain Adaptation (ADA) to essentially use labeled data from one platform to the other. To use question pairs between quite heterogeneous data sets, they included a domain adaptation component that aims to make the model unable to differentiate whether a question is from the source or the target domain. The experiments concluded that direct transfer is not particularly effective between heterogeneous domains, that adversarial adaptation is most effective when the domains are rather similar, and that the observed effectiveness is positively correlated with the domains' n-gram similarity.

Hindle and Onuczko proposed an interesting solution to aid in duplicate bug report prevention before they occur by continuously querying for duplicate reports [HO19]. Their solution aspires to provide a new kind of duplication detection by the means of prevention. The idea is that as the user is typing the bug report, the system is continuously checking for duplicates in the bug database, and in case a duplicate is found, alert them to the fact. For the information retrieval task, they used Term Frequency–Inverse Document Frequency (TF-IDF) to model the vector space and cosine similarity as the distance function between the vectors. The word limit for the bug report query was set to 25 words to make the evaluation time-efficient, although better query performance was observed at the 100-word threshold. In terms of evaluation metrics, classical information retrieval measures were considered such as top-k, Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), as well as 3 new ones pertaining to the continuous querying task of the paper, namely aveP-TOP5, MRRTOP5, and MRRTOP5⁻¹.

Wang et al. were one of the first to utilize an Recurrent Neural Network (RNN)-based architecture to detect duplicate questions in a SO specific task [WZJ19], [WZJ20, p. 25965]. They used "134,261 non-master and 88,476 master questions" from SO that are dated from 2008 to 2014. The question tags selected, represented six different question groups that included Java, HyperText Markup Language (HTML), Python, C++, Ruby, and Objective-C. Word2Vec word embeddings were created by concatenating the titles, bodies, and tags of the questions. They used three different deep learning approaches Word Vector Convolutional Neural Network (WV-CNN), Word Vector Recurrent Neural Network (WV-RNN), and Word Vector Long Short-Term Memory (WV-LSTM), that utilized CNN, RNN, and Long Short-Term Memory (LSTM) respectively. For the evaluation, recall-rate@k was employed at $k \in \{5, 10, 20\}$ which measures the percentage of questions whose duplicate was found within the top k results. The best-performing model was the LSTM which

achieved a recall rate of around 82% for all k .

Rücklé et al. employed self-supervised training, weakly supervised training methods, as well as question generation models to produce a new question title using the body. Afterward, they used the generated title as a duplicate of the original in an effort to enrich the dataset [RMG19]. Similarly, they trained models directly on the original title and body pairs with the goal of predicting whether they belong to the same question. They report that title-body pairs produced better Area Under Curve (AUC) (0.05) results than question-answer pairs, something that could be explained by the calculation of the TF-IDF score and Jaccard coefficient. The same metrics show that the title overlaps more with the body than the title and body combined, i.e., the whole question does with the answer. This approach, while rather interesting, assumes that the questions are short and well-formulated. However, this is not necessarily the case with Discord conversations, where the formulation of a question can span multiple messages and utterances and more importantly do not include titles.

While most approaches try to solve the problem of Duplicate Question Detection (DQD) by following a supervised learning approach, Poerner and Schütze further explore the unsupervised path via representation learning. According to the authors of [PS19, p. 1631], unsupervised DQD “is related to the task of unsupervised STS” which scores sentences based on similarity, in this case, cosine similarity. Motivated by the work of [RVDA15] and [SLS18] on Canonical-Correlation Analysis (CCA) they take the idea of ensembling different word embeddings and applying it to the sentence level using unlabeled task-specific data for the training.

Xu and Yuan pursued the task of duplicate question detection in Massive Open Online Course (MOOC) forums using a model trained on 5 Stack Exchange domains using the CQADupStack dataset [XY20]. The MOOC dataset of 1000 question pairs was collected and labeled by actual course participants. The produced model was CNN based and aimed to capture the semantic interaction between questions by encoding the question titles using Global Vectors for Word Representation (GloVe) embeddings pretrained on the Common Crawl corpus. The final prediction for the binary classification task in the target domain was achieved by using an ensemble of the proposed model, that was individually trained on each one of the different subdomains from Stack Exchange. It is worth noting that a small decrease in performance was observed in the target domain of the MOOC forum.

Koswatte and Hettiarachchi examined the problem of duplicate question detection in Q&A platforms from the perspective of real-world application feasibility [KH21]. They identified the efficiency drawbacks that result from comparing every query with the complete candidate duplicate question corpus. The main concern that the paper addressed was the fast retrieval of duplicate questions that require a computationally and space-efficient way by reducing the search space. The novel idea of hashing embeddings for efficient word representations comes from Svenstrup et al. [TSHW17] but Koswatte and Hettiarachchi are the first, to the best of our knowledge, to use the idea in the context of SO and with the combination of Semantic Text Similarity. Concerning the specifics of the task, they used a generalized Bidirectional Encoder Representations from Transformers (BERT) model that was fine-tuned on the domain dataset. The queries of the test dataset are after training encoded using the BERT model and hashed. The final rank was obtained using cosine similarity.

2.2.1 Bi-Directional LSTM Models

The two most promising solutions that focused on SO posts were published within the last two years and both report state-of-the-art performance for their respective SO datasets. Interestingly, both approaches employ Bidirectional Long Short-Term Memory (BiLSTM) based architectures with attention layers for capturing semantic matching. The two models in question are ASIM and SRDM. Their similarities end with the formulation of the examined task as a classification prob-

lem and the evaluation metrics that were consistent between the two. More on their differences as well as a comparison of the two models and their use cases are presented in Chapter 3.

2.3 Available Datasets

A crucial part of any problem definition is finding the appropriate data for the task. The following section presents the most widely used datasets in duplicate question research.

2.3.1 WikiQA

The WikiQA dataset [YYM15] is a publicly available dataset¹ of questions and their sentence pairs, that were collected and annotated in 2015 which can be used to research open-domain question answering. In an effort to effectively represent the information-seeking patterns of the average user, query logs from Bing were utilized to produce the source of questions. Each of the resulting questions is then matched to a Wikipedia page that could best contain the answer. To create the corpus the use of crowdsourcing was key to output approximately 3,000 questions and 30,000 sentences.

2.3.2 Quora Question Pairs

Quora Question Pairs is arguably one of the most popular datasets for paraphrase question detection tasks. It was initially released for a Kaggle Competition sponsored by Quora in 2015 and includes over 400,000 question-only, open-domain pairs along with the binary classification of whether a pair is duplicate or not². The Quora Dataset contains general theme questions and in the majority are not software-development related.

Stack Exchange

As discussed in Section 2, there are various examples of research that use some form of SO data to predict duplicate questions. Stack Exchange, which is a network of websites dedicated to question-answering of various fields, has been an invaluable source of duplicate question pairs for research. Stack Exchange's Data Dump³ can be used to extract question pairs from specific subdomains and time ranges.

CQADupStack

CQADupStack⁴ is a Stack Exchange dataset for community question-answering created by Hoovegreen et al. [HVB15]. It includes posts from twelve different StackExchange subdomains, which are annotated as duplicate questions. The dataset is used heavily in research for benchmarking both classification and retrieval tasks, and train, dev, and test splits are already provided in the dataset. The CQADupStack dataset was created from a Stack Exchange data dump that was released on September 26, 2014. While it is a well-constructed and frequently used dataset, its wider domain focus that does not include SO questions, as well as its quite outdated timeframe

¹<http://aka.ms/WikiQA>

²<https://www.kaggle.com/competitions/quora-question-pairs>

³<https://archive.org/details/stackexchange>

⁴<http://nlp.cis.unimelb.edu.au/resources/cqadupstack/>

rendered it unfit as a corpus candidate for the task of near-duplicate question detection on Discord. In addition, the Spring Framework, which is one of the most popular Java Frameworks, was released in 2014 and has gained its popularity in more recent years.

AskUbuntu

AskUbuntu is an example of a Stack Exchange sub-domain that has been the subject of NLP research over the years, potentially because of its active user base. A version of this dataset is available by [RSM⁺17], assumes the usage of the September 2014 data dump and contains 167K questions and 17K labeled as duplicate pairs.

The focus of this work is of course SO, the most popular website of the Stack Exchange network, and previous work usually focuses on similar subsets in order to facilitate effective comparison between newer and past approaches. An observed trend is using the most popular question tags across SO, the data dump from 2016, and considering fields like the title, question, and tags.

Stack Overflow Knowledge Unit

Shirani et al. [SXL⁺19] created a SO dataset of "domain-specific Question-Question relatedness"⁵. In contrast to most other approaches, which mostly consider the title and question of the post, this approach considers a SO discussion as a complete KU. Utilizing questions containing the Java tag, they built a KU graph that helped uncover relationships between the KUs. Each KU is a graph node and each datapoint in the dataset is a KU pair that contains information about both questions' titles, bodies, (accepted) answers, along with any code blocks, tags, and SO Identifiers (ID)s.

Moreover, they categorized the relationships between posts into four distinct classes, namely "duplicate", "direct", "indirect", and "isolated". The quality of the labeling was tested via a user study where the 3 annotators' labeling agreed with the dataset labels 82% of the time.

The **semantic** meaning between the classes was explained intuitively by the researchers as follows:

- **Duplicate** is a pair of KUs that involve the same question discussed in different ways, and both questions can be answered by the same answer.
- **Direct** relationship is when two questions cannot be answered by the same answer, but their topics are similar enough that one KU can help solve the problem in the other. For instance by including examples, explaining common concepts, or solving only a sub-point of a greater complex problem.
- An **indirect** pair denotes the existence of related topics which do not answer the question of the other KU.
- The **isolated** class marks the unrelatedness of two KUs.

From a more technical standpoint and as far as the KU graph is concerned, the classes are created according to the **graph's nodes** and **edges** as follows:

- **Duplicate** - The questions that are marked duplicate in the SO website.
- **Direct** - The KUs that are directly linked in the graph are questions that have been provided by a community member in the form of a hyperlink at the KU itself.

⁵<https://anonymoussaai2019.github.io>

- **Indirect** - The KUs that are derived from the KU graph and have up to a certain degree of separation, i.e a bounded range of [2,5] was implemented.
- **Isolated** - KUs that have no connection in the knowledge graph.

Intuitively, an advantage of this multi-class categorization of question relatedness could be derived from the fact that even though two questions might not be duplicates, the answer of one can somehow help answer the second question. Therefore, the need for strictly duplicate questions is eliminated to a degree because a user does not have to wait for the question to be answered but can refer to already available related questions. For the sake of completeness it is worth noting that the order of relatedness of the classes is *duplicate* > *direct* > *indirect* > *isolated*.

2.3.3 Communication Platform Datasets

Despite the widespread use of platforms such as **Slack** and **Discord**, datasets derived from developer communities hosted on them are scarce. Some research has focused on mining and disentangling conversations and thus their output could provide opportunities for detecting conversation and SO question pairs.

Chatterjee et al. [CDKP20] modified a technique from Elsner and Charniak [EC08] to disentangle software-related Slack chats and the available dataset includes 39,000 conversations from four channels dedicated to Python, Clojure, and Elm. Based on the same work, Subash et al. [SKV⁺22] also created a Discord dataset of disentangled Discord conversations of four programming language communities, namely Python, Go, Clojure, and Racket. The topic modeling approach they implemented revealed basic conversation topics within the communities and opens up new research opportunities. Interestingly, according to Subash et al., the two platforms share chat conversation patterns which could indicate further generalization potential in duplicate conversation detection. While these datasets do not provide any kind of classification or semantic labeling for the conversations, they were definitely considered as a dataset source. Moreover, available datasets are worth mentioning not only because of their scarcity, but to facilitate future work too.

This Chapter focused on researching and presenting the existing work that has focused on the Stack Overflow domain, as well as the available duplicate-question datasets. In turn, the related work and available data will inform the selection process of the approach to be implemented.

Approach

The following chapter presents in more detail the two state-of-the-art solutions that were considered as the model basis for this thesis and explains the reasoning behind the choice of ASIM over SRDM. As prefaced in Section 1.1, the scope of the selection is to find the best-performing duplicate detection model on SO questions, while also considering the transferability potential to the Discord domain. In other words, the approach selection should be mindful of the characteristics that would make each model a better fit for the target use case.

3.1 ASIM: Attention-Based Sentence Pair Interaction Model

ASIM aims to predict the relatedness between questions on SO. The model "adopts the attention mechanism to capture the semantic interaction information between the questions". The model architecture is using GloVe embeddings and BiLSTM with attention layers to encode SO posts. The embeddings of 300 dimensions were pretrained using data from SO, which infers that the embeddings are domain-specific and are shown to improve the overall model performance. Figure 3.1, shows a schematic representation of the model architecture. The dataset used was the Shirani et.al KU dataset discussed in Section 2.3.2, which contains multiple levels of relevance between Java-related SO posts. The input of the model is two KUs that concatenate the fields of title, question body, and answers, and therefore takes into consideration a big part of each conversation for the prediction, but it ignores code blocks. The duplicate detection task is formulated as a multi-class classification problem that aims to predict the relatedness class of two question pairs. The researchers report a precision and F1 score of 0.82, making the model the state-of-the-art solution of the specific dataset. Interestingly, they test the generalization of the model, by applying the prediction task on the AskUbuntu dataset (Sec. 2.3.2) and achieving state-of-the-art performance on this dataset as well (0.96). The authors claim that the generalization power can be at least partly attributed to the domain-specific embeddings.

3.2 SRDM: Semantics and Relevance Duplicate Questions Detecting Model

Liao et al. [LLZY21] proposed a solution that uses Siamese BiLSTM to detect duplicates in SO. Their model, which is called SRDM, uses Siamese BiLSTM that encodes each question's title and

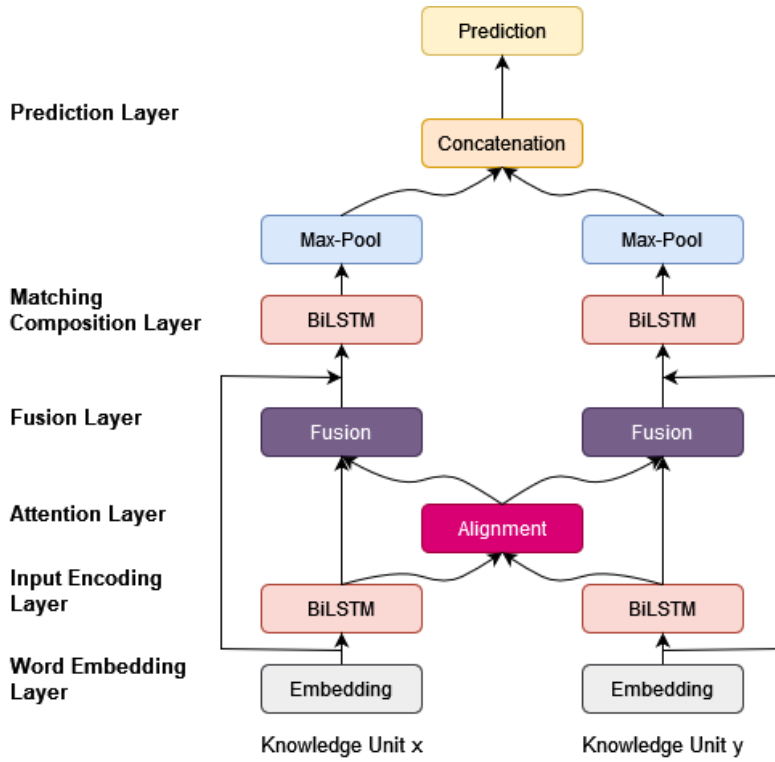


Figure 3.1: The ASIM Architecture as presented by [PWQ⁺21].

body to capture the semantic interaction information with a pairing question's title and body through "soft align attention and inference composition". Apart from the semantic matching, the researchers aim to perform relevance matching as well by soft term matching using only the title. They explain the difference between the two as semantic matching being the one that "emphasizes the '*meaning*' correspondences by exploiting lexical information and compositional structures", while relevance matching focuses on exact lexical matches of the title field. The formulated binary classification problem uses duplicate SO questions from six programming languages, namely Java, Python, HTML, C++, Ruby, and Objective-C. The model used for encoding the input was the generic skip-gram Word2Vec embeddings from gensim, and from each SO post, the question title and body were utilized for prediction. The dataset is the one produced by [SPM18] and includes a total of 197k question pairs which is analogous to the KU dataset of [SXL⁺19] of Section 2.3.2. In terms of performance, the overall reported precision and F1 score were 0.94 for both metrics.

An interesting observation of this paper was derived from the utilization of the pretrained BERT [DCLT18] and SBERT [RG19] models as a baseline for benchmarking performance. The experimental results showed that SRDM performed better than both BERT models, with SBERT having the worst performance among the three, potentially because only the title of the question was utilized. This strengthens the confidence in utilizing BiLSTM with self-attention for the duplicate detection task.

Looking at Figure 3.2 which shows the architecture of SRDM, we can see that the titles and

question bodies are fed separately as pairs to the network. This effectively means that there needs to be a distinction of title and body in the conversation for this architecture to be employed.

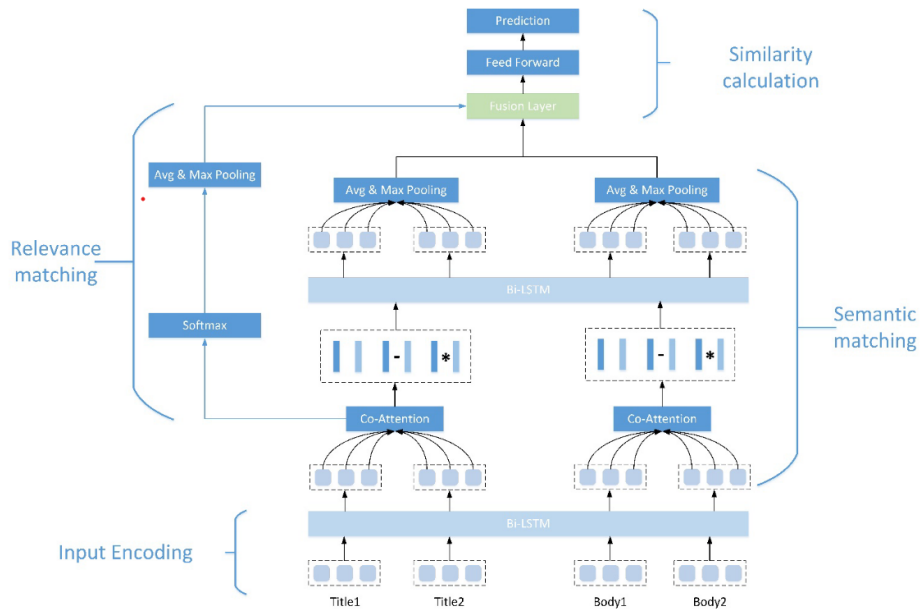


Figure 3.2: The SRDM Architecture as presented by [LLZY21].

3.3 Common Architecture Elements

It is quite clear that the two approaches are similar to each other, and it is safe to claim that the fundamental common elements of the two, contribute a lot towards their performance. Thus, considering the similar elements of the two models, the most fundamental ones are presented below, along with reasoning around their importance.

BiLSTM

A Long Short Term Memory network, or LSTM, is a special version of a Recurrent Neural Network RNN which solves the short-term memory problem of RNNs. As we have seen in previous work in Chapter 2, many approaches utilize this kind of layer to capture the semantic dependencies in text. LSTMs are used to handle long-term dependencies within the text which means that they can learn dependencies even when the lexical distance between the relevant words is large. In practice, they are used for encoding a sequence, and the bidirectional approach, i.e. **BiLSTM** encodes both in the forward and the backward direction. The purpose of this is for the model to understand the words before and after a specific word.

However, before encoding the input, the text needs to be transformed into word embeddings. It is a crucial step that affects the performance of a model significantly, so special care should be

taken when choosing the algorithm that produces the embeddings.

Word Embeddings for Sentence Similarity

The idea of word embeddings, i.e. different processes of representing words by transforming them into number vectors is a fundamental research area of Natural Language Processing. Various approaches have been employed by researchers and professionals over the years, and many of them, like Continuous Bag of Words (CBOW) and Skip-Gram, are based on statistics that consider the occurrence of words in the surrounding context. More recent approaches focus on mapping the words into a vector space, to map words that have similar context, spatially closer to each other.

Since the concept associated with a word is high-dimensional, the strategy of word embedding is to also reduce the dimensions of a word and only keep the dimensions that represent the meaning of the respective word. Word2Vec and GloVe (Global Vectors for Word Representation) fall into the category of neural word embeddings and are very similar in the way they are created. They are both unsupervised learning algorithms that through a word-to-word co-occurrence matrix, which shows how often a word is found in the same context as all the other words, calculate the probability that two words occur in the same context. The main difference between them is that GloVe which was proposed by Pennington et al. [PSM14] produces word representations that also consider the global interaction and context of words within the text. In other words, their novelty results from the consideration of word co-occurrence in the whole corpus rather than only locally per sentence.

As Mikolov et al. pointed out [MYZ13, p. 746], “these representations are surprisingly good at capturing syntactic and semantic regularities in language, and each relationship is characterized by a relation-specific vector offset. This allows vector-oriented reasoning based on the offsets between words” which can be showcased by a widely known example that characterizes GloVe embeddings. The example shows the learned relationship between equivalent male and female words and that the produced vector representations can be used to perform simple calculations on the respective word meanings. Thus, “King – Man + Woman” results in a vector that is quite close to the word “Queen” in the vector space.

Attention Mechanism

Arguably, one of the most important components of the model networks is the attention mechanisms which are the catalysts in the ability of a model to capture the semantic similarity between sentences. Attention mechanism in deep learning was first used by Bahdanau [BCB14] for machine translation but the concept can be applied to numerous semantic tasks. In fact, this concept has been so greatly influential, that Vaswani et al. when they proposed the Transformer architecture contended that, attention is all you need. [VSP⁺17].

In our models, attention is used to capture the semantic correlation between each word of one sequence with the second sequence and vice-versa. In practice, the input sequence is passed to an RNN that functions as an encoder which has a hidden state and contains information from all the previous words in our sequence. The challenge with the non-attention approach is that if the sentence was long, all of the information could not be compressed in the last hidden state and hence the output was inaccurate. The role of the attention mechanism is to assign attention weights to the input. These weights represent the relative importance of each input element versus the rest of the input elements. This way, the model is guided to pay greater attention to particular words that are more critical to performing the task at hand. Therefore with attention, we can give the context vector access to the entire input sequence, instead of just the last hidden state, and still be able to input and process long sequences.

3.4 Chosen Solution

The starkest difference between the two contenders is SRDM's use of relevance matching and the division of the input with the usage of two Siamese Networks. While the methodology of SRDM is novel and produces promising results, it arguably cannot be faithfully replicated and implemented for the Discord use case, as questions in the help channels do not have titles, which the model uses for relevance matching.

Certainly one has to consider the option to adapt the relevance matching to the complete question or conversation in general. However, this is far from ideal if we consider the reported observations of the paper's authors. As an experiment, they explored the case of implementing relevance matching to the body similarly to the title. The experimental results showed that extending the relevance matching to the body decreases performance, perhaps because the more frequently used common words are introducing noise. Their explanation suggested that because the body of the question is longer and it has many description sentences, the text of irrelevant questions will contain lots of vocabulary repetitions, and the usage of relevance matching for the body will introduce errors. Moreover, Discord conversations are not structured in the form of titles and questions, hence the use of multiple Siamese networks that fragment the input, is not optimal.

On the contrary, ASIM considers SO questions as complete KUs, meaning that they utilize not only the title but the body of the question as well as the accepted answer. The same is reflected in its architecture, as each SO pair is used independently as input to the encoder, which poses no restriction to the field that is needed at encoding time. Its approach of concatenating the input, grants more experimentation freedom in terms of the conversation sequence lengths that can be used. In addition, research has shown the positive impact of Domain-Specific Word Embeddings ([XYX⁺16], [PWQ⁺21]) on performance metrics versus the use of multi-source embeddings which is expected to be an aid in Discord's domain transfer scenario.

In regards to the reported performance of the two models, a first look suggests that SRDM performs better than ASIM, as the reported F1-score is 0.94 and 0.82 respectively. However, the performance of ASIM in duplicate and isolated classes of ASIM as well as the experimental results when applied to the AskUbuntu domain, suggest otherwise. Indeed, the replication and experimental results of Section 4.6 show that the two models perform comparatively equally in the binary classification task, with ASIM achieving 2% better precision; an edge that could arguably be attributed to the domain-specific embeddings.

Certainly, the search for an optimal solution has to mention transformers and the models that utilize them, which have undeniably been the catalysts in the advancement of NLP research during recent years and are being successfully used for various tasks from classification and summarization, to language translation and generation. Their parallelizability has enabled the research community to efficiently train models on great amounts of data and achieve state-of-the-art results on most of the aforementioned tasks. However, Ezen-Can [EC20] contends that the accuracy gains of these models diminish when used with small datasets. The results of their experiments comparing LSTM and BERT for "intent classification and out-of-scope prediction" seem to confirm the hypothesis for a corpus of approximately 27,000 utterances [LMP⁺19, p. 5]. The same indication arises from the related work review of Chapter 2 [LLZY21]. Simultaneously, the reality is that more often than not, the available task-specific datasets for many use cases tend to be on the smaller side in both academic research and industry alike.

Therefore given the above, and since the scope and scale of this thesis involve a small subset of Discord conversations, the BiLSTM solution of ASIM was chosen as the optimal approach as the basis of the work. Any limitations and threats to validity that may result from using this approach will be discussed in Chapter 6.

3.5 Evaluation Metrics

Similarly to the selection of the appropriate methodologies to be used, the selection of metrics that will help evaluate their performance is of equal importance. The following Section presents the evaluation metrics available, that are commonly used across existing work and best practice standards. The two main tasks that need to be evaluated as noted in Section 1.4 are classification and information retrieval.

Classification Metrics

The most common evaluation metrics that are taken into account for evaluating classification tasks framed under the assumption of the question-relatedness task are accuracy, precision, recall, and F1 score.

Accuracy is defined as the proportion of true results among the total number of cases examined and is calculated by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

Precision is the proportion of the predicted classes (positive cases) that are actually their predicted classes (positive).

$$Precision = \frac{TP}{TP + FP} \quad (3.2)$$

Recall represents the proportion of classes that were correctly classified.

$$Recall = \frac{TP}{TP + FN} \quad (3.3)$$

F1 Score calculates the harmonic mean between precision and recall and is a good metric when the goal is to minimize both the false positives and negatives [GBV20].

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \quad (3.4)$$

AUC is another evaluation metric suitable for classification, is designed around the binary task, and is used to evaluate how well the model can distinguish between classes. To use AUC for multi-class classification it has to be calculated using one of the two available strategies, namely “one versus one” (OVO) or “one versus rest” (OVR). More specifically, one versus one computes the AUC of all possible pairwise combinations of classes and then takes the average to output a single score, whereas one versus rest computes the AUC of each class against the rest and then takes the average.

An approach that definitely adds variability to the classification task, is the introduction of new data in a near duplicate detection task. One needs to consider the effects that it can have on

the dataset as it can lead to the introduction of numerous false negatives and in turn affect the results. In other words, within the new dataset, there might exist new question pairs which were not identified as duplicates and thus were not annotated. For this reason, [SLM⁺18] and [RMG19] used the AUC to evaluate the model's rank between positive and negative pairs. Specifically, **AUC(0.5)** is the AUC with a set threshold for false positives and it can be considered as more stable when many unrelated questions are present and we cannot control for false negatives. The caveat in its usage is that there does not exist a robust implementation of a generalization for the multi-class classification assumption [Dö18].

Neural Retrieval Metrics

Neural Retrieval metrics do not differ from the standard Information Retrieval metrics. Below are a few fundamental metrics that can be used for the evaluation of any retrieval approach.

Top-k or **hit-rate** measures whether a query returned a successful duplicate within the top k results. If the algorithm succeeds then the particular query gets a score of 1. The reported measure is the average across all the queries. It is a straightforward metric that is usually a simplistic way to measure performance.

Recall-rate@k as explained in [WZJ19] is the number of questions whose master question pairs appear in the list of *topk* retrieved results, while N_{all} is the total number of duplicate questions used for testing. It essentially measures the percentage of queries whose master pairs are successfully retrieved in the list.

$$Recall - rate@k = \frac{Nk}{N_{all}} \quad (3.5)$$

Obviously, the above definition of *recall - rate* assumes that the pairs to be retrieved are unique and there is a master result for each query. Realistically, a retrieval task can have multiple relevant results. A generalization of the metric considers all the results that are relevant to the query, calculates the percentage of the relevant answers that were retrieved at the *topk* for each query, and returns the average across multiple queries. In case the concern is how many of the *topk* retrieved results are relevant, then **Precision@k** should be used instead.

MRR which stands for Mean Reciprocal Rank, gives priority to the rank of the expected result and evaluates the full results and not just the top k . It calculates the average of the reciprocal rank of the correct answer in a query and the final measure is again the average amongst n queries. This subsequently means that correct results with low ranking are penalized and it simultaneously supports only a single result as correct.

Mean Average Precision or **MAP** is another information retrieval measure that does take into consideration multiple matches and achieves it by ranking the effectiveness of a number of queries but penalizes results that are correct, yet lower ranked. MAP does not have a cap on the number of results that evaluates and is calculated by measuring the average precision (aveP) of a single query. As usual, it then produces the mean of the aveP for all n queries. However, it does not take into consideration any false negative results which can be prevalent in the duplicate question clusters. For instance, consider the case of a commonly asked question in Java. By definition, one can anticipate the existence of many duplicate instances. The same can already be observed in the KU dataset and the graph of Section 4.1, where some nodes have upwards of 60 connections.

Model Replication

The following chapter will focus on the first research question of Section 1.2 by replicating ASIM and exploring its classification capabilities with respect to the SO dataset and different variations in text input. The first matter to be addressed is the dataset. As already mentioned, the KU Dataset of Shirani et al. [SXL⁺19] is utilized, which was introduced in Section 2.3.2. Aiming to understand its underlying structure and characteristics, we begin by exploring the dataset.

Scope	Indicator	Size
Whole KU	# of distinct KUs	160,161
	# of four types of KU pairs	347,372
Title	avg. # of words in title	8.52
	avg. # of words in body(exclude code snippets)	97.02
	# of distinct KUs whose body has at least one code snippet	117,139(73%)
Body	avg. # of code snippets in one body	1.46
	avg. # of words in single code snippet in one body	118.46
Answers	# of distinct answers	318,491
	avg. # of answers within single KU	1.99
	# of distinct KUs contain at least one answer	140,122(87%)
	# of distinct KUs contain an accepted answer	90,672(57%)
	# of distinct KUs whose answers has at least one code snippet	96,707(60%)
	avg. # of words in an answer (exclude code snippets)	68.39
	avg. # of code snippets within one answer	0.60
	avg. # of words in single code snippet	81.98

Table 4.1: Knowledge Unit Dataset Statistics (Shirani et al.).

4.1 Dataset Exploration

The dataset creators provided some basic statistics about the corpus, as seen in Table 4.1. The number of distinct questions is 160K, while the derived pairs in the four balanced classes are almost 350K. A note on terminology; for the purpose of describing an individual question within the SO corpus, the terms question, SO post, and KU will be used interchangeably. In addition, since the corpus is divided to train, dev, and test dataset, it is worth mentioning that a single line within the corpus is a pair of two KUs, along with their attributed class.

Given that the corpus is the result of a KU Graph, it is interesting to know how the questions are connected with each other and how many related questions each one roughly has. Figure 4.1 shows the frequency of each question in the knowledge graph, i.e the distribution of the node connections. The x-axis represents the number of times a question occurs in the knowledge graph, and the y-axis, which is logarithmically scaled in the figure, represents how many questions with each occurrence count exist in the graph. We can see that most of the KUs have relatively few connections, while a few of them have up to 60 connections. In addition, the data connection visualization supports the observations made by Park et al. [PLY05] who stated that self-organized scale-free network structures, i.e the KU network graph, follow a power law degree distribution. This means that just a handful of posts get the majority of users' attention, while the rest of the posts amount to the long tail of the distribution and are therefore under-linked. Interestingly, the same can also be explained by preferential attachment theory also known as *"the rich get richer"*.

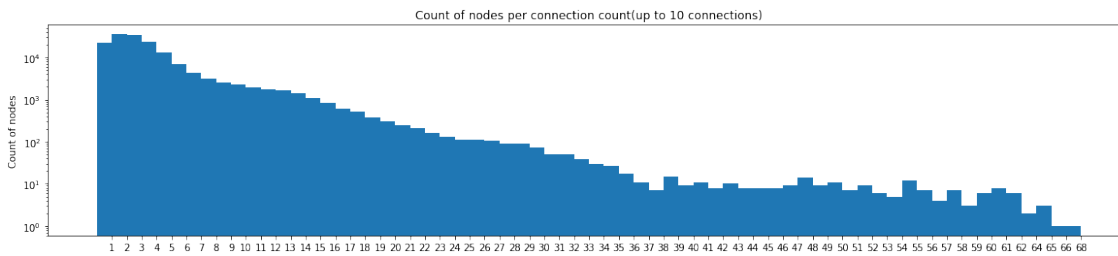


Figure 4.1: Count of nodes per connection degree - Distribution of question occurrence in the dataset - x: Frequency of question - y: Question appearance counts

Thus it seems that popular answers get more frequently shared than others or that there are some questions, some form of which, are commonly asked by users. It is worth noting though, that the collection of pairs across the KU dataset is unique, meaning that for every $[KU1, KU2]$ pair in the dataset, there is no reversed $[KU2, KU1]$ pair. The assertion was performed to ensure that an already learned pair by the model, does not exist in the evaluation dataset, something that would give a false signal of the model's rote learning rather than its true performance.

Knowledge Unit Text

Each KU text sequence consists of fields found in the Stack Exchange data representation and includes the title, the question body, as well as all the answers. The KUs do not take into consideration other data fields such as the code blocks and question tags. A KU is constructed by simply concatenating all the fields as strings into a text sequence. Looking again at Table 4.1 and considering that the input sequence of the model is limited to 250 words, we can infer that the model input before data preprocessing contains on average, the title, body, and two answers, and they contribute to the KU size by 3.5%, 38.8% and 57.7% respectively.

How do I loop back to the start of the TRY statement? [duplicate]

Asked 8 years, 10 months ago Modified 8 years, 10 months ago Viewed 6k times

1 This question already has answers here: [How do you implement a re-try-catch?](#) (30 answers)

Closed 8 years ago.

How do I make it loop back to the start of the FOR loop again? I had try searching the answers. Use of continue don't seems to work for me.

```
try{
    for (int i = 0; i < Array1.length; i++){
        System.out.println("enter Values ");
        Scanner input = new Scanner(System.in);
        Array1[i]= input.nextInt();
    }
} //try
catch (InputMismatchException e)
{
    System.out.println("pls enter integers only ");
} //catch
```

How do I go about continuing the process again? For example

Enter Values 5 enter Values 1 enter Values g pls enter integers only -> error exception is over here

After this error show, how am I able to continue the enter values process without retyping from value 1?

How do you implement a re-try-catch?

Asked 9 years, 10 months ago Modified 2 months ago Viewed 270k times

253 Try-catch is meant to help in the exception handling. This means somehow that it will help our system to be more robust: try to recover from an unexpected event.

We suspect something might happen when executing an instruction (sending a message), so it gets enclosed in the try, if that something nearly unexpected happens, we can do something: we write the catch. I don't think we called to just log the exception. I think the catch block is meant to give us the opportunity of recovering from the error.

Now, let's say we recover from the error because we could fix what was wrong, it could be super nice to do a re-try:

```
try{ some_instruction(); }
catch (NearlyUnexpectedException e){
    fix_the_problem();
    retry();
}
```

This would quickly fall in the eternal loop, but let's say that the fix_the_problem returns true, then we retry. Given that there is no such thing in Java, how would YOU solve this problem? What would be your best design code for solving this?

This is like a philosophical question, given that I already know what I'm asking for is not directly supported by Java.

java exception exception-handling try-catch

Share Follow

edited Oct 18, 2013 at 6:57 asked Nov 5, 2012 at 20:56

Peter O. 31.2k • 14 • 77 • 92 Andres Farias 2,633 • 2 • 12 • 9

(a) Example of duplicate questions in SO

Apache POI: Show HTML text in Excel

Asked 5 years, 5 months ago Modified 5 years, 5 months ago Viewed 1k times

1 Need a help. Honestly I don't have a clue now how to start on this.

<p>This is a test. Will this text be bold or <i>italic</i>?</p>

So in my excel, it should be

This is a test. Will this text be **bold** or *italic*

Again, I do have example which write this as plain text. Could anyone please help me in making this text as Rich Text in excel. Note: this is just an example. In realtime, it could be any html with styles.

java apache-poi

Share Improve this question Follow

asked Mar 9, 2017 at 19:37

5k 1,346 • 2 • 24 • 52

related: [stackoverflow.com/questions/9697330/](#) - user180100 Mar 9, 2017 at 20:13

2 You probably should read [poi.apache.org/spreadsheet/](#) - user180100 Mar 9, 2017 at 20:19

2 "In realtime, it could be any html with styles." Then you will need a HTML parser - a tag soup parser like internet browsers have - before you can create [Rich Text Strings](#). That's not trivial and so cannot be totally answered in a question here. Choose a HTML parser for Java and try something. Then come back with a concrete question if you are stuck. - Axel Richter Mar 10, 2017 at 4:48

Add a comment

Setting a part of the cell contents to bold using apache poi?

Asked 10 years, 5 months ago Modified 1 month ago Viewed 18k times

12 How can I put this string in to a cell using java apache poi?

The string is "Hello **world** Hello"

As u can see I need to make a part of the text bold?

I'm able to set the entire contents of the cell to bold but not specific parts.

Please help me.

java html apache-poi

Share Improve this question Follow

asked Mar 14, 2012 at 7:05

Saicharan S M 820 • 3 • 11 • 24

Add a comment

3 Answers

Sorted by: Trending sort available Highest score (default)

18 This is probably what you are looking for: <http://poi.apache.org/spreadsheet/quick-guide.html#DrawingShapes>

Find this in the explanation:

It's possible to use different fonts to style parts of the text in the textbox. Here's how:

```
HSSFFont font = wb.createFont();
font.setItalic(true);
```

(b) Example of a direct question-pair in SO

Safe class imports from JAR Files

Asked 12 years, 9 months ago Modified 12 years, 9 months ago Viewed 3k times

1 Consider a scenario that a java program imports the classes from jar files. If the same class resides in two or more jar files there could be a problem.

- In such scenarios what is the class that imported by the program? Is it the class with the older timestamp?
- What are the practices we can follow to avoid such complications.

Edit: This is an example. I have 2 jar files my1.jar and my2.jar. Both the files contain com.mycompany.CrazyWriter

java class import jar class-library

Share Follow

edited Jan 21, 2010 at 10:20 asked Jan 21, 2010 at 8:50

Paul Wagland 26.5k • 9 • 50 • 74 Chathuranga Chandrasekara 20.2k • 29 • 96 • 137

1 same class resides in two more classes -can you clarify? - Aadith Ramia Jan 21, 2010 at 8:55

1 "If the same class resides in two more classes"? Did you mean "If the same class resides in two more jars"? - phisch Jan 21, 2010 at 8:57

Java classloader with same jar

Asked 8 years, 1 month ago Modified 8 years, 1 month ago Viewed 747 times

1 I know that many subject speak about classpath loader and transitive dependency but i m little confused.

I m working on a legacy application that use spring 2.0.5 and spring-ws and all work fine from the beginning of the project to now. And recently we are faced to a runtime problem with an exception like methodnotfoundexception.

In fact we see that spring-ws depend on spring-2.5. So maven transitive dependency add spring 2.5 in my webinf/lib directory near spring 2.0.5

But i dont understand why all working fine during many years and now why weblogic decide to load spring2.5 before spring2.0.5 and cause this error?

We have correct the problem and now i m looking for same jar with different version and do build failure when i have a conflit to avoid dependency problem in the future.

But i just want to understand why weblogic decide to load one or another jar ? Because if its alphabetical order, same jar will be loaded all the time. but in my exemple the order change... So i don t understand clearly whats happened.

Thanks by advance

java weblogic classpath

(c) Example of indirect questions in SO

Figure 4.2: Example of KUs in the Stack Overflow corpus.

4.2 RQ1: Model Reproducibility

RQ1: Can the results of current state-of-the-art approaches in SO duplicate detection be successfully reproduced?

4.3 Preprocessing

In order to effectively replicate ASIM and to use the pre-trained GloVe embeddings¹, the SO questions need to be preprocessed and thus detailed instructions should ideally be present for each step of the process. A missing element in the ASIM authors' reporting, could arguably be the lack of details about the steps followed for preparing the corpus. The available instructions provided in their paper stated that they "used the posts part to extract the content in the body tag, and clean the content by removing the HTML tag, code snippets, Uniform Resource Locator (URL) link, punctuation, stop words, and changing all words to lowercase and stemming" [PWQ⁺21, p.4].

Therefore, the exact preprocessing steps had to be approached. This is to ensure that the preprocessed text matches as close as possible the word embeddings because unknown words add unwelcome noise to the vectors and do not contribute any information. An effective way to achieve the closest approximation is by measuring the vocabulary coverage and the Out of Vocabulary (OOV) words of the preprocessed corpus. OOV words result in a loss of information that the model cannot utilize, so the closer these two metrics get to the reported or the best possible accuracy, the more confident one can be about the correct replication and validity of the experiments. We report that the best approximation yielded vocabulary coverage of 90.87% and OOV instances 0.03%, i.e 30, 464 out of the 100, 779, 588 total words.

The following steps summarize the approach that achieved the above results:

1. **KU creation:** The first step is to create the KUs by concatenating each question's Title, Body, Accepted Answer, and rest of the Answers.
2. Hyperlinks are located using a regular expression and the assumption is made that they are **normalized** by replacing them with the *url* token since it was present in the GloVe vocabulary.
3. A step that was not mentioned in the initial approach but was found to increase the vocabulary coverage was the splitting of **CamelCase words** to individual words.
4. **Punctuation Marks** are all removed, as well as **unicode values** and **numbers**
5. All words are transformed to **lowercase**
6. **Stopword removal** was performed by using *nltk.corpus*
7. **Stemming** of the words was achieved by using *nltk's PorterStemmer*, while Lancaster stemmer, Snowball, as well as *WordNetLemmatizer* were additionally considered and tested.

4.3.1 Misspelled Words

It is admittedly quite common for misspelled words to appear in the text produced by humans which is something that translates to misspelled words in the available questions and answers.

¹<https://zenodo.org/record/4641569>

So, in the attempt to mitigate these and potentially match the domain-specific embeddings more closely, the *Textblob* library was considered as a way to find and fix typos. However, this had the opposite effect that showed a considerable decrease in vocabulary coverage, something that can perhaps be attributed to the fact that the vocabulary used by the library is not well suited for the particular domain. Moreover, the vocabulary of the pre-trained embeddings cannot be used because it contains stemmed words that would not perform well with the misspelled words that cannot be stemmed by nature. Other explanations were provided by Homma et. al who observed an analogous decrease in performance when spellchecking was applied during preprocessing [HSY16]. They pointed out that if the GloVe embeddings were trained on unsanitized datasets, the resulting vocabulary can already include common misspellings. In addition, some words that would not be recognized by the GloVe vocabulary could potentially be changed to real, but unrelated, words, something that would introduce noise to the text.

Creation of KU pairs for model input

The preprocessed dataset has to be shaped into appropriate text input and saved to *.txt* files (training, dev, test) in order to be used as model input for the classification task. Each line of the file represents a KU pair and is appended to the file in the form of `< KU1 KU2 class >`, each element is separated by the tab character. The task we aim to replicate is predicting the relatedness class between such KU pairs. As a reminder, the relationship between classes was explained in Section 2.3.2, and for reference, some question pairs for each class are shown in Figure 4.2.

4.4 Model Architecture

To showcase how the KU pairs are utilized and how the class prediction is achieved, a more detailed view of the model structure can be examined. We can refer once again to Figure 3.1 which shows the schematic overview of the model architecture as well as the following brief summary of its components:

1. The **Word Embedding Layer** is where the input of the model, i.e KUx and KUy are transformed to their vector representations using the pre-trained domain-specific GloVe word embeddings that were discussed in Section 3.3 and are truncated to a max length of 250.
2. The vector representations are passed to the **Input Encoding Layer** through two BiLSTM modules. The purpose of BiLSTM is to capture semantic dependencies within each KU both in forward and backward directions and determine the important information within each individual KU.
3. The **Attention Layer** follows, which we have already established as an integral part of capturing semantic relatedness between KUs. The aim of the attention layer is to calculate the interaction between KUs. The same is achieved by first calculating the dot product of the two sequences which produces the Inter-Attention matrix between each word in KUx with each word in KUy . Subsequently, the Inter-Attention matrix between each word of KUx in relation to the complete KUy is also calculated and vice-versa for KUy . Thus, the interaction information is captured for each vector.
4. The purpose of the **Fusion Layer** is to combine for each KU the contextual and Inter-Attention representations of the two previous steps, which is achieved with three different fusion methods. Suppose X is the contextual information for KUx from the Input Encoding Layer and \hat{X} is the output from the Attention Layer for KUx . The three fusion methods $F_{1,2,3}$,

concatenate the vectors, calculate their difference and calculate their dot product as shown below:

$$\widetilde{X1} = F1([X; \hat{X}]) \quad (4.1)$$

$$\widetilde{X2} = F2([X; X - \hat{X}]) \quad (4.2)$$

$$\widetilde{X3} = F3([X; X \circ \hat{X}]) \quad (4.3)$$

The outputs of the three fusion methods are concatenated $[\widetilde{X1}; \widetilde{X2}; \widetilde{X3}]$ and passed to a feed-forward network which produces the output of the Fusion Layer \widetilde{X} for KUx and \widetilde{Y} for KUy .

5. The **Matching Composition Layer** is another BiLSTM module that encodes the aggregated contextual information from the previous layer and produces the final encoded vectors Vx and Vy for each KU.
6. Finally, the **Prediction Layer** needs to convert the encoded vectors to fixed-sized vectors through max-pooling, and concatenate the two vectors by using different matching heuristics (i.e., simple concatenation, element-wise product, and the difference between the vectors according to [MML⁺15] and Equation 4.4) to create the feature input for a multi-layer feed-forward network. The Feedforward Neural Network (FNN) L will calculate the probability distribution for each class by using a softmax function on the output.

$$pred = L[Vx; Vy; Vx - Vy; Vx \circ Vy] \quad (4.4)$$

4.5 Model Training

The implementation of the model uses the Pytorch² Framework and is based on the code-base of RE2 [YZG⁺19], as well as the adaptation of [PWQ⁺21]. For the model replication task, the code base was updated to reflect newer versions of library requirements, and the configuration file was adapted slightly in terms of hyperparameters and settings. All the changes and settings are captured in the project's code repository. The training was performed with an NVIDIA GeForce RTX 2080 Ti Graphics Processing Unit (GPU).

4.6 Replication Results

After the recreation of the KU pairs, the replication of the model pipeline and the trained model is saved, the test dataset can be used for evaluation against the reported results of ASIM. By simply comparing the replication metrics with the paper's reported metrics, we can evaluate performance by training a model according to the reported specifications.

The reported results are calculated based on the test dataset of around 104K question pairs, while the pairs across each class are kept balanced. The model reaches an average Precision of 80% across all classes, with classification recall and F1-score also maintaining the same score. Table 4.2 shows a summary of the replication results of the original ASIM model. It can be clearly observed that the model performs well on duplicate and isolated questions and it has some trouble with direct and indirect classes.

²<https://pytorch.org/>

Class Name	Precision	Recall	F1-score	class size
"isolated"	0.94	0.93	0.93	26053
"indirect"	0.65	0.61	0.63	26053
"direct"	0.71	0.68	0.69	26053
"duplicate"	0.87	0.97	0.92	26053

accuracy			0.80	104212
macro avg	0.79	0.80	0.79	104212
weighted avg	0.79	0.80	0.79	104212

Table 4.2: ASIM Replication results.

Compared to the reported results from the authors of the paper, the performance degrades slightly as they reported Precision, F1-score, and Recall at 0.82. An explanation for the drop could be small differences in the data preprocessing steps that were followed during the replication. A closer look at the confusion matrix of the evaluation step in Figure 4.3a, reveals that the most difficult class to identify is the direct one which is misclassified one-third of the time as mostly indirect and duplicate for the rest of the misclassified cases. Something similar is observed with the indirect class. The test pairs are often predicted as being direct instead.

The results can intuitively be explained by the definition of the classes. Direct and indirect classes are not technically duplicates, but rather aim to capture different levels of relatedness between KUs. This is an inherently more challenging task than the binary case of duplicate and non-duplicate, therefore the difficulty in differentiating between them is to be expected.

The replication results lead to the need of understanding more about the relationship between the classes and how the results change once we consider different relationships between pairs. To better understand how well the model can differentiate between classes and better understand their relationship, more experiments were performed by combining and dropping classes. This can also help understand performance gains by using fewer classes and highlight whether the multiclass classification approach is adding value to the overall prediction task.

A performance summary of the different models in terms of the F1 score is shown in Table 4.3 and Figure 4.3 shows the confusion matrices for each experiment. Below follows an overview of the most notable ones ³, along with the resulting observations.

Experiment 1

Since the replication showed that the most difficult distinction is between the direct and indirect classes, we can combine them and measure the performance gains. To keep the classes balanced, half of the pairs from the new class were dropped to account for that. Indeed, the evaluation results show a 12% increase in precision, recall, and F1 score which is introduced by merging the two classes. The confusion matrix, in turn, reflects the change but as expected, the direct/indirect pairs that are misclassified as isolated or duplicate pairs stay relatively constant.

Experiment 3: ASIM-RV3

The next experiment was formulated with the end user of a recommendation or a retrieval system in mind. Taking into consideration the definition of the available classes, a user would probably be interested in finding primarily any duplicate posts that answer their question directly, and secondly the direct case where they would get a KU that is directly relevant to their question.

³Experiment numbering skips numbers to represent the model versions in the project archive.

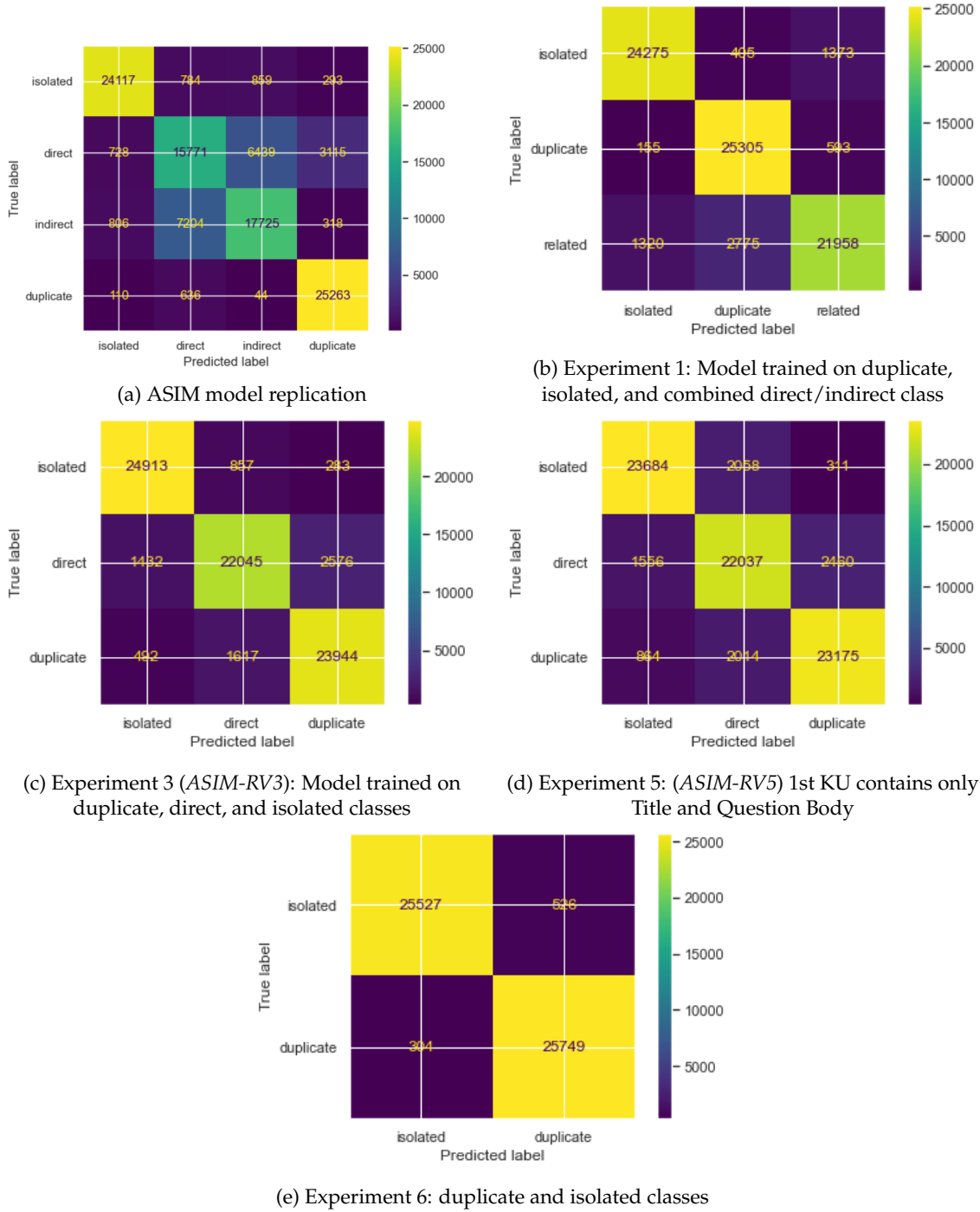


Figure 4.3: Confusion Matrices of Model Experiments.

Model	isolated	direct	indirect	duplicate	related	Overall
ASIM	0.94	0.68	0.73	0.93	-	0.82
ASIM replication	0.94	0.65	0.71	0.87	-	0.79
ASIM-RV1	0.94	-	-	0.93	0.88	0.91
ASIM-RV3	0.94	0.87	-	0.91	-	0.91
ASIM-RV5	0.91	0.84	-	0.89	-	0.88
ASIM-RV6	0.98	-	-	0.98	-	0.98

Table 4.3: F1 score of ASIM and Model Variations.

With that in mind, the case where the indirect class is dropped is examined to see the expected accuracy given the available dataset and the ASIM architecture. In terms of evaluation metrics as overall and per class performance, the results are rather consistent with Experiment 1. The confusion matrix also shows that although the direct classes that are misclassified as duplicates stay the same, there is a noticeable increase in duplicate pairs that are misclassified as direct from around 600 to more than 1600.

Experiment 5: ASIM-RV5

By now it is perfectly clear that the distinction in the level of similarity between pairs is the main challenge of the approach. But when it comes to the user experience of a potential end user, there is one more point to be made. In the case of a retrieval or recommendation system, the input to the model cannot be the full KU or, in the case of Discord, a full conversation because that would mean that the user is provided with a similar solution after their problem is being addressed. For that reason, it is necessary to test the case where the first KU is limited in length. If we consider a newly posted question, that consequently means that a KU consists of just the title and the question body. Thus, for this experiment, the first KU is limited, while the second one is kept at a full length to approximate a real-time prediction. The results show a decrease in performance to 0.88 from 0.91 in the previous reduced-class experiments. Taking a closer look at the confusion matrix in Figure 4.3d, one can see that the direct pairs stay unchanged but there is a significant increase in isolated and duplicate KU pairs that are classified as direct when the additional information that the answers provided are removed from the input.

Experiment 6

Lastly, since most of the available approaches for duplicate detection in SO follow a binary classification approach, the same could be replicated with the chosen architecture for the sake of comparison. For the training and evaluation, only the duplicate and isolated pairs were considered, while the rest were dropped. This of course means that the amount of data was effectively halved, with roughly 100k pairs for training and 50k for testing. Unsurprisingly, the precision and F1 score increased to 0.98 and the confusion matrix in Figure 4.3e shows a clear distinction between the two classes.

Discussion

The classification task's replication results as well as the experiments that considered different combinations of classes for the model creation show that the selected solution indeed performs adequately in comparison to the existing state-of-the-art research, and outperforms them especially when we are considering only the binary classification and a targeted domain.

However, when it comes to usability in a real-world application, considering more than one level of relatedness between conversations can arguably be beneficial to the end user. The cases where a user's question has an equivalent duplicate question are more scarce than the cases where a similar question exists which provides relevant information to help with the original query. For this reason, the final approach should include duplicate, direct and isolated question pairs, and disregard the indirect ones, because they add more noise than a potential benefit. Ultimately, the user requires the most relevant results, and indirect pairs do not provide the level of relatedness to justify the introduction of potential noise. Similarly, for the sake of real-time applicability, one needs to consider the case where prediction is achieved using only the question of a new query versus the already existing complete conversation. Therefore the remaining tasks and approaches will consider the *ASIM-RV3* model of Experiment 3.

Overall, in terms of RQ1, we conclude that the replication of the chosen model was successful and produced quite similar results to the original (Tables 4.2 and 4.3). The small variability in the results could be attributed to small differences in data preprocessing. We were additionally able to eliminate some noise caused by the indirect class through a small ablation study. The identified class will be eliminated for the purpose of this work in order to gain better class distinction and overall performance.

Detecting Discord and Stack Overflow Pairs

The previous chapter answered the first research question, showed that the replication of ASIM is successful, and while considering different input lengths to cater to the retrieval task, the performance degrades slightly but still achieves acceptable results during evaluation. This chapter will explore and address the rest of the research questions and investigate the produced model's transferability as well as the feasibility of a retrieval solution. We begin by considering the available datasets.

5.1 Discord Dataset Construction

Datasets extracted from developer communities and platforms, for instance, Slack or Stackoverflow is scarce. Some interesting work is been done on disentangling Programming Community Question-Answer (PCQA) platform and Slack conversations [EC08, CDP⁺19, CDKP20], with the latter focusing on Python, Clojure, and elm communities as mentioned in Section 2.3.

For the purpose of this work, the optimal dataset should include conversations of Java-related conversations labeled similarly to the selected baseline model, i.e. duplicate, related and isolated. A fitting dataset for the task is not available as far as the research for this thesis is concerned, so a part of the work focused on extracting Discord conversations from one of the largest Discord Java communities, the Java Community | Help. Code. Learn¹. The chosen community is quite large with more than 17,000 users. The mined conversations span a period of almost 9 months from October 2021 to June 2022. Discord servers do not have a strict limit for storage or a certain retention period for messages, so the extracted conversations include the full history of the community up to the point of extraction. The goal is to match Discord conversations with SO posts and create a dataset that can be used for evaluation and further training.

5.1.1 Conversation Extraction

For the task of extracting Discord conversations, the help channels which are dedicated channels for help-related queries can be utilized. An example of a Java community conversation in such a channel can be observed in Appendix B. Apart from the actual message content, it is worth noting the organizational structure of the Java community, especially concerning the help channels on

¹<https://javadiscord.net/>

which we focus. Despite the seemingly disorganized nature of Discord community discussions, the structure of a popular community like the Java Community we are examining is, upon inspection, quite defined. The help section, in particular, is divided into multiple channels. When a person needs help they can use an available channel that is not currently in use by another member and ask their question. A bot helps coordinate and moderate the channels by reserving them, marking them as dormant, and making them available again. This means that each channel contains multiple different conversations and subtopics which can in most cases be separated into distinct conversation threads with the help of a parser. The extraction of separate conversations, as far as this thesis is concerned, was performed using code provided by the HASEL lab, which takes into account bot messages that moderate the usage of help channels to perform the extraction. This is achieved by using the convention that when a user posts a new question to a channel, the bot responds by reserving the channel until it gets unreserved manually or automatically after a certain period of inactivity. Using this convention, the conversations can be segmented and treated as a single unit.

Furthermore, each conversation consists of multiple messages. It is rather common in chat discourse that a user is sending more than one consecutive message, which in a way divides the utterance into sentences. With the term utterance, research in the NLU field generally defines a single unit that can be used for analysis. For the purpose of analyzing the Discord dataset, an utterance will be defined as the set of messages that a user sends before another user responds. Considering multiple messages as an utterance will facilitate more effective tracking of message exchanges among community members and will ensure that any given sentence or question is confined within the same utterance. For instance, in the Discord conversation of Figure B.3, the first utterance consists of the three message lines before and after the code block.

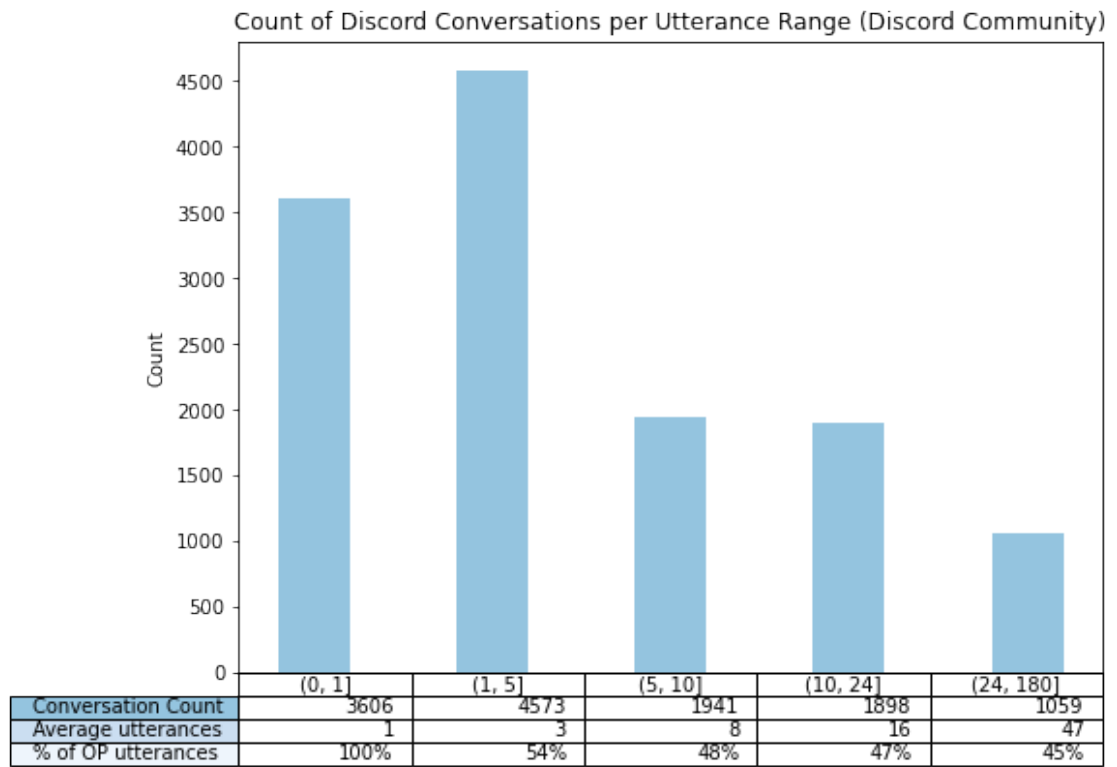
The mined dataset reveals the existence of 17 help channels in total, that collectively contain a sum of 13602 identified conversations. Figure 5.1 gives an insight into the distribution of the community's help-related conversations in terms of utterance count. We can see that over 35% of the conversations in the help channels have up to 5 utterances. A closer look reveals that 27.5% of all the conversations are messages without responses. This means that over 60% of the conversations remain quite brief. The rest of the help requests are distributed at 14.8% for cases with (5, 10] utterances, and another 14.5% with (10, 24] utterances. Finally, the last 8% have over 24 utterances with a mean of 47 in this particular range. Figure 5.1a shows the above schematically, along with the conversation counts and average utterances for each range. An interesting observation is that the Original Poster (OP), i.e the person who asks for help, accounts for half of the utterances in all community conversations, regardless of the discussion length.

5.1.2 Mining of Discord-SO Pairs

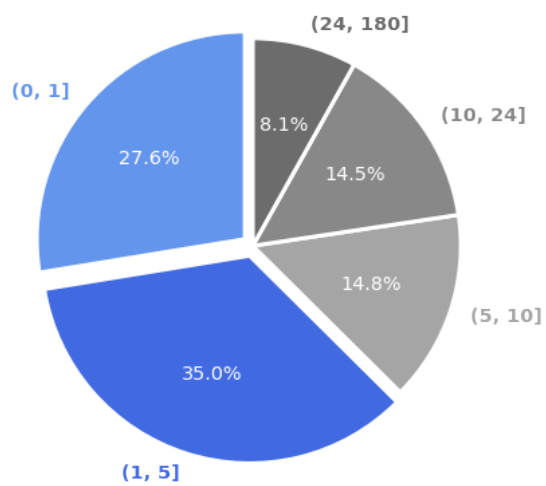
In order to test the transferability of the approach described in Section 3.4 to the Discord use case, the requirement to create a dataset that pairs discord help inquiries to identified related SO posts or even duplicate so questions need to be fulfilled. In order to ease the expensive and time-consuming task of dataset creation and annotation, the following approaches were used for pairing and labeling: Firstly, the conversations which contain a SO link were taken into consideration. SO links are detected using a regular expression which is run against the messages. It is a straightforward way of pairing help inquiries with SO posts which works under the assumption that if a community member provides a SO link, that link is probably relevant to the conversation. As a result, a pair between the conversation and the SO post can be noted.

The advantage of this approach is that we can base the dataset creation on real-world conversation pairing instances and use the methodology to extract pairs from additional communities and domains.

Therefore, a **Discord-SO** subset was created by finding such occurrences and following the



(a) Conversation Count per Range
Conversation Length



(b) Percentage of conversations per Utterance Range

Figure 5.1: Conversation statistics of Discord Community.

steps below to complete the dataset:

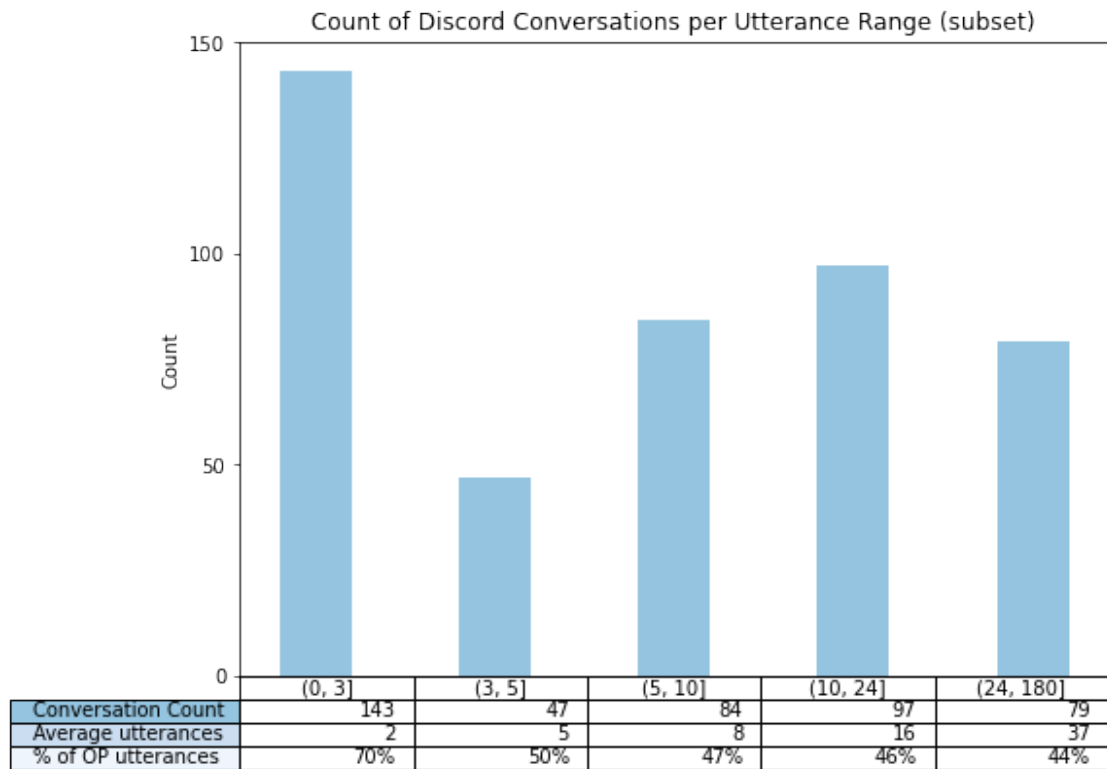
1. **Extract the SO question ID** to match to the Discord conversation ID. This entails the identification of the link type. In the examined conversations, three types of links were observed. The links to SO questions, links to a specific answer of a SO post, and the rest of the links refer to a general help page on the website. With the use of regular expressions, we can
 - extract the SO question ID (*qid*) from the identified URL,
 - extract the SO answer ID from the URL and query the Stack Exchange Application Programming Interface (API) to retrieve the corresponding question ID,
 - disregard the conversations (approx. 20) where the SO question has been deleted and is no longer available, and
 - remove the general help links which in all the examined cases referred to the same page. The page provides information on '*How to create a Minimal, Reproducible Example*'.
2. From the threads that are now matched to a SO question ID, it was verified that 114 were already present in the initial KU dataset.
3. The remaining 383 SO conversations that cannot be matched to the existing data points, had to be retrieved and cleaned according to Shirani et al. [SXL⁺19]. This step will ensure that the new questions, and in turn KUs, will match the existing questions and will not introduce any noise. We utilize the questions endpoint of the Stack Exchange API, along with a filter to retrieve the question and accepted answer. Additional information on the process is available in Appendix A.1. Moreover, the cleaning steps followed for the SO HTML rich text are explained in Section 5.1.4

The above steps brought the total of the paired Discord-SO dataset to 497 conversation pairs. Examining the distribution of the subset's conversations in terms of utterance count, we can see its differences in regard to the complete dataset. In Figure 5.2 we can see that 42.2% of the conversations have up to five utterances, (the same was at 62.6% in the whole community), 61% up to ten (versus 77% in the community), and 39% above ten (versus 22.6%). The main difference between the complete dataset and the extracted subset is that there are quite a lot of unaddressed messages in the former. Moreover, for the created subset we can say that roughly, one-third of the conversations consist of up to three utterances, one-third between three and ten, and the last one-third above ten.

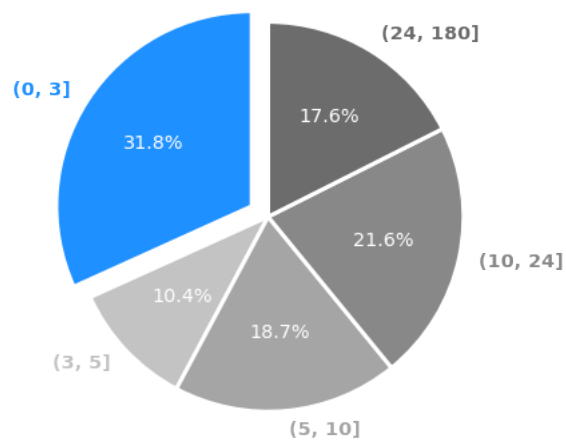
5.1.3 Annotation of Conversation Pairs

As we saw in the previous Section, the total number of the extracted conversations that contain a SO link was almost 500. These collected conversation pairs were manually labeled to classify the level of relatedness between the pairs. The three levels that were considered follow the model *ASIM-RV3* of Section 4.6. The resulting collection of labeled classes includes 155 duplicate conversations, 251 *related* (analogous to the *direct* class), and 68 *isolated* pairs, counts that translate to 53% related, 38% duplicate, and 9% isolated pair distribution within the dataset. The label *related* is favored over the *direct* one, since it is not technically the equivalent to the *direct* label that was introduced in Section 2.3.2. For the task of annotating the resulting *Discord-SO* conversation pairs the following points were taken into consideration:

The straightforward case of duplicate pairs is when the poster has already created a SO post of their problem and posted their question to Discord with a good enough explanation of their problem. An issue of course is if the user just posts their link without much explanation. But as long as there is a discussion that follows, the pairs are considered duplicates. Thus most of these



(a) Conversation Count per Range
Conversation Length



(b) Percentage of conversations per Utterance Range

Figure 5.2: Conversation statistics of Discord dataset

cases were labeled as duplicates except for the ones where a link is posted and there is not much of a conversation following and were labeled as isolated. Another duplicate case is the one where a question is asked with a relatively concise explanation of the problem, one of the first users to answer provides a SO link, and the conversation soon ends thereafter.

However, most of the conversations do not follow these patterns, and most often than not, an inquiring user provides code snippets, or screenshot errors and simply asks for help. These instances are the observed default case and naturally also account for the longer occurring conversations.

The biggest challenge is when conversations are not centered around a single topic but rather evolve as the conversation progresses. In this case, a linked SO conversation has to be considered only somewhat relevant to the conversation. Similarly, a conversation might be too long and cut off before the relatedness to a question even comes up. This is due to the embedding length limit the model has to impose. The reality is that many of these contain more than 100 utterances, so these cases were treated on a per-case basis, depending on the position of the SO link. In practice, the link was assessed based on its relatedness to the early messages of the conversation.

In some other cases, the original question has been deleted. These instances do contain some related messages to the SO link but they were not considered enough to support a satisfactory level of relatedness, thus they are annotated as isolated. One could argue that these instances could be removed from the dataset. Nonetheless, they were retained because they can be considered as conversations that do contain some signal of relatedness but are nonetheless not useful under proper context. Lastly, there are a handful of cases where a link is shared but the original poster finds the link not useful and ignores it altogether, so these instances are also marked as isolated.

A few examples of labeled pairs will be discussed in Section 5.2.1 in the context of evaluating the classification performance of the model.

5.1.4 Data Cleaning

Due to the format of Discord conversations, many users begin their questions by asking whether they can ask a question. This is such a common phenomenon, that the community moderators have added a community rule urging people to avoid similar behavior and rather simply ask their question. From a KU standpoint, such questions add noise to the model input, especially since the specific behavior is not as commonly seen in SO. In an effort to mitigate this issue, a **custom stopword** list was created that contains stopwords or frequently occurring bi-grams or tri-grams that can appear when a user is asking whether they could as a question, such as "ask a question", "need help", "can I ask", and "new here". It also contains abbreviations and slang tokens usually found in online informal discussions for instance "lol" and "btw". Lastly, it includes identified usernames and Discord commands such as "/reserve" and "/unreserve" that are targeted toward predefined bot actions.

Following the approach of Shirani et al., an important step in improving the quality of the dataset is removing code snippets. The core difference between SO and Discord is that the former represents the code in the form of HTML formatted text, whereas the latter is in the form of markdown code. Three cases of code were identified in the Dataset and were handled in the following way:

- **Markdown code blocks** is the ideal case, where the beginning and end of a code block are indicated with three backticks (""). These can be easily removed by using a regular expression.
- **Inline code snippets** are short snippets of code that contain a single line of code. While these cases can also be easily removed, it was decided not to since they often contain key method

names, error codes, and error messages, information that was deemed a useful signal for question relatedness. The snippets were further normalized by removing any punctuation and numbers.

- **Non-markdown code snippets** is the last and most unwelcome case because code is pasted as part of the conversational message text. These were handled by catering to the specific domain of Java, an approach that involved removing text that is included within curly brackets and was afterward normalized by removing any remaining punctuation and numbers.

Sharing a link in Discord results in the appearance of a small preview of the target URL which shows the title of the link along with a part of the page content. This shows up in the content of an embedded message within the extracted conversation which is utilized for the extraction of SO links. These short descriptions from the shared SO links have to therefore be removed to avoid introducing signal to the data. The same point is also highlighted in [SRM⁺18] by Silva et al., who measured a decrease in accuracy by up to 20% when such signals were removed from the AskUbuntu dataset. It is worth noting though that the Knowledge Unit dataset described in Section 2.3.2 reports this identified issue as taken into consideration when creating the dataset.

Text Normalization

With the term text normalization, we refer to the processing of human-generated text with the goal of reducing its randomness and bringing it as close as possible to the form of the text in the source domain, which in our case is the SO corpus. The preprocessing detailed in Section 4.3 was applied, which was previously applied to SO KUs. As an additional step, we handle the presence of emojis in the conversations, by removing punctuation and Unicode, which is not considered a loss of information since the aim is not to capture the sentiment. Since SO posts do not include emojis, removing them will actually contribute toward normalization.

5.2 RQ2: Classification Task

Using the identified Discord-SO pairs, the second research question that we seek to answer is the following: RQ2: How does the identified methodology perform on the new dataset and how do the results compare to the reproduced results?

The main goal of RQ2 is to examine how much classification power is retained when we introduce the new Discord conversations to the model. To answer this question and justify the confidence in the hypothesis that the transfer task is indeed feasible, we need to consider the construction of the task. Since the labeled Discord dataset is quite limited, the task needs to take into consideration the elements that can provide the best possible results on unseen, unsupervised data. Specifically in the case of low-resource "forums" like Discord, where labeled similar questions are not available, the choice of word embeddings can play a crucial role in performance. The ideal qualities of good word embeddings as highlighted by [PS19], should be:

- domain-specific, because we assume that the similarity of two questions is going to depend on certain domain-specific words. The domain specificity is covered by the fact that the embeddings are trained specifically on Java Conversations, even though their origin is strictly from SO posts.
- high-coverage, as a high number of unseen words, will probably harm the results. This is both a requirement and an indication of a good fit for the embeddings, which was successfully fulfilled by the domain-specific embeddings.

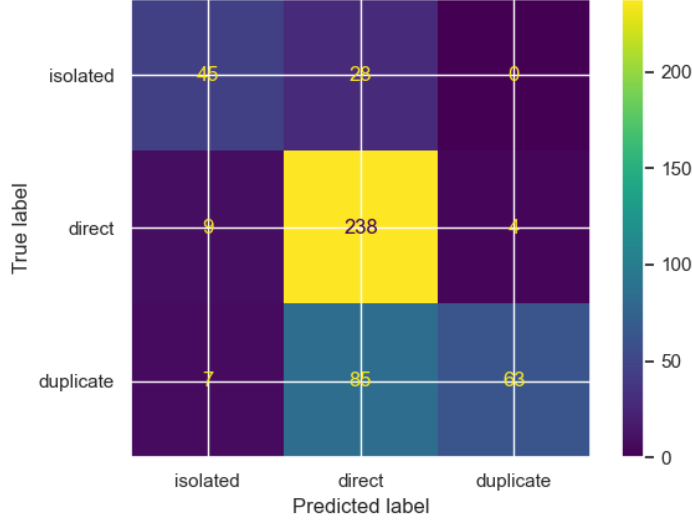


Figure 5.3: Confusion Matrix of Discord-SO Dataset Classification

Evaluation input	ASIM-RV3	
	Precision	F1 Score
Stack Overflow KU Test	0.91	0.91
Discord Conversation	0.77	0.70

Table 5.1: Classification Evaluation of Datasets using Complete Conversations as Input and the **ASIM-RV3** model

- capture lexical semantics, which is successfully achieved by using the GloVe embedding paradigm. This method achieves word representation in a way that captures the meaning, semantic relationships, and global context of each word used.

Simultaneously, the embedding vocabulary coverage can once again be used for the new dataset, to verify that there are not many new OOV words that can introduce noise. Indeed, the results show that the new domain is covered by the pre-trained embeddings quite well, as the vocabulary coverage is 99.85% and the OOV instances 0.28%, i.e 183 out of the 63873 total words. While these measures can certainly be a positive indication of a good performance on the Discord Java domain, they are not sufficient on their own. That is because, while a word can be recognized and is part of the GloVe vocabulary, the meaning of words and the semantic interaction between words can vary between different domains. However, the fact that the domains are kept almost constant between tasks by design, makes the domains homogenous, and thus the transferability is further supported by research that shows positive results in analogous cases [SLM⁺18].

5.2.1 Results

Testing the direct transferability hypothesis is straightforward after the dataset creation as it uses the same evaluation method used in Section 4.6.

The evaluation of the Discord-SO dataset was performed by using the model **ASIM-RV3**

which was described in Section 4.6. As a reminder, it considers three levels of question-relatedness and used complete conversations (KUs) for training. At its core, the evaluation takes the identified full conversation pairs between Discord and Stack Overflow and passes them through the trained model, which predicts their relatedness. The results in Table 5.2.1 show a comparison of the model performance on SO data versus the new Discord data. The model achieves 0.77 Precision and 0.70 F1-Score, which is weighted to account for the class imbalance in the dataset. In contrast, the evaluation performance of the source SO dataset is 0.91 for both metrics. The confusion matrix in Figure 5.3 gives us more information about how well the model is able to predict each class. It seems that the model tends to classify pairs as related, something that makes sense given the way the dataset pairs are organically occurring.

We can perform manual analysis for some of the classified pairs and gain a better insight into the results. Table 5.2 contains examples of correctly classified pairs, while Table 5.3 showcases misclassified examples. For the classification, the short version of the Discord conversation was used and the complete SO Knowledge Unit as usual. The tables present a snippet of the KU for brevity reasons, but an extended view of conversations as well as links to the community and SO posts are available in Appendix B.

We can examine the correctly classified pairs in Table 5.2 first. In Example 1, both conversations refer to sorting arrays. The SO question is asking specifically about sorting using a custom comparator. In the Discord conversation, the OP is having trouble sorting the array, and the conversation concludes with someone suggesting a custom int comparator, but it arrives at that suggestion after some further analysis of the specific issue, so the two conversations were deemed as semantically similar but not duplicate. Similarly, in Example 2, the Discord OP asks about formulating HTTP requests and how to add information to the URL, while the SO post is about the URL Encoding Parameter of the native java-http-client. As for a duplicate pair, in Example 3 the OP explains their issue and mentions the SO post. The question was deemed as a very accurate paraphrasing instance of the SO post and was labeled and classified as duplicate.

In Table 5.3, Example 4 was labeled as *duplicate* because both questions refer to the usage *chdir* in java and both answers refer to the usage of a relative path. Similarly, Example 6 was also labeled as a *duplicate* pair because the user has posted both questions on the two platforms. The model predicted the pair as *isolated* and the second as *related*. Perhaps the first case is because *chdir* refers to a non-native term to the java domain and the model has not seen similar question pairs, while the second Discord post, potentially does not contain enough detail. The common denominator in Examples 4 and 6, is that both are short in length and as we have seen, the longer conversations tend to produce better classification results. Example 5 was labeled as *related*, but the *duplicate* class was predicted instead.

Considering the above, one can support the argument that some noise might still exist in the dataset that does not allow for the optimal semantic alignment of the conversations and that the model could benefit from training on more question pairs. Nonetheless, the fact that the model can detect semantic similarity within unseen conversations is a significant outcome that can potentially be utilized for the retrieval of similar conversations.

Table 5.2: Examples of Discord-SO pairs, along with their assigned class(label) and the predicted class by ASIM-RV3

Ex.	Discord Conversation	Stack Overflow Title & Question	Label / Prediction	
1	<p><i>ID</i> : [977541076548222976]</p> <p>> Hey, I want to sort an Array of Indexes based on the value they represent in an int-Array. Why does my solution not work?</p> <p>>how does it not work are you trying to sort the array in place? if so you need to use Arrays.sort here you're just setting the variable to a sorted version Can you show any errors? it might be something to do with the variables given to the function</p> <p>[Fig:B.1]</p>	<p><i>ID</i> : [3699141]</p> <p>How to sort an array of ints using a custom comparator?</p> <p>> I need to sort an array of ints using a custom comparator, but Java's library doesn't provide a sort function for ints with comparators (comparators can be used only with objects). Is there any easy way to do this?</p> <p>[Fig:B.2]</p>	related	related
2	<p><i>ID</i> : [979813430401835078]</p> <p>> Good evening, I've got this nice and easy Proxmox Request with password and username as Paramters in Postman. Is there any way to recreate this exact thing in Java? I've tried using the HttpURLConnection but I found no way to do it and possible solutions just did not work.[...]</p> <p>> java has an http client, or multiple http client libraries that you can use to send parameters in the the request body/or as a form</p> <p>> I've tried to it with the HttpsURLConnection but there was litterally now way to add Fields in the Body and when searchin the Internet there was no solution that made it possible to add them in the body :/</p> <p>[Fig:B.3]</p>	<p><i>ID</i> : [56728398]</p> <p>Java 11: New HTTP client send POST requests with x-www-form-urlencoded parameters</p> <p>> I'm trying to send a POST request using the new http client api.Is there a built in way to send parameters formatted as x-www-form-urlencoded ?</p> <p>My current code:</p> <p>What I'm looking is for a better way to pass the parameters. Something like this:</p> <p>Do I need to build myself this functionality or is something already built in?</p> <p>I'm using Java 12.</p> <p>[Fig:B.4]</p>	related	related
3	<p><i>ID</i> : [971797369874182194]</p> <p>> I am using the json.simple package. Let's assume I have a parsed JSONArray from the file:I am trying to put the contents into a HashMapString, Object(). Would anyone happen to know how to iterate over this JSONArray to get the keys and values? I found this, but this is exact while I need relative: URL</p> <p>[Fig:B.5]</p>	<p><i>ID</i> : [33595941]</p> <p>How to convert JSONArray to Hashmap</p> <p>> I've been able to get a jsonarray from a json string, but don't know how to put it in a Hashmap with a String that shows the type of cargo and an Integer showing the amount.The string:</p> <p>This fixed it for me:Try creating a new Hashmap, and loop through the JSONArray, adding each element to the hashmap.I've been able to solve this using the Google GSONWhere "yourJsonString" is the whole json that contains the cargo json array.</p> <p>[Fig:B.6]</p>	duplicate	duplicate

Table 5.3: Examples of Discord-SO pairs that were misclassified by ASIM-RV3

Ex.	Discord Conversation	Stack Overflow Title & Question	Label / Prediction	
4	<p><i>ID</i> : [963170643640213576] > Is there an equivalent of chdir() in Java?> No I don't think there is <URL> Also what's the point of such a function use relative path [Fig:B.7]</p>	<p><i>ID</i> : [43130952] Why chdir does not support in java? > I would like to know the actual reason behind why Java does not provide chdir call implementation however it provides native interfaces for the same. [Fig:B.8]</p>	duplicate	isolated
5	<p><i>ID</i> : 904644087200755723 >Need help At this point i screwed up my code only runs if i import java.util. concurrent.CopyOnWriteArrayList; but i want it to just work as an arraylist at this point im screwed i wasted 3 days doing the wrong this could i please get help This is the zip of my three files Term class Polynomial class and Tester class Could I please get some help</p> <p>>Why are you using an `Copy-OnWriteArrayList` just use a normal `ArrayList` [Fig:B.9]</p>	<p><i>ID</i> : 223918 "Iterating through a Collection avoiding ConcurrentModificationException when removing in loop"</p> <p>>"We all know you ca not do this ConcurrentModificationException etc.. this apparently works sometimes but not always Here's some specific code This of course results in e .. even though multiple threads are not doing it.. Anyway What's the best solution to this problem How can I remove an item from the collection in a loop without throwing this exception I am also using an arbitrary Collection here not necessarily an ArrayList so you ca not rely on get. [Fig:B.10]</p>	related	duplicate
6	<p><i>ID</i> : 959725068391432212 > does anyone know about jpa and hibernate here? <URL> does anyone know why I am getting two id's inserted into my user_profile table with spring jpa and hibernate? figured it out, thanks anyways [Fig:B.11]</p>	<p><i>ID</i> : 71714015 Why is my Entity class inserting two Ids into my table? > Spring Boot JPA I have two Entity classes User and UserProfile. The User table has a primary key for the User ID which is a long data type. This User ID is supposed to be the primary key in user_profile as well. User also has email as a column, which I want to be linked to the user_profile also. The issue I am having is that for some reason, a column named id is being inserted into my table when I already have the primary key user_id set in the user_profile table. Does anyone know what I am doing wrong? User: User_Profile: User table: User_Profile table: [Fig:B.12]</p>	duplicate	related

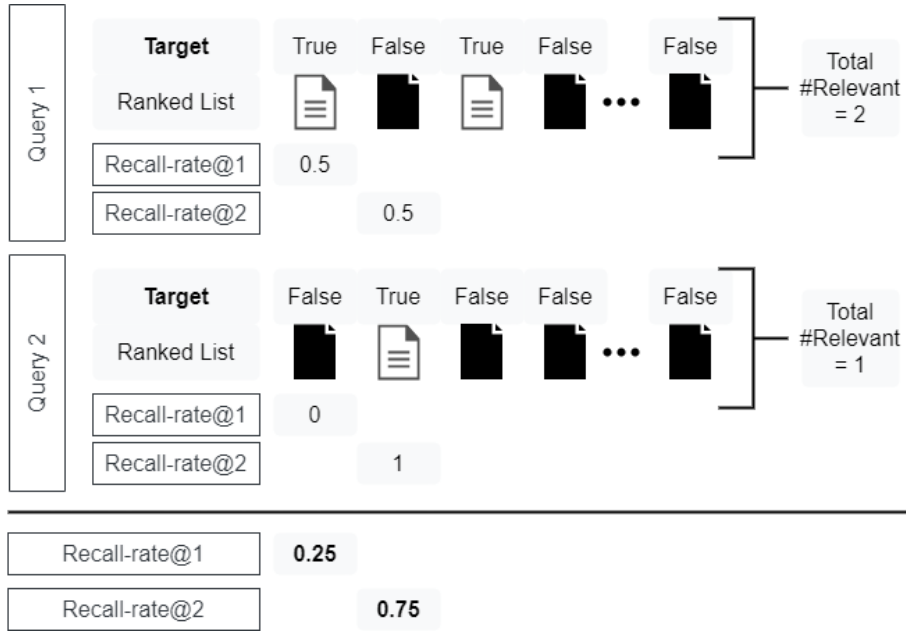


Figure 5.4: Example of the retrieval score calculation as defined by the Pytorch framework.

5.3 RQ3: Semantic Retrieval Task

In the previous Section, we tested how well the created model can perform on unseen Discord Conversations under the premise of the classification task. The next question to be examined is the following:

RQ3: Can the same methodology be used for semantic question retrieval?

In other words, can it maintain the performance and does the model have the ability to identify semantic relatedness when used for retrieval tasks?

In practice, the inference function needs to evaluate a given query's semantic interaction with each available SO post in our available dataset. Therefore, for each conversation in the Discord dataset(query), pairs of KUs between the query and all the SO posts need to be created and fed to the model in batches, to avoid running into memory issues. We evaluate the model outputs by calculating the total retrieval *recall-rate* for all queries. The number reported is the average of five runs due to the small dataset size, similar to [RMG19]. Following the paradigm of previous work, Silva et al. [SPM18] and Zhang et al. [ZSL⁺18] used approximately 1000 questions for ranking the evaluation. For this reason, 1000 random SO posts are selected for each evaluation as a sample of the 160K questions. This is also a way to calculate the recall rates of our small dataset while eliminating the majority of potentially false negative results.

A note on the calculation of *recall-rate@k*. As discussed in Section 3.5, the *recall-rate* assumes a single relevant result for each query. To implement the generalized case, the retrieval-recall metric implementation from Pytorch was utilized, which calculates the fraction of relevant documents that were retrieved among all the relevant documents. *Recall-rate@k* calculates the fraction of the relevant results in the top k results, among all the relevant documents. This means that if, for instance, $k = 1$ and the first returned result is relevant, but the total amount of relevant results is two, the *recall-rate@1* for this query is 0.5. Naturally, if the k is adapted to the maximum number

Dataset	recall@2	recall@5	recall@10	recall@20
Discord-SO (Full Conversation)	0.39	0.55	0.69	0.80
Stack Overflow KU	0.36	0.47	0.57	0.65

Table 5.4: Retrieval Recall Performance of the best performing model (ASIM-RV3) on the Discord-SO dataset and the Stack Overflow dataset.

of relevant results, then the metric is not affected by the aforementioned limitation. Therefore, we adapt the lowest k for the evaluation of the current discord-SO dataset at $k = 2$. Figure 5.4 shows an example that illustrates the above point for an evaluation run of two queries.

In addition, the metric assumes the binary case for the result relevancy, which means that it considers a retrieved post either as relevant or non-relevant. To adapt the metric for multi-class labeling, we consider a result relevant if it is classified with a positive class and non-relevant otherwise, and the probability of relatedness is the combined probability of the positive classes.

5.3.1 Results

To benchmark the *recall-rate* for the Discord-SO dataset, we can again use the source dataset of SO. A randomly created subset of the test dataset is utilized, to create a dataset proportionally similar to the target dataset. For the same reason, the complete conversations are used as the query inputs and the retrieval results are used to calculate *recall-rate@k* for $k \in \{2, 5, 10, 20\}$. The experimental results are presented in Table 5.4 and show that the model performs equally well on both datasets. Specifically, for the Discord-SO dataset *recall-rate@2* is 0.39 and reaches 0.80 for *recall-rate@20*.

Comparing these results with the previous work that was examined in Chapter 2, *ASIM-RV3* performs better than the original, the replication of *DupPredictor* [SPM18], and better than *DQ-LSTM* overall ([WZJ19]) who reported a *recall-rate@20* at 0.59. Conversely, our model performs worse than *WV-LSTM* [WZJ20] who report a *recall-rate@20* of 0.82 in the Java Question group. Although these benchmark results are not directly comparable, as they were evaluated on different datasets, they are nonetheless provided as an indication of performance in past work.

The performance of the Stack Overflow dataset in the retrieval task indicates that while the model is able to identify semantic relatedness between conversations as seen in 5.2.1, it needs re-training to predict the scoring between pairs. The same becomes more apparent if we consider that the Discord-SO pairs are more distinct within the dataset and in a Knowledge Unit graph context, would have relatively few connections with other nodes. The same could explain the slightly better performance in the Discord-SO dataset for this task.

5.4 RQ4: Conversation Length Variability

The motivation behind *RQ4* is ultimately drawn from the motivation of the thesis. In order to develop an applied solution that can be used in the help channels of Discord servers, one has to keep in mind how the inference is performing in real-time. Therefore, a significant parameter to consider is the available information at runtime. When a user is asking for help, they initially form their query in some type of a question. Ideally, we would like to successfully identify similar questions from the moment the inquirer forms their first sentences. This is arguably the case in which the most community effort is avoided and the inquirer gets the most value. However, the best case is rarely the most realistic one, so we can also consider the case where the inquirer provides some further clarifications, potentially even after the first responses of other community members. The goal is to explore how well the model performs under the aforementioned circumstances. In other words:

RQ4: How is model performance affected when different parts of the conversation thread are considered as input?

To answer this question, two levels of conversation length with respect to utterance count will be considered. As a reminder, by the term utterance under the examined Discord domain, we define all the consecutive messages of a community member before a message from another member appears in the conversation. One conversation length that will be examined is the first utterance of the user who initiates the conversation, whom we call OP. The second will examine the first utterance of the OP, a first utterance from another user, and a second utterance of the OP. In case a third user joins the conversation before the second OP's utterance occurs, the conversation length is limited to the first two utterances.

In terms of the model evaluation, we examine how the metrics are affected when we use as input the aforementioned conversation segments, and we evaluate both the classification and retrieval tasks. The performance will thus be compared to the observed performance of *RQ2* (Section 5.2) and *RQ3* (Section 5.3) for which the complete conversation thread was used. In addition to the *ASIM-RV3* model that was used so far, we will evaluate the above using the *ASIM-RV5* as well. To reference Section 4.6, the second model was trained using only the title and question body of *KU1* and the complete conversation of *KU2*. By using that as well, we seek to evaluate what effect it will have on the Discord dataset when its conversations are similarly limited in length.

5.4.1 Results

The question can be answered by the observations in Table 5.5. Considering the classification task in Table 5.5a, we see that the longer the conversation, the better the performance. The difference between using the full conversation versus the first question as an input translates to a difference of roughly 20% in both Precision and F1 Score. We continue to see that the model which was trained on shorter *KU1* conversations, i.e. *ASIM-RV5*, performs worse than the standard *ASIM-RV3* and the same occurs in every Discord conversation length. One might have expected the limited input model to perform better with test data that have limited input, but we clearly see that there is no benefit in a separate model.

Continuing on the retrieval task and Table 5.5b, we can again see the trend of longer input performing better. The *recall-rates* for each *k* were produced as the average of five evaluation runs, and for each run, the retrieval dataset was enriched with random questions from the complete SO Knowledge Unit Corpus. The detailed results of each run can be observed in Table A.1, but we can clearly see that the retrieval results are quite stable for a rather small dataset and unseen

Length of Discord Input	ASIM-RV3		ASIM-RV5	
	Precision	F1 Score	Precision	F1 Score
Full Conversation	0.77	0.70	0.68	0.60
Short Conversation (Two OP Utterances)	0.68	0.66	0.64	0.60
First question (One OP Utterance)	0.61	0.58	0.60	0.55

(a) Classification Evaluation of Discord Dataset using Different Combinations of Conversation Length and Models

Length of Discord Query	recall-rate@2	recall-rate@5	recall-rate@10	recall-rate@20
Full Conversation	0.39	0.56	0.69	0.80
Short Conversation (Two OP Utterances)	0.34	0.51	0.65	0.75
First Question (One OP Utterance)	0.28	0.43	0.55	0.64

(b) Retrieval Recall Performance of the best performing model (ASIM-RV3) on the Discord-SO dataset using different query length.

Table 5.5: Classification and Retrieval Evaluation of Different Conversation Lengths.

Discord data. Irrespective of input length, the average *Recall-rate@2* ranges between 0.28 and 0.39, whereas *Recall-rate@20* between 0.64 and 0.80.

The better performance of longer conversations could be explained in two ways. The main reason could be that the conversations were labeled while taking into consideration a sizeable part of the conversation and not necessarily the first question. Another reason could be that longer conversations simply contain more signal, which helps in attributing more semantic relatedness to pairs. Ultimately though, the exact point in the conversation where a user is presented with useful information is less relevant than the value that the results provide and their utilization. However, a way to both improve performance on shorter conversations and increase the rate of relevant results in the top retrievals, is to create or find datasets that explicitly rank the pairs by relatedness, so that the model can learn a ranking function.

5.4.2 Retrieval Examples

In addition to the experimental results, we can also evaluate the ranking performance by manually reviewing a few ranking results and empirically assessing their relatedness. For retrieving the top candidate results for a conversation, we consider the input length of a *Short Conversation*, i.e. the first two utterances of the inquirer. Tables 5.6 and 5.7 showcase two retrieval examples. The left columns contain the Discord Conversation queries and the right columns the *top-5* retrieved results. Once again, the complete conversations and original links are available in Appendix B.

The first example of Table 5.6 showcases the way the retrieval evaluation is performed. This means that the retrieval corpus includes all the SO posts associated with the Discord dataset plus an additional 500 random posts from the complete corpus. For this example, an identified as a frequently asked question is selected, because it has a higher chance of retrieving multiple

relevant results even with the reduced dataset. Indeed, among the top results, three of them can be considered relevant, and specifically, the first one is the identified duplicate question from the Discord-SO dataset. In other words, this is a query for which $recall@1 = 1$.

The second example of Table 5.7 involves an already visited conversation (Ex. 2 of Table 5.2) which was classified as related with its SO pair. The goal of this example is to explore the value behind related pairs. Another difference in this example is that we are using the complete corpus for retrieval. The retrieved results are ranked by the maximum combined probability of the positive classes, or simply by the minimum probability of the negative class (isolated). At first look, it appears that the identified as relevant SO question is not present in the top-5 results and not in the top-20 results either. By tweaking the ranking to prioritize the classified as duplicate results with the highest probability, we get the results shown in the table. We can easily infer that the results are rather relevant to the query, while especially the second result is also close to the identified as related original pair.

In conclusion, this chapter followed the process of mining and creating the novel Discord conversation, verified the generalizability of the ASIM-RV3 model in terms of identifying semantic relatedness through the classification task (RQ2), and explored the possibilities of adapting the methodology and the dataset to a retrieval task (RQ3). The experimental results for the retrieval task showed that performance is analogous to previous work cases and empirical evaluation revealed positive signs of model utilization for near-duplicate question mining. Regarding RQ4 and the variability of input-query length, we observed that our current model performs better when longer conversations are considered at inference time.

Table 5.6: Example: Retrieval Results Using Short Conversation Query

Discord Conversation	k	Top-5 Retrieved SO Posts
<i>ID</i> : [966404222058893322]		
When the if / if else conditions are true why the `String english = scan.nextLine(); and String sentence = scan.nextLine();` does not get executed the program just exists with no errors Whats your expected output and whats the actual output ? And why should both if and else if both be true ? the user should enter a morse code and the program will translate it to english words opposite for the else if condition And the tranlation doesnt work? [Fig:B.13]	1	<i>ID</i> : [13102045] Scanner is skipping nextLine after using next nextInt or other nextFoo > I am using Scanner methods nextInt and nextLine for reading input Basically it looks like this The problem is that after entering the numerical value the first input.nextLine is skipped and the second input.nextLine is executed so that my output looks like this I tested my application and it looks like the problem lies in using input.nextInt If I delete it then both string1 = input.nextLine and string2 = input.nextLine are executed as I want them to be. [...] [Fig:B.14]
	2	<i>ID</i> : [17443201] Why doesn't == work on String > I just started Java programming I love it so far but I have been stuck on this problem for a while When I run this code whenever I type in "boy" it will just respond with GIRL Why [...] [Fig:B.15]
	3	<i>ID</i> : [12273794] Scanner nextLine NoSuchElementException > I\'ve been programming a Bukkit plugin for a while now and this one issue has me stumped I am trying to read a line from a file using a Scanner and add everything on the line before to a Hash-Set Every time I try I get a NoSuchElementException looking like this This is the class where I try to do what I explained above I\'ve been told that I am calling in.nextLine twice and when I asked for advice on how to fix it he just ignored me [...] [Fig:B.16]
	4	<i>ID</i> : [12544564] Why this output when comparing args 0 with a string > When I run the following code as java XYZ BOY the output is Invalid type how However when I use args 0 .equals BOY it gives the desired output I want to know why things go wrong here when I do not use String.equals [...] [Fig:B.17]
	5	<i>ID</i> : [36096097] I am doing an assignment and I keep getting a NullPointerException error in Processing what is this and what can I do to fix it > I am writing this code for an assignment and I would greatly appreciate anyone\'s help It\'s supposed to be a video game it\'s more like a choose your own adventure game and I keep getting errors that say NullPointerException On top of that I ca not get my images to show up and my video in the code is not playing If anyone could possibly give me some pointers [...] [Fig:B.18]

Table 5.7: Example: Mining Results Using Short Conversation Query

Discord Conversation	k	Top-5 Retrieved SO Posts
<i>ID</i> : [979813430401835078]		
<p>> Good evening, I've got this nice and easy Proxmox Request with password and username as Parameters in Postman. Is there any way to recreate this exact thing in Java? I've tried using the <code>HttpsURLConnection</code> but I found no way to do it and possible solutions just did not work. Best solution I could do till now is to add them as Parameters in the URL but that's not save and would basically mean that I'm sending my passwords across the Internet xD</p> <p>> java has an http client, or multiple http client libraries that you can use to send parameters in the the request body/or as a form [...] [Fig:B.3]</p>	1	<p><i>ID</i> : [4185320]</p> <p>hitting a URL using java</p> <p>> How to use <code>java.net.URLConnection</code> to fire and handle HTTP requests using java how do i hit any url for instance opening of <code>http www.xyz.com</code> node1 in a browser will tell xyz.com that node1 is hit so in this java program which sends sms text say <code>\`node1\`</code> in example above embedded in the url itself to a sms gateway server how do i achieve it without opening any browser or using servlet.</p> <p>>You can use an <code>HttpURLConnection</code> [...] For more advanced flexible URL fetching you could use <code>HttpClient</code> <code>http hc.apache.org httpclient-3.x</code> [...] [Fig:B.19]</p>
	2	<p><i>ID</i> : [7487341]</p> <p>Send Http Post Message using a stream in Java</p> <p>> I have to send an http post message to a server This message must have an header and a body and to send it i must use a Stream Both the header and the body must be an array of Byte</p> <p>> From the server you can get header information separately from <code>HttpServletRequest.getHeaders</code> [...] [Fig:B.20]</p>
	3	<p><i>ID</i> : [13056980]</p> <p><code>HttpClient</code> with redirect</p> <p>> I am making a request using <code>httpclient</code> and after receiving the response I get automatically redirected to another page How can I stop this from happening Here are some more details I am trying to get a token from Google using <code>oAuth2.0</code> I make the request for the token from the server using <code>httpclient</code> but because of the <code>redirect_uri</code> I get somehow redirected [...] [Fig:B.21]</p>
	4	<p><i>ID</i> : [11328043]</p> <p>Java What is the purpose of <code>setDoInput</code> in <code>URLConnection</code></p> <p>> Would this method be used for example if I want to read the response to a POST method call from the server I am connected to [...] [Fig:B.22]</p>
	5	<p><i>ID</i> : [12888780]</p> <p>how to send HTTP request to a servlet</p> <p>> I only have one servlet running on the Tomcat server side Now I want to send a HTTP request to this servlet from a Swing application and it\'s not an <code>APPLET</code> application because I see some examples sending request from applet [...] [Fig:B.23]</p>

Discussion

In the previous two chapters, we presented the results of two semantic similarity tasks on a small and novel Discord dataset. This chapter will discuss the merits and applicable use cases for the presented methodologies, as well as their limitations, and guidance for future work.

The performed experiments showed that the presented methodologies work quite well in the examined tasks of near-duplicate question classification and semantic search. For the second task, in particular, the work examined both symmetric and asymmetric approaches with respect to the query input, i.e. it sought to retrieve relevant KUs using both full conversations and short questions as a query. The experimental results of the first task, firmly support our belief that the solution has proven capable of detecting semantic similarity, a capability that we argue can facilitate more semantic relatedness tasks.

A prime candidate task is near-duplicate question mining. Given a large corpus of Knowledge Units, and a separate corpus of input queries, the model could be used to create semantically relevant pairs with little to no manual effort depending on the methodology. Furthermore, the semi-automated creation of the Discord-SO pair dataset which is based on real-world conversation pair instances can be further utilized. The same methodology can be applied to extract pairs from additional communities and domains. Manually identifying relevant questions to new conversations is not only time-consuming but also requires a certain level of domain knowledge to effectively perform the matching. And certainly as the question complexity increases, so does the need for nuanced domain understanding. There might still be a need for human ranking or evaluation of the results, but the effort is certainly brought to a minimum with an automated pairing approach. The analysis identified many cases where links from other websites were shared under a similar context to the SO links, which indicates that help is not limited to SO. In fact, for the purpose of creating a sophisticated solution that can be utilized by a community, the corpus can be expanded to include documents from other sources such as the language's documentation pages and other resource-rich websites. That would constitute a more complete semantic information retrieval solution that certainly would require significant amounts of data.

6.1 Limitations

One of the biggest limitations of the approach is probably the scarcity of labeled data and the bias that might be introduced while labeling them. Certainly, good domain knowledge is essential during the creation of conversation pairs, especially if the task is to score the relevancy of candidate results. This work aims to avoid this limitation by automatically finding the pairs, but the need for scoring the results was not mitigated entirely. In terms of labeling bias, it can be mitigated in the future by being manually labeled by more researchers and ensuring that the

inter-rater agreement is acceptable. A good Cohen’s Kappa inter-rater agreement score [McH12] would provide more confidence in the results. However, the two positive labels were not taken too much into consideration for the retrieval task, as we are evaluating whether a positive class is present in the results overall. So, the retrieval task is not currently significantly affected by this limitation.

Furthermore, the observed performance at inference time was less optimal when considering performance that is fitting to a scalable retrieval solution. Granted, the inference was performed on a CPU and had to use a reduced batch size due to memory, but there are certainly ways beyond the use of a GPU to speed up the inference and support the scalability of a future solution, which will be discussed in a following Section.

6.2 Threats to Validity

The following Section discusses a few fundamental threats to validity as it pertains to the used data, methodologies, generalization potential, and bias.

Starting with threats to **internal validity**, the term refers to errors in the experimental data and methodology implementation. The approach taken for this thesis is based on pre-existing methodology and data that have proven effective in similar tasks under specific conditions according to research standards. We made sure to evaluate each step to ensure that each one was fitting for the applied domain. Furthermore, checks were performed to ensure that noise and bias were not introduced in the dataset by ensuring the uniqueness of pairs within the dataset and by removing duplicate-question signals from the Discord conversation text (Sec. refsec:dataset-exploration).

Threats to **external validity** refer to the generalizability of the proposed methodology. We made sure that the work focused on carefully selected methodologies that have proven successful in terms of their generalizing strength. Complimentary to the past research, are the findings from this work as well. We believe, that given the careful selection of data and methodologies, the approach will be able to produce similar results in more domains. In addition, the usage of a small-sized dataset does pose a threat to external validity, but in the future, such threats can be mitigated by expanding the Discord dataset to include more question pairs as well as new domains to extend the generalizability of the methodology to communities beyond Java. As a result, the research will have more data to train and test the performance of a methodology. In addition, a fundamental next step is to perform a user study, in which people that have a certain level of domain knowledge can evaluate the retrieval results and rate how helpful they are.

Construct validity relates to the identification and usage of the appropriate measures during evaluation. To mitigate the threat, we performed an extensive analysis of the appropriate evaluation metrics per task, while referring to previous work to ensure best practices are adhered to. Thus we believe that the use of precision and *F1 score* is pertinent to the classification task and *recall-rate@k* to the retrieval task. Furthermore, the reasons why other commonly used metrics were not utilized, are presented and discussed. Therefore, we argue that there is little to no threat in terms of construct validity.

Finally, **instrumentality bias** is a threat to the reproducibility of the results, which is mitigated by providing the code, data, and a detailed description of experiments and implementation.

6.3 Toward an Efficient, Real-Time Neural Retrieval Application

When designing a solution that can retrieve fast, reliable, and ultimately useful results for developers, one has to consider the fundamental requirements of real-time usage. The ideal applied solution must work as a semantic search engine that can almost instantly provide the most relevant results from an input query. It is true that the utilized solution can successfully identify semantic relatedness between candidate results and with further training, we believe that it can be used for effective semantic retrieval under certain conditions. However, in terms of efficiency, it requires each candidate pair to be passed through the network in order to calculate their alignment through attention and produce their sequence representations.

Certainly, an efficient solution should have minimal latency which means that evaluating every pair is not practical, and especially as the candidate corpus increases, the solution's scalability and feasibility suffer. This means that in order to build a real-time neural retrieval application, it is crucial that the embeddings of the search corpus are already calculated, ideally hashed for fast retrieval, and stored so that they can be easily accessible.

Retrieve and Rerank

To avoid the computational complexity of comparing the query with each embedding in the corpus, the solution has to involve two steps. The first step needs to efficiently limit the search space of the query. It has to be fast and inexpensive and can be as simple as performing a lexical search with BM25 or TF-IDF, or it can also be a bi-encoder model that is trained for semantic search and can be very computationally efficient. The goal of this step is to eliminate the most irrelevant results and limit the candidate results to a small subset. The next step involves a trained model that can perform the reranking step. This is where our trained model can be used and utilize the limited candidate list of the first step and rerank the results to produce the final output.

Another approach involves the usage of Fast Near Neighbour Lookup using (Facebook AI Similarity Search (FAISS)) [JDJ19,JDJ18] for retrieval, which allows for quick comparison between embedding vectors by inputting the faiss index of the embedded question, and then returns the desired number of the most similar documents. This of course requires a solution that allows for pre-computed sentence embeddings of the corpus which can be achieved with the bi-encoder architecture of sentence transformers [RG19], [WK20].

Conclusion and Future Work

7.1 Conclusion

This thesis examined past research in duplicate question detection for SO posts and reproduced the state-of-the-art model successfully. In addition, a Discord software-related community was explored, and a dataset of approximately 500 conversations was created. Mining Discord conversations and SO pairs led to the creation of a test dataset that was annotated for evaluation. The transferability of the model was then tested against the created dataset in a classification and retrieval task. Lastly, the reusable methodology can be an aid for expanding the examined communities and domains as well as for the task of creating and enriching novel datasets.

7.2 Future Work

There are two directions future work could take with respect to the Neural Retrieval Task and more specifically in terms of the type of search it will seek to implement. A clear distinction should be made in terms of whether a Symmetric or an Asymmetric search is the desired approach (Sec. 1.3.3). We propose that future work focuses on an asymmetric approach, where a single question or small parts of the conversation are used as query input to retrieve Knowledge Units from the corpus. This way, different parts of the same conversation can contain multiple queries and thus the results are less affected by shifts in conversation. More importantly, a Retrieval-Rerank approach should be utilized, to make the retrieval as efficient as possible and mitigate bottlenecks in retrieval. The retrieval results could potentially be improved by using a dataset that provides relevance-scoring of questions, information that a model could utilize to finetune relatedness.

Ultimately though, at the forefront of any future work should be the evaluation of a neural retrieval application in terms of helpfulness for developers who can empirically evaluate the performance. Therefore, we propose that future work considers a study in which the participants are provided with a top-k list of relevant posts to various input queries and can evaluate the helpfulness of the retrieval results by adding a helpfulness score to the results. The scoring results can further be used as training data in a reinforcement learning task that will aim to further improve the reranking by predicting relatedness scores.

The expansion of the labeled Discord Conversation and SO pairs could really facilitate future research. Given the automated mining of conversation pairs described in section 5.1.2, more Discord community servers can be parsed with little effort. The same would also expand the task domain to more languages and frameworks. The existence of a larger dataset would allow

for the implementation of other techniques such as Adversarial Domain Adaptation, that seeks to blend the source and target domain and make the two indistinguishable to the model, thus gaining more generalization power. It is worth mentioning that since the beginning of this thesis, there has been a small paradigm shift in the examined Discord community. The help channels are slowly phased out and a new way of asking questions is introduced. The new channel allows for distinct conversations that have a structure that includes title, question, tags, etc, and more closely resembles the SO question-answering format. Certainly, if this new beta version becomes the standard for Discord help channels, it will facilitate further research in the field. Finally, given that Discord conversations rely heavily on code snippets, i.e users tend to ask for help by providing code than trying to describe the issue in detail, similarity based on code could be explored in the future. The approach certainly comes with its own set of challenges, but some work is already been done in the space [SLL⁺18], [YDC⁺18].

Acronyms

ADA Adversarial Domain Adaptation	10
AI Artificial Intelligence	4
API Application Programming Interface	36
ASIM Attention-Based Sentence Pair Interaction Model	ix
AUC Area Under Curve	11
BERT Bidirectional Encoder Representations from Transformers	11
BEIR Benchmarking Information Retrieval	90
BiLSTM Bidirectional Long Short-Term Memory	11
CBOW Continuous Bag of Words	18
CCA Canonical-Correlation Analysis	11
CNN Convolutional Neural Network	10
CQA Community Question Answering	2
DQD Duplicate Question Detection	11
FAISS Facebook AI Similarity Search	53

FNN Feedforward Neural Network	28
GloVe Global Vectors for Word Representation	11
GPU Graphics Processing Unit	28
HTML HyperText Markup Language	10
ID Identifiers	13
IR Information Retrieval	6
JSON JavaScript Object Notation	61
KU Knowledge Unit	ix
LDA Latent Dirichlet Allocation	9
LSA Latent Semantic Analysis	90
LSTM Long Short-Term Memory	10
MAP Mean Average Precision	10
MOOC Massive Open Online Course	11
MRR Mean Reciprocal Rank	10
NLG Natural Language Generation	5
NLP Natural Language Processing	ix
NLU Natural Language Understanding	ix
OOV Out of Vocabulary	26
OP Original Poster	34

PCQA Programming Community Question-Answer	33
RNN Recurrent Neural Network	10
SBERT Sentence Bidirectional Encoder Representations from Transformers	6
SRDM Semantics and Relevance Duplicate Questions Detecting Model	ix
SoA state-of-the-art	4
SO Stack Overflow	1
STS Semantic Textual Similarity	6
TF-IDF Term Frequency–Inverse Document Frequency	10
URL Uniform Resource Locator	26
WV-CNN Word Vector Convolutional Neural Network	10
WV-RNN Word Vector Recurrent Neural Network	10
WV-LSTM Word Vector Long Short-Term Memory	10

Implementation Details

A.1 Stack Exchange API

To call the Stack Exchange API one should take into consideration the rate limits set in place by the service. From experience, the limit for requests for an application without an access token seems to be 300 per day when hitting the same endpoint - despite the limit of 10,000 stated in the documentation.¹ A wait time of 10 seconds between calls helps avoid getting a backoff response. According to documentation, the same care should be exercised when sending requests that return the same results semantically, reportedly due to the API's heavy caching implementation. To avoid hitting the endpoint for the same ID multiple times, the raw response of each query is saved in a JavaScript Object Notation (JSON). Before a new query, the JSON keys are checked against the query ID, and if a match is found the query is not sent but returns the saved JSON. To bypass the cache and refresh the output, the JSON dump file needs to be deleted.

¹<https://api.stackexchange.com/docs/throttle>

A.2 Discord Retrieval Runs

Full Conversation

#Run	recall-rate@2	recall-rate@5	recall-rate@10	recall-rate@20
1	0.33	0.49	0.59	0.69
2	0.40	0.56	0.71	0.81
3	0.39	0.55	0.69	0.80
4	0.43	0.60	0.75	0.85
5	0.42	0.60	0.73	0.84
Avg.	0.39	0.56	0.69	0.80

(a) Retrieval Runs with Discord Full Conversation Input

Short Conversation

#Run	recall-rate@2	recall-rate@5	recall-rate@10	recall-rate@20
1	0.29	0.53	0.64	0.73
2	0.35	0.50	0.64	0.73
3	0.34	0.48	0.62	0.75
4	0.36	0.54	0.69	0.78
5	0.35	0.51	0.66	0.76
Avg.	0.34	0.51	0.65	0.75

(b) Retrieval Runs with Discord Short Input

Single Question

#Run	recall-rate@2	recall-rate@5	recall-rate@10	recall-rate@20
1	0.21	0.46	0.55	0.65
2	0.28	0.41	0.54	0.62
3	0.29	0.42	0.54	0.62
4	0.31	0.44	0.58	0.67
5	0.29	0.43	0.56	0.66
Avg.	0.28	0.43	0.55	0.64

(c) Retrieval Runs with Discord Question Input

Table A.1: Retrieval Evaluations of Discord-SO pairs. For each run the Retrieval Dataset is enriched by Random Subsets of Stack Overflow Knowledge Units.

Discord-SO Conversation Examples

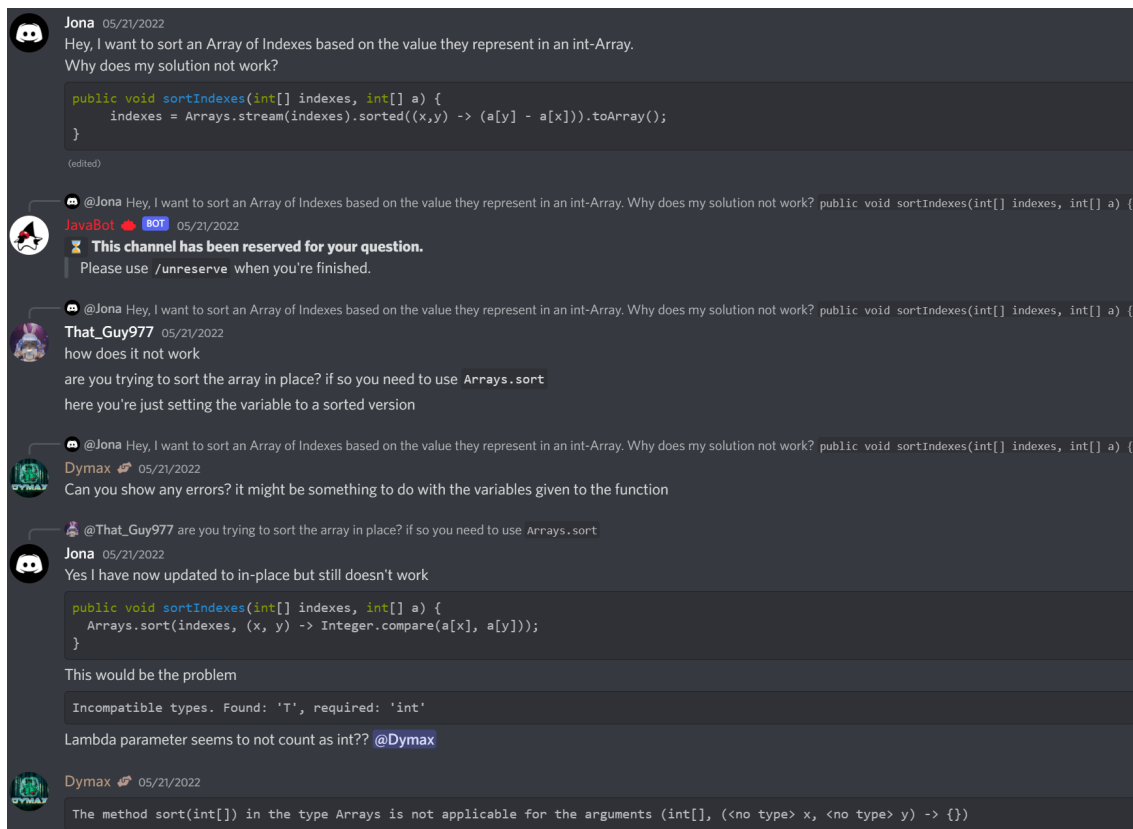


Figure B.1: Discord Conversation [#977541076548222976](#).

How to sort an array of ints using a custom comparator?

Asked 12 years, 2 months ago Modified 1 year, 8 months ago Viewed 138k times

98 ▲ I need to sort an array of ints using a custom comparator, but Java's library doesn't provide a sort function for ints with comparators (comparators can be used only with objects). Is there any easy way to do this?



java

sorting



Share Edit Follow Flag

asked Sep 13, 2010 at 9:25



Alexandru

24.3k



17



68



78



do you just want to sort the array in descending order or do you want to perform something more complicated? – Roman Sep 13, 2010 at 12:29

1



Something more complicated. I want to sort the int using absolute value as a key. – Alexandru Sep 13, 2010 at 13:36

[Add a comment](#)

[Start a bounty](#)

11 Answers

Sorted by: Highest score (default) ▾



If you can't change the type of your input array the following will work:

69



```
final int[] data = new int[] { 5, 4, 2, 1, 3 };
final Integer[] sorted = ArrayUtils.toObject(data);
Arrays.sort(sorted, new Comparator<Integer>() {
    public int compare(Integer o1, Integer o2) {
        // Intentional: Reverse order for this demo
        return o2.compareTo(o1);
    }
});
System.arraycopy(ArrayUtils.toPrimitive(sorted), 0, data, 0, sorted.length);
```

This uses [ArrayUtils](#) from the commons-lang project to easily convert between `int[]` and `Integer[]`, creates a copy of the array, does the sort, and then copies the sorted data over the original.

Share Edit Follow Flag

edited Mar 18, 2017 at 12:30



kevinarpe

19.6k



25



122



150

answered Sep 13, 2010 at 10:21



Jon Freedman

9,389



4



42



56

Figure B.2: Stack Overflow Question [#3699141](#).

David | nextsrv.net 05/27/2022
Good evening, I've got this nice and easy Proxmox Request with password and username as Paramters in Postman. Is there any way to recreate this exact the HttpURLConnection but I found no way to do it and possible solutions just did not work.

POST https://[redacted] 30000/api2/jsonocsws/ticket

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
username	[redacted]
password	[redacted]
Any	Value

@David | nextsrv.net Good evening, I've got this nice and easy Proxmox Request with password and username as Paramters in Postman. Is there any way to recreate this ex

JavaBot BOT 05/27/2022
This channel has been reserved for your question.
Please use /unreserve when you're finished.

David | nextsrv.net 05/27/2022
Best solution I could do till now is to add them as Parameters in the URL but that's not save and would basically mean that I'm sending my passwords acro

Kemikals 05/27/2022
java has an http client, or multiple http client libraries that you can use to send parameters in the the request body/or as a form

@Kemikals java has an http client, or multiple http client libraries that you can use to send parameters in the the request body/or as a form

David | nextsrv.net 05/27/2022
I've tried to it with the HttpsURLConnection but there was litterally now way to add Fields in the Body and when searchin the Internet there was no solutio

Kemikals 05/27/2022
i havn't played with the built in httpclient yet for it, but here's one using apaches httpclient <https://codippa.com/apache-httpclient-post-request-java/> loo

codippa

codippa

Java - Apache HttpClient POST with key-value and json body

May 23, 2022 - Learn how to send HTTP POST request with Apache HttpClient having a body with data in key-value pairs and json string with example.

<https://stackoverflow.com/questions/56728398/java-11-new-http-client-send-post-requests-with-x-www-form-urlencoded-parameter>

Stack Overflow

Java 11: New HTTP client send POST requests with x-www-form-urlencoded...

I'm trying to send a POST request using the new http client api. Is there a built in way to send parameters formatted as x-www-form-urlencoded?

You're viewing older messages

Figure B.3: Discord Conversation #979813430401835078.

Java 11: New HTTP client send POST requests with x-www-form-urlencoded parameters

Asked 3 years, 4 months ago Modified 9 months ago Viewed 15k times



25



I'm trying to send a POST request using the new http client api. Is there a built in way to send parameters formatted as `x-www-form-urlencoded` ?

My current code:

```
HttpRequest request = HttpRequest.newBuilder()
    .uri(URI.create(url))
    .header("Content-Type", "application/x-www-form-urlencoded")
    .POST(BodyPublishers.ofString("a=get_account&account=" + URLEncoder.encode(ac
    .build());
```

What I'm looking is for a better way to pass the parameters. Something like this:

```
Params p=new Params();
p.add("a", "get_account");
p.add("account", account);
```

Do I need to build myself this functionality or is something already built in?

I'm using Java 12.

java

java-http-client

Share Edit Follow Flag

asked Jun 23, 2019 at 23:24



Doua Beri

10.2k ● 18 ● 85 ● 131

- 3 ▲ Frustratingly, you need to do it yourself. There's been [an open bug for it](#) since Java 6. – VGR Jun 24, 2019 at 1:09
- 2 ▲ as described here: [golb.hplar.ch/2019/01/...](http://golb.hplar.ch/2019/01/) you can build your own BodyPublisher which takes a Map interface as input – pero_hero Mar 4, 2020 at 9:39

[Add a comment](#)

[Start a bounty](#)

Figure B.4: Stack Overflow Question [#56728398](#).

Drax 05/05/2022
I am using the json.simple package. Let's assume I have a parsed JSONArray from the file:

```
[
  {
    "Object1": "This",
    "Object2": 1
  }
]
```

I am trying to put the contents into a `HashMap<String, Object>()`. Would anyone happen to know how to iterate over this JSONArray to get the keys and values?

@Drax I am using the json.simple package. Let's assume I have a parsed JSONArray from the file: [{ "Object1": "This", "Object2": 1 }] I am trying to

JavaBot BOT 05/05/2022
This channel has been reserved for your question.
Please use `/unreserve` when you're finished.

Drax 05/05/2022
I found this, but this is exact while I need relative: <https://stackoverflow.com/questions/33595941/how-to-convert-jsonarray-to-hashmap>

Stack Overflow

How to convert JSONArray to HashMap

I've been able to get a jsonarray from a json string, but don't know how to put it in a HashMap with a String that shows the type of cargo and an Integer showing the amount.

The string:

```
"cargo":[...
```

JavaBot BOT 05/05/2022
Channel marked as dormant
It is no longer possible to send messages in this channel until it becomes available again.
If your question was not answered yet, feel free to claim a new available help channel.

May 24, 2022

JavaBot BOT 05/24/2022
This channel is now available!
This channel is no longer reserved. Feel free to ask your question here!
Remember to follow the [# ! | how-to-get-help](#)

泣いオオカミ 05/24/2022
hi my goal is when i do a request to sql, it should return the user and its type if it admin,manager or customer

You're viewing older messages

Figure B.5: Discord Conversation [#971797369874182194](#).

How to convert JsonArray to Hashmap

Asked 7 years ago Modified 7 months ago Viewed 41k times



I've been able to get a jsonArray from a json string, but don't know how to put it in a Hashmap with a String that shows the type of cargo and an Integer showing the amount.

The string:

```
"cargo": [
  {"type": "Coals", "amount": 75309},
  {"type": "Chemicals", "amount": 54454},
  {"type": "Food", "amount": 31659},
  {"type": "Oil", "amount": 18378}
]
```

java arrays json

Share Edit Follow Flag

edited Aug 10, 2021 at 9:39



Sang Nguyen
3 ● 2

asked Nov 8, 2015 at 16:03



Zoef
383 ● 2 ● 6 ● 18



possible this link can help stackoverflow.com/questions/4307118/jsonarray-to-hashmap – Shriram Nov 8, 2015 at 16:05



no sorry, already tried that one, some methods don't seem to work with that example – Zoef Nov 8, 2015 at 16:17



what did you try and what was the result ? – BrunoLevy Nov 8, 2015 at 16:45



I tried the code from the link you gave me. But "JSONObject" and "optJSONObject" couldn't be resolved, maybe because I use javax.json libraries and that guy uses another library? – Zoef Nov 10, 2015 at 9:53

[Add a comment](#)

[Start a bounty](#)

3 Answers

Sorted by: Highest score (default)



Figure B.6: Stack Overflow Question [#33595941](#).

lilkitkat3 04/11/2022
Is there an equivalent of `chdir()` in Java?

@lilkitkat3 Is there an equivalent of `chdir()` in Java?

JavaBot BOT 04/11/2022
🔔 **This channel has been reserved for your question.**
Please use `/unreserve` when you're finished.

@lilkitkat3 Is there an equivalent of `chdir()` in Java?


マエヴェ (彼の友達) (ping on reply) 04/11/2022
No I don't think there is

dan1st | Daniel 04/11/2022
<https://stackoverflow.com/q/43130952/10871900>

Stack Overflow

Why `chdir` does not support in java?

I would like to know the actual reason behind why Java does not provide `chdir` call implementation however it provides native interfaces for the same.



マエヴェ (彼の友達) (ping on reply) 04/11/2022
Also what's the point of such a function

@lilkitkat3 Is there an equivalent of `chdir()` in Java?

Polarian 04/11/2022
use relative path

Polarian 04/11/2022
@lilkitkat3 if you no longer need help press unreserve, you can also press the heart buttons to thank the people who helped you this allows others to use this channel to be helped so if you do not need help do not hog the channel thanks 😊

JavaBot BOT 04/11/2022
🔔 **Channel marked as dormant**
It is no longer possible to send messages in this channel until it becomes available again.
If your question was not answered yet, feel free to claim a new available help channel.

JavaBot BOT 04/11/2022
✅ **This channel is now available!**
This channel is no longer reserved. Feel free to ask your question here!
Remember to follow the [# ! | how-to-get-help](#)

Figure B.7: Discord Conversation #963170643640213576.

Why chdir does not support in java?

Asked 5 years, 7 months ago Modified 5 years, 7 months ago Viewed 1k times



I would like to know the actual reason behind why Java does not provide chdir call implementation however it provides native interfaces for the same.

1



java



Share Edit Follow Flag

edited Mar 31, 2017 at 1:58



John Kugelman

339k ● 67 ● 514 ● 563

asked Mar 31, 2017 at 1:30



MD05

325 ● 3 ● 8

3 ▲ It is unusable in any multi-threaded application. You really don't want to do this. – [user207421](#) Mar 31, 2017 at 1:33



Hi Ejp, Thanks but I need to know the reason. I mean if I do chdir from the running program in java what problems it would arise ? And why native interfaces are being provided ? Is there any specific reason for this case ? – [MD05](#) Mar 31, 2017 at 1:40



If you use a native interface *and* you use Java's security model for file accesses, then you can break the security model - if you chdir while another thread (or close to the time another thread - given memory consistency issues with multiple thread) does the access control check, you may be able to access a file that you should be able to given the active security policy. – [Erwin Bolwidt](#) Mar 31, 2017 at 1:52 ✎



Thanks a lot. That means it contrary to the safety principle provided by the Java. Got it! – [MD05](#) Mar 31, 2017 at 1:56



If one thread calls it with one directory and another thread calls it with another directory, the first thread is now not in the directory it thinks it's in, without notice. – [user207421](#) Mar 31, 2017 at 2:06 ✎

[Add a comment](#)

[Start a bounty](#)

1 Answer

Sorted by: Highest score (default) ▾



The full evaluation can be found in Sun/Oracle's bug database with issue number [JDK-4045688](#) : [Add chdir or equivalent notion of changing working directory](#).

2

Figure B.8: Stack Overflow Question [#43130952](#).

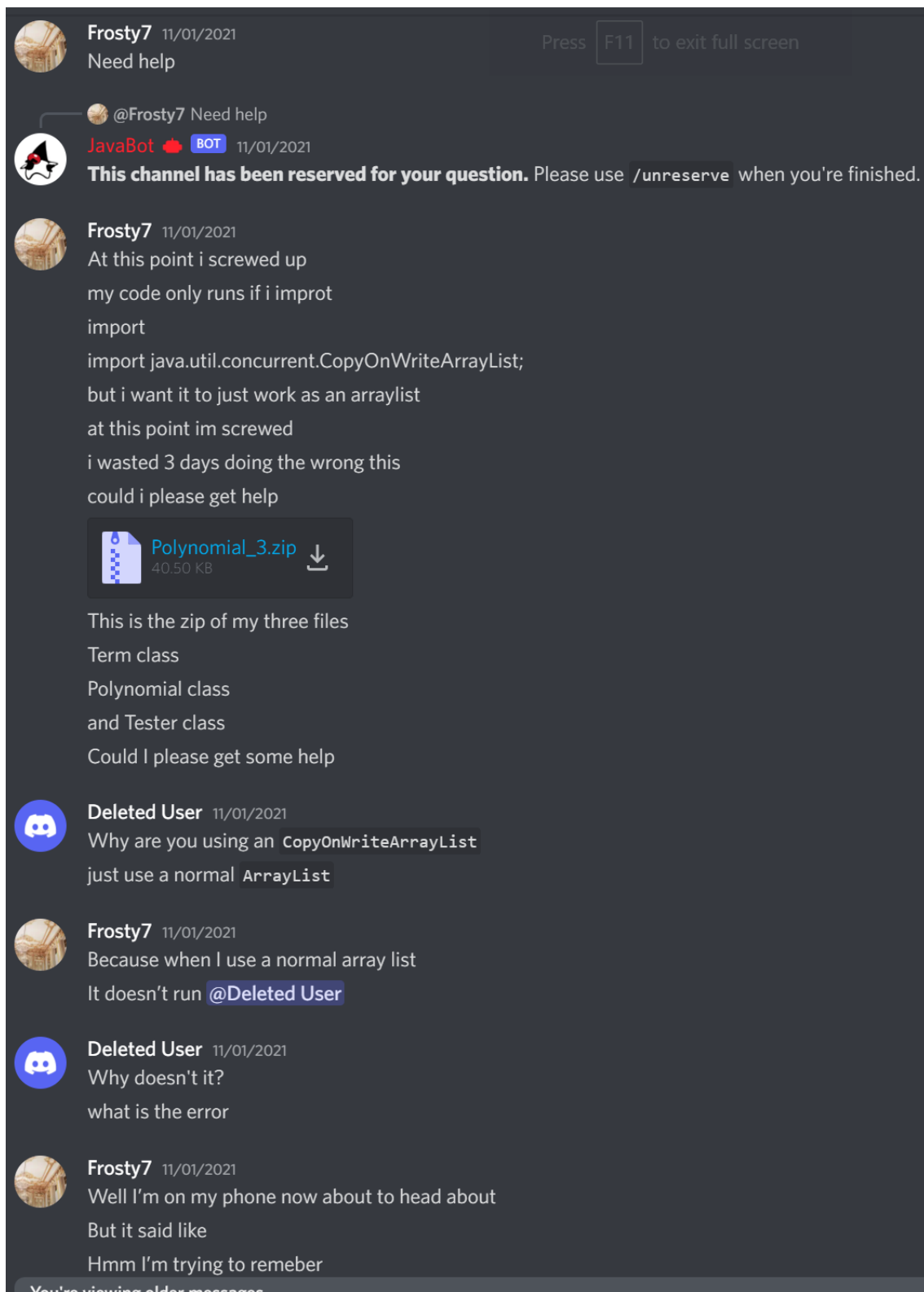


Figure B.9: Discord Conversation #904644087200755723.

Iterating through a Collection, avoiding ConcurrentModificationException when removing objects in a loop

Asked 14 years ago Modified 7 months ago Viewed 512k times

▲ We all know you can't do the following because of `ConcurrentModificationException`:

1285



```
for (Object i : l) {
    if (condition(i)) {
        l.remove(i);
    }
}
```

But this apparently works sometimes, but not always. Here's some specific code:

```
public static void main(String[] args) {
    Collection<Integer> l = new ArrayList<>();

    for (int i = 0; i < 10; ++i) {
        l.add(4);
        l.add(5);
        l.add(6);
    }

    for (int i : l) {
        if (i == 5) {
            l.remove(i);
        }
    }

    System.out.println(l);
}
```

This, of course, results in:

```
Exception in thread "main" java.util.ConcurrentModificationException
```

Even though multiple threads aren't doing it. Anyway.

What's the best solution to this problem? How can I remove an item from the collection in a loop without throwing this exception?

I'm also using an arbitrary `Collection` here, not necessarily an `ArrayList`, so you can't rely on `get`.

java collections iteration

Figure B.10: Stack Overflow Question [#223918](#).

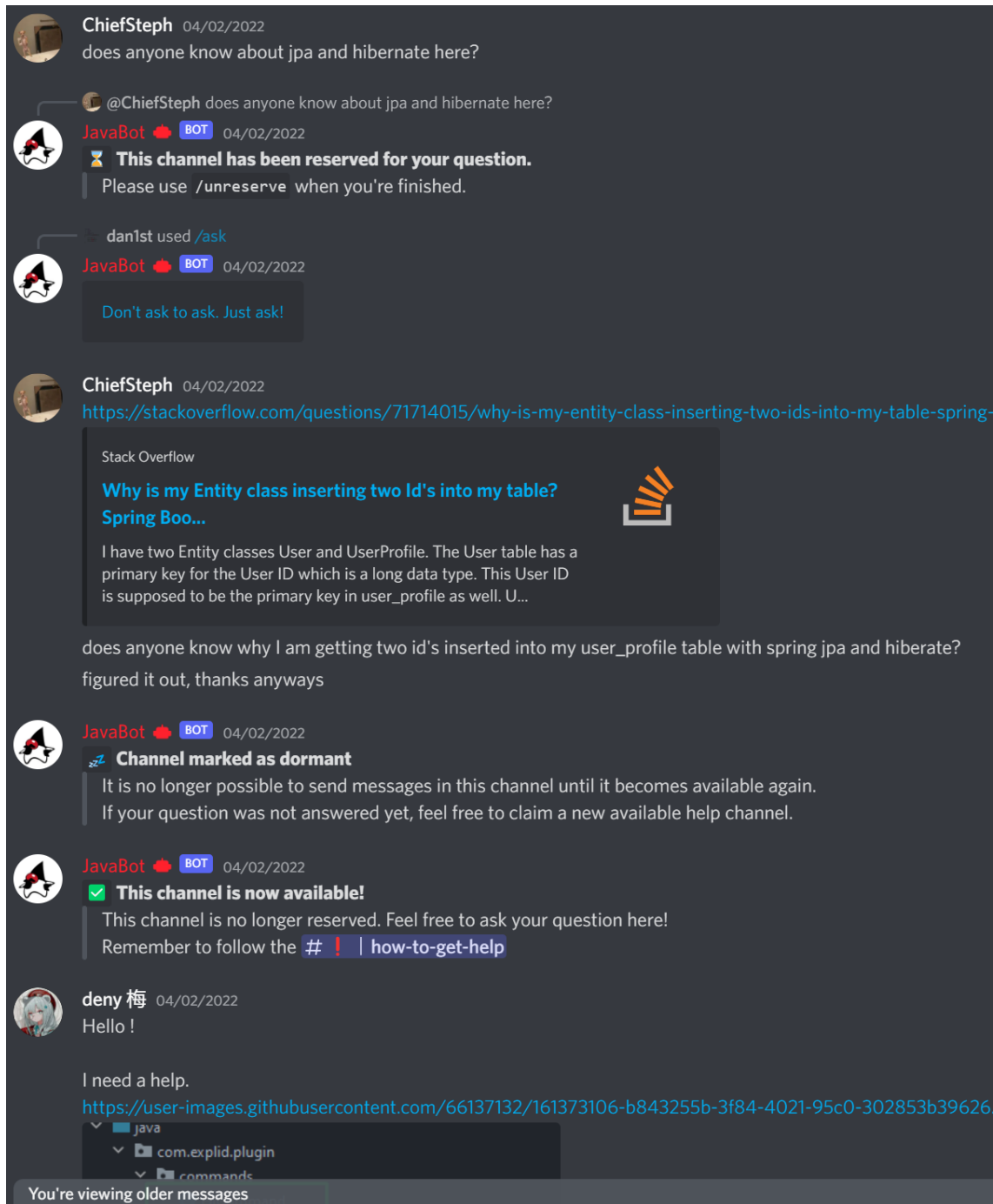


Figure B.11: Discord Conversation #959725068391432212.

Why is my Entity class inserting two Id's into my table? Spring

Asked 7 months ago Modified 7 months ago Viewed 529 times

I have two Entity classes `User` and `UserProfile`. The `User` table has a primary key for the User ID which is a long data type. This User ID is supposed to be the primary key in `user_profile` as well. `User` also has email as a column, which I want to be linked to the `user_profile` also. The issue I am having is that for some reason, a column named `id` is being inserted into my table when I already have the primary key `user_id` set in the `user_profile` table. Does anyone know what I am doing wrong?



User:

```
import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;

@Entity
@Table( name = "users",
        uniqueConstraints = {
            @UniqueConstraint(columnNames = "username"),
        })
@SecondaryTable(name = "user_profile")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column(name = "username")
    @NotBlank
    @Size(max = 20)
    private String username;


    @Column(name = "email", table = "user_profile")
    @NotBlank
    @Size(max = 50)
    @Email
    private String email;
```

Run code snippet

Copy snippet to answer



Expand snippet


Figure B.12: Stack Overflow Question #71714015.



 **Arlind** 04/20/2022
When the if / if else conditions are true why the `String english = scan.nextLine();` and `String sentence = s`
no errors (im new to java)


```
int s = scan.nextInt();



if(s == 1) {
    System.out.println("Enter the code");
    String morse = scan.nextLine();
    String english = null;
    try {
        english = morseTranspiler.morseToLatin(morse);
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("The Code" + english);
    System.out.println("Translation:" + morse);
}
else if(s == 2) {
    System.out.println("Enter a word or sentence");
    String sentence = scan.nextLine();
    String morse = null;
    try {
        morse = morseTranspiler.latinToMorse(sentence);
    } catch (Exception e) {
        e.printStackTrace();
    }
    System.out.println("Sentence/Word: " + sentence);
    System.out.println("Translation to Morse: " + morse);
}
```


 **JavaBot**  04/20/2022
This channel has been reserved for your question.
Please use `/unreserve` when you're finished.


 **@Arlind** When the if / if else conditions are true why the `String english = scan.nextLine();` and `String sentence = s`

 **Fischstaebchen14**  04/20/2022
Whats your expected output and whats the actual output ?
And why should both if and else if both be true ?

 **Arlind** 04/20/2022
the user should enter a morse code and the program will translate it to english words
opposite for the else if condition

 **Fischstaebchen14**  04/20/2022
And the tranlation doesnt work ?

 **That_Guy977** 04/20/2022
`nextInt` doesn't consume the newline, so when you call `nextLine` again it'll just get the empty string (iirc)

 **Arlind** 04/20/2022
it does work but the if conditions just exits when the program is executed


 **That_Guy977** 04/20/2022
<https://stackoverflow.com/questions/13102045/scanner-is-skipping-nextline-after-using-next-or-nextfoo>

Figure B.13: Discord Conversation #966404222058893322.

Scanner is skipping nextLine() after using next() or nextFoo()?

Asked 10 years ago Modified 1 month ago Viewed 769k times

I am using the `Scanner` methods `nextInt()` and `nextLine()` for reading input.

911 It looks like this:



```
System.out.println("Enter numerical value");
int option;
option = input.nextInt(); // Read numerical value from input
System.out.println("Enter 1st string");
String string1 = input.nextLine(); // Read 1st string (this is skipped)
System.out.println("Enter 2nd string");
String string2 = input.nextLine(); // Read 2nd string (this appears right after reading the 1st string)
```

The problem is that after entering the numerical value, the first `input.nextLine()` is skipped and the second `input.nextLine()` is executed, so that my output looks like this:

```
Enter numerical value
3 // This is my input
Enter 1st string // The program is supposed to stop here and wait for my input, but it skips this line
Enter 2nd string // ...and this line is executed and waits for my input
```

I tested my application and it looks like the problem lies in using `input.nextInt()`. If I delete it, then both `string1 = input.nextLine()` and `string2 = input.nextLine()` are executed as I want them to be.

java io java.util.scanner

Share Edit Follow Flag

edited Apr 4, 2018 at 13:31



Lii

11.2k ● 8 ● 60 ● 82

asked Oct 27, 2012 at 16:37



blekione

9,872 ● 3 ● 19 ● 25

10 Or you could be like me and use `BufferedReader` :) I don't care if it's old school, it has always worked and always will work for me. Also, knowledge of `BufferedReader` has application elsewhere. I simply don't like `Scanner`. – [Cruncher](#) Aug 27, 2013 at 19:36

[Add a comment](#)

[Start a bounty](#)

Figure B.14: Stack Overflow Question [#13102045](#).

Why doesn't == work on String? [duplicate]

Asked 9 years, 4 months ago Modified 4 years ago Viewed 31k times

This question already has answers here:

[How do I compare strings in Java?](#) (23 answers)

Closed 9 years ago.



I just started Java programming. I love it so far, but I have been stuck on this problem for a while.

When I run this code, whenever I type in "boy" it will just respond with GIRL:

```
import java.util.Scanner;

public class ifstatementgirlorboy {
    public static void main(String args[]) {
        System.out.println("Are you a boy or a girl?");
        Scanner input = new Scanner(System.in);
        String gender = input.nextLine();

        if(gender=="boy") {
            System.out.println("BOY");
        }
        else {
            System.out.println("GIRL");
        }
    }
}
```

Why?

java

Share Edit Follow Flag

edited Oct 14, 2013 at 5:37



Suresh Atta

119k ● 37 ● 192 ● 301

asked Jul 3, 2013 at 8:30



user2545642

325 ● 1 ● 3 ● 14

▲ use equals instead of == for string in java. Check other questions on this site for more precisions about this. – [benzonico](#) Jul 3, 2013 at 8:31

▲ This has been asked a lot of time in SO. Please take a while to explore before posting [stackoverflow.com/questions/513832/...](https://stackoverflow.com/questions/513832/) – [sanbhat](#) Jul 3, 2013 at 8:34

Figure B.15: Stack Overflow Question #17443201.

Scanner nextLine() NoSuchElementException

Asked 10 years, 2 months ago Modified 10 years, 2 months ago Viewed 10k times

I've been programming a Bukkit plugin for a while now, and this one issue has me stumped. I'm trying to read a line from a file using a Scanner, and add everything on the line before ":" to a HashSet. Every time I try, I get a NoSuchElementException, looking like this:

```
21:01:01 [SEVERE] Could not pass event PlayerLoginEvent to ServerProtect
org.bukkit.event.EventException
    at org.bukkit.plugin.java.JavaPluginLoader$1.execute(JavaPluginLoader.java:341)
    at org.bukkit.plugin.RegisteredListener.callEvent(RegisteredListener.java:62)
    at org.bukkit.plugin.SimplePluginManager.fireEvent(SimplePluginManager.java:477)
    at org.bukkit.plugin.SimplePluginManager.callEvent(SimplePluginManager.java:462)
    at net.minecraft.server.ServerConfigurationManagerAbstract.attemptLogin(ServerConfigurationManagerAbstract.java:273)
    at net.minecraft.server.NetLoginHandler.d(NetLoginHandler.java:112)
    at net.minecraft.server.NetLoginHandler.c(NetLoginHandler.java:41)
    at net.minecraft.server.DedicatedServerConnectionThread.a(DedicatedServerConnectionThread.java:44)
    at net.minecraft.server.DedicatedServerConnection.b(SourceFile:29)
    at net.minecraft.server.MinecraftServer.q(MinecraftServer.java:578)
    at net.minecraft.server.DedicatedServer.q(DedicatedServer.java:213)
    at net.minecraft.server.MinecraftServer.p(MinecraftServer.java:474)
    at net.minecraft.server.MinecraftServer.run(MinecraftServer.java:406)
    at net.minecraft.server.ThreadServerApplication.run(SourceFile:539)
Caused by: java.util.NoSuchElementException: No line found
    at java.util.Scanner.nextLine(Unknown Source)
    at com.gmail.thecotsdragon98.ServerProtect.AltAccounts.CheckForUsedIP(AltAccounts.java:49)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at org.bukkit.plugin.java.JavaPluginLoader$1.execute(JavaPluginLoader.java:339)
    ... 13 more
```

This is the class where I try to do what I explained above.

```
package com.gmail.thecotsdragon98.ServerProtect;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
```

Figure B.16: Stack Overflow Question [#12273794](#).

Why this output when comparing args[0] with a string? [duplic

Asked 10 years, 1 month ago Modified 10 years, 1 month ago Viewed 5k times

This question already has answers here:

Closed 10 years ago.

Possible Duplicate:

[How do I compare strings in Java?](#)

When I run the following code as "java XYZ BOY" the output is "Invalid type", how??

```
public class XYZ
{
    public static void main(String args[])
    {

        if(args[0]=="BOY")
            System.out.println("He is boy");

        else if(args[0]=="GirL")
            System.out.println("She is girl");

        else
            System.out.println("Invalid type");

    }
}
```

However when I use `args[0].equals("BOY")`, it gives the desired output. I want to know why things go wrong here when I don't use `String.equals()`?

java string comparison

Share Edit Follow Flag

edited May 23, 2017 at 12:08



Community Bot

1 • 1

asked Sep 22, 2012 at 14:22



Jignesh

555 • 1 • 8 • 18

2 [How do I compare strings in Java?](#) – Baz Sep 22, 2012 at 14:24

Figure B.17: Stack Overflow Question #12544564.

I'm doing an assignment and I keep getting a NullPointerException Processing, what is this and what can I do to fix it? [duplicate]

Asked 6 years, 8 months ago Modified 6 years, 8 months ago Viewed 40 times

This question already has answers here:

0

[What is a NullPointerException, and how do I fix it?](#) (12 answers)

Closed 6 years ago.



I'm writing this code for an assignment and I would greatly appreciate anyone's help. It's supposed to be a "video game" (it's more like a choose your own adventure game), and I keep getting errors that say "NullPointerException."

On top of that I can't get my images to show up and my video in the code is not playing.

If anyone could possibly give me some pointers, I would sincerely appreciate your help because I honestly don't know what I'm doing wrong.

Thank you so much, here's my code.

```
import processing.video.*;
String PATH = "/Project_2_Scetch/data/video_library/img4.mp4";
Movie mov;

Capture cam;
float xin, yin;
int maxImages = 06;
int imageIndex = 0;
int value = 0;
PImage img;
PImage[] images = new PImage[maxImages];
float positionX = xin;
float positionY = yin;

void setup() {
  size(1000, 650);
  smooth();
  noStroke();
  img = loadImage("img0.jpg");
  rectMode(CENTER);

  if (key == 's'){
    String[] cameras = Capture.list();

    if (cameras.length == 0) {
```

Figure B.18: Stack Overflow Question [#36096097](#).

hitting a URL using java [duplicate]

Asked 12 years ago Modified 12 years ago Viewed 28k times



5



This question already has answers here:

Closed 12 years ago.

Possible Duplicates:

[How to send HTTP request in java?](#)

[How to use java.net.URLConnection to fire and handle HTTP requests?](#)

using java, how do i hit any url?

for instance, opening of <http://www.xyz.com/node1> in a browser will tell xyz.com that node1 is hit. so in this java program (which sends sms text say 'node1' in example above embedded in the url itself to a sms gateway server)

how do i achieve it without opening any browser or using servlet.

java http url

Share Edit Follow Flag

edited May 23, 2017 at 12:08



Community Bot
1 • 1

asked Nov 15, 2010 at 14:34



aksci
140 • 1 • 2 • 10



Exact dupe: stackoverflow.com/questions/1359689/... – Sam152 Nov 15, 2010 at 14:36



Tutorial: [How to use java.net.URLConnection to fire and handle HTTP requests?]
(stackoverflow.com/questions/2793150/...) – BalusC Nov 15, 2010 at 14:38

1 @Sam152: That's a completely different question, dealing with sending an HTTP request. That's different than just loading a URL. – Mark Peters Nov 15, 2010 at 14:43

[Add a comment](#)

2 Answers

Sorted by: Highest score (default)



You can use an [HttpURLConnection](#).

Figure B.19: Stack Overflow Question #4185320.

Send Http Post Message using a stream in Java [duplicate]

Asked 11 years, 2 months ago Modified 11 years, 1 month ago Viewed 5k times



3



This question already has answers here:

Closed 11 years ago.

Possible Duplicate:

[How to use java.net.URLConnection to fire and handle HTTP requests?](#)

I have to send an http post message to a server. This message must have an header and a body and to send it i must use a Stream.

Both the header and the body must be an array of Byte.

How can I do this?

Thank you

java http post stream bytearray

Share Edit Follow Flag

edited May 23, 2017 at 12:29



Community Bot

1 • 1

asked Sep 20, 2011 at 15:01



Marcon

39 • 1 • 2

2 See this [thread](#). In particular, the section titled "**Streaming Mode**". – mrkhrt Sep 20, 2011 at 15:06

[Add a comment](#)

1 Answer

Sorted by: Highest score (default)



1



From the server you can get header information separately from

`HttpServletRequest.getHeaders()`. Data should be read in the form of parts (`multipart/form-data`) using input stream that can be opened from `Request` object. Hope this help.

Ref: <http://www.servlets.com/cos/javadoc/com/oreilly/servlet/MultipartRequest.html>

<http://www.jguru.com/faq/view.jsp?EID=1045507>

Figure B.20: Stack Overflow Question [#7487341](#).

HttpClient with redirect [duplicate]

Asked 10 years ago Modified 10 years ago Viewed 80 times



1



This question already has answers here:

Closed 10 years ago.

Possible Duplicate:

[How to prevent apache http client from following a redirect](#)

I am making a request using httpclient and after receiving the response, I get automatically redirected to another page. How can I stop this from happening?

Here are some more details:

I am trying to get a token from Google using OAuth2.0. I make the request for the token from the server using httpclient, but because of the `redirect_uri` I get somehow redirected. How can I stop this?

java

apache-httpclient-4.x

Share Edit Follow Flag

edited May 23, 2017 at 12:12



Community Bot

1 • 1

asked Oct 24, 2012 at 20:15



Dragos

2,861 • 11 • 37 • 55

1 @TomaszNurkiewicz I do not think it is a duplicate. Maybe I don't understand the other question/answers, but I have tried to implement the code provided there and it did not work. Do you have another solution?
– [Dragos](#) Oct 24, 2012 at 20:35

@Dragos The final answer on the page pointed to by TomaszNurkiewicz gives a working solution.
– [GreyBeardedGeek](#) Oct 25, 2012 at 0:53

[Add a comment](#)

Browse other questions tagged [java](#) [apache-httpclient-4.x](#) or [ask your own question](#).

Figure B.21: Stack Overflow Question [#13056980](#).

Java: What is the purpose of setDoInput in URLConnection [duplicate]

Asked 10 years, 4 months ago Modified 10 years, 4 months ago Viewed 13k times



7



This question already has answers here:

Closed 9 years ago.

Possible Duplicate:

[Java HttpURLConnection](#)

Would this method be used for example if I want to read the response to a POST method call from the server I am connected to?

java http-post urlconnection

Share Edit Follow Flag

edited May 23, 2017 at 12:16



Community Bot

1 • 1

asked Jul 4, 2012 at 11:19



Teerera Marange

1,972 • 4 • 31 • 48

1 `setDoInput(false)` = Ignores the response body, and disallows use of `getResponseCode()` also.
– [Yousha Aleayoub](#) Apr 19, 2016 at 0:50

That's incorrect. `setDoInput(false)` just means you aren't interested in a response body, you still get the headers, and you can still call `getResponseCode()`. `setDoInput(false)` might be valid for a REST API where you expect a 204 response. – [Adrien](#) May 13, 2018 at 12:15

[Add a comment](#)

1 Answer

Sorted by: Highest score (default)



6



Share Edit Follow Flag



I've never encountered a case for setting it false. You always want to read something, at least the response code.

answered Jul 4, 2012 at 11:28





user207421

1

Figure B.22: Stack Overflow Question [#11328043](#).



how to send HTTP request to a servlet [duplicate]

Asked 10 years, 1 month ago Modified 10 years, 1 month ago Viewed 946 times


-2


This question already has answers here:

Closed 10 years ago.

Possible Duplicate:

[How to send HTTP request in java?](#)

I only have one servlet running on the Tomcat server side. Now I want to send a HTTP request to this servlet from a Swing application, and it's not an APPLLET application (because I see some examples sending request from applet). How can I do this?


java


swing


tomcat

servlets

Share Edit Follow Flag


edited May 23, 2017 at 12:27
 Community Bot
1 • 1



asked Oct 15, 2012 at 3:08
 Cacheing
3,341 • 19 • 45 • 65

 *"I see some examples sending request from applet"* It is much the same when done from an applet, command-line application or servlet. Try using the applet based example and get back to us with a specific question if you run into trouble. – [Andrew Thompson](#) Oct 15, 2012 at 3:32

[Add a comment](#)


1 Answer


Sorted by: Highest score (default) 




2


While you can open a direct socket connection and send the raw HTTP headers & content and receive a response back, I would urge you to take a look at [HttpRequestBase](#).

Share Edit Follow Flag

edited Oct 15, 2012 at 3:33
 Andrew Thompson
167k • 40 • 213 • 423

answered Oct 15, 2012 at 3:12
 Pradeep Pati
5,689 • 2 • 29 • 43

[Add a comment](#)

Figure B.23: Stack Overflow Question [#12888780](#).

Bibliography

- [BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate.
<https://arxiv.org/abs/1409.0473>, 2014.
- [CDKP20] Preetha Chatterjee, Kostadin Damevski, Nicholas A Kraft, and Lori Pollock. Software-related Slack chats with disentangled conversations.
<https://dl.acm.org/doi/abs/10.1145/3379597.3387493>, 2020.
- [CDP⁺19] Preetha Chatterjee, Kostadin Damevski, Lori Pollock, Vinay Augustine, and Nicholas A. Kraft. Exploratory study of Slack Q&A chats as a mining source for software engineering tools.
<https://ieeexplore.ieee.org/document/8816749>, 5 2019.
- [CKS⁺17] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data.
<https://arxiv.org/abs/1705.02364>, 2017.
- [DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.
<https://arxiv.org/abs/1810.04805>, 2018.
- [Dri16] Kevin Driscoll. Social media’s dial-up ancestor: The bulletin board system.
<https://spectrum.ieee.org/social-medias-dialup-ancestor-the-bulletin-board-system>, 2016. Accessed: 2022-10-10.
- [Dö18] Matthias Döring. Performance measures for multi-class problems.
<https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems>, Dec 2018.
- [EC08] Micha Elsner and Eugene Charniak. You talking to me? a corpus and algorithm for conversation disentanglement.
<https://aclanthology.org/P08-1095/>, 2008.
- [EC20] Aysu Ezen-Can. A comparison of LSTM and BERT for small corpus.
<http://arxiv.org/abs/2009.05451>, 9 2020.
- [Edu] By: IBM Cloud Education. What is natural language processing?
<https://www.ibm.com/cloud/learn/natural-language-processing>.

- [GBV20] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview.
<https://arxiv.org/abs/2008.05756>, 2020.
- [GFP⁺20] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval.
<https://arxiv.org/abs/1903.06902>, 2020.
- [GM86] G. Guida and G. Mauri. Evaluation of natural language processing systems: Issues and approaches.
<https://ieeexplore.ieee.org/document/1457848>, 1986.
- [Gol16] Yoav Goldberg. A primer on neural network models for natural language processing.
<https://arxiv.org/abs/1510.00726>, 2016.
- [goo] What is natural language processing? Google Cloud.
<https://cloud.google.com/learn/what-is-natural-language-processing>.
- [HO19] Abram Hindle and Curtis Onuczko. Preventing duplicate bug reports by continuously querying bug reports.
<https://dl.acm.org/doi/10.1007/s10664-018-9643-4>, 4 2019.
- [HSY16] Yushi Homma, Stuart Sy, and Christopher Yeh. Detecting duplicate questions with deep learning.
<https://zhiguowang.github.io>, 2016.
- [HVB15] Doris Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. CQADupStack: A benchmark data set for community question-answering research.
<https://doi.acm.org/10.1145/2838931.2838934>, 2015.
- [IBM] IBM Watson Natural language understanding - overview.
<https://www.ibm.com/cloud/watson-natural-language-understanding>.
- [JDJ18] H. Jegou, M. Douze, and J. Johnson. Faiss: A library for efficient similarity search.
<https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/>, Jun 2018.
- [JDJ19] J. Johnson, M. Douze, and Hervé Jégou. Billion-scale similarity search with gpus.
<https://arxiv.org/abs/1702.08734>, 2019.
- [JHC⁺20] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization.
<https://aclanthology.org/2020.acl-main.197/>, 2020.
- [KH21] Dehami Deshan Koswatte and Saman Hettiarachchi. Optimized Duplicate Question Detection in Programming Community Q&A Platforms using Semantic Hashing.
<https://ieeexplore.ieee.org/document/9606030>, 8 2021.
- [KS21] May 13, Chrissy Kidd, and Bharat Saxena. NLP vs NLU: What’s the difference?
<https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/>, May 2021.

- [LLZY21] Zhifang Liao, Wenlong Li, Yan Zhang, and Song Yu. Detecting duplicate questions in stack overflow via semantic and relevance approaches.
<https://ieeexplore.ieee.org/document/9712015>, 2021.
- [LMP⁺19] Stefan Larson, Anish Mahendran, Joseph Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan Kummerfeld, Kevin Leach, Michael Laurenzano, Lingjia Tang, et al. An evaluation dataset for intent classification and out-of-scope prediction.
<https://arxiv.org/abs/1909.02027>, 2019.
- [Mac14] Bill MacCartney. Understanding natural language understanding.
<https://nlp.stanford.edu/wcmac/papers/20140716-UNLU.pdf>, 2014.
- [MC17] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval.
<https://arxiv.org/abs/1705.01509>, 2017.
- [McH12] Mary L McHugh. Interrater reliability: the kappa statistic.
<https://pubmed.ncbi.nlm.nih.gov/23092060/>, 2012.
- [MFdlC⁺20] Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. MUSS: Multilingual Unsupervised Sentence Simplification by Mining Paraphrases.
<https://arxiv.org/abs/2005.00352>, 2020.
- [MML⁺15] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching.
<https://arxiv.org/abs/1512.08422>, 2015.
- [MS08] Christopher D. Manning and Hinrich Schütze. Foundations of statistical natural language processing.
<https://mitpress.mit.edu/9780262133609/foundations-of-statistical-natural-language-processing/>, 2008.
- [MTMR22] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive Text Embedding Benchmark.
<https://arxiv.org/abs/2210.07316>, 2022.
- [MYZ13] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations.
<https://aclanthology.org/N13-1090/>, 2013.
- [PLY05] Kwangho Park, Ying-Cheng Lai, and Nong Ye. Self-organized scale-free networks.
<https://journals.aps.org/pre/abstract/10.1103/PhysRevE.72.026131>, 2005.
- [PS19] Nina Poerner and Hinrich Schütze. Multi-view domain adapted sentence embeddings for low-resource unsupervised duplicate question detection.
<https://aclanthology.org/D19-1173/>, 2019.
- [PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation.
<https://nlp.stanford.edu/pubs/glove.pdf>, 2014.
- [PWQ⁺21] Jiayan Pei, Yimin Wu, Zishan Qin, Yao Cong, and Jingtao Guan. Attention-based model for predicting question relatedness on stack overflow.
<http://arxiv.org/abs/2103.10763>, 3 2021.
- [Rei] Nils Reimers. SBERT docs: Semantic search.
<https://www.sbert.net/examples/applications/semantic-search/README.html>.

- [RG19] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks.
<https://aclanthology.org/D19-1410.pdf>, 11 2019.
- [RMG19] Andreas Rücklé, Nafise Sadat Moosavi, and Iryna Gurevych. Neural duplicate question detection without labeled training data.
<https://www.aclweb.org/anthology/D19-1171>, 2019.
- [RSM⁺17] Joao Rodrigues, Chakaveh Saedi, Vladislav Maraev, Joao Silva, and António Branco. Ways of asking and replying in duplicate question detection.
<https://aclanthology.org/S17-1030/>, 2017.
- [RVDA15] Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. Multiview Latent Semantic Analysis (LSA): Representation learning via generalized CCA.
<https://aclanthology.org/N15-1058/>, 2015.
- [SCA] NLP vs. NLU: From understanding a language to its processing.
<https://www.scalenut.com/blogs/nlp-vs-nlu/>.
- [SKV⁺22] Keerthana Muthu Subash, Lakshmi Prasanna Kumar, Sri Lakshmi Vadlamani, Preetha Chatterjee, and Olga Baysal. DISCO: A Dataset of Discord Chat Conversations for Software Engineering Research.
<https://ieeexplore.ieee.org/document/9796319>, 2022.
- [SLL⁺18] Saksham Sachdev, Hongyu Li, Sifei Luan, Seohyun Kim, Koushik Sen, and Satish Chandra. Retrieval on source code: a neural code search.
<https://dl.acm.org/doi/10.1145/3211346.3211353>, 2018.
- [SLM⁺18] Darsh J Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. Adversarial domain adaptation for duplicate question detection.
<http://arxiv.org/abs/1809.02255>, 9 2018.
- [SLS18] Prathusha K Sarma, Yingyu Liang, and William A Sethares. Domain adapted word embeddings for improved sentiment classification.
<https://aclanthology.org/W18-3407/>, 2018.
- [SPM18] Rodrigo F.G. Silva, Klérison Paixão, and Marcelo De Almeida Maia. Duplicate question detection in stack overflow: A reproducibility study.
<https://ieeexplore.ieee.org/document/8330262>, 4 2018.
- [SRM⁺18] Joao Silva, Joao Rodrigues, Vladislav Maraev, Chakaveh Saedi, and António Branco. A 20% Jump in Duplicate Question Detection Accuracy? Replicating IBM team’s experiment and finding problems in its data preparation.
http://lrec-conf.org/workshops/lrec2018/W25/pdf/7_W25.pdf, 2018.
- [SXL⁺19] Amirreza Shirani, Bowen Xu, David Lo, Tamar Solorio, and Amin Alipour. Question relatedness on stack overflow: The task, dataset, and corpus-inspired models.
<http://arxiv.org/abs/1905.01966>, 5 2019.
- [TRR⁺21] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Benchmarking Information Retrieval (BEIR): A heterogenous benchmark for zero-shot evaluation of information retrieval models.
<https://arxiv.org/abs/2104.08663>, 2021.

- [TSHW17] Dan Tito Svenstrup, Jonas Hansen, and Ole Winther. Hash embeddings for efficient word representations.
<https://arxiv.org/abs/1709.03933>, 2017.
- [TUR50] A. M. TURING. I.—COMPUTING MACHINERY AND INTELLIGENCE.
<https://doi.org/10.1093/mind/LIX.236.433>, 10 1950.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need.
<https://arxiv.org/abs/1706.03762>, 2017.
- [WK20] B. Wang and C. . J. Kuo. SBERT-WK: A sentence embedding method by dissecting BERT-based word models.
<https://arxiv.org/abs/2002.06652>, 2020.
- [WZJ19] Liting Wang, Li Zhang, and Jing Jiang. Detecting duplicate questions in stack overflow via deep learning approaches.
<https://ieeexplore.ieee.org/document/8945690>, 12 2019.
- [WZJ20] Liting Wang, Li Zhang, and Jing Jiang. Duplicate question detection with deep learning in stack overflow.
<https://ieeexplore.ieee.org/document/8964380>, 2020.
- [XY20] Zhuojia Xu and Hua Yuan. Forum duplicate question detection by domain adaptive semantic matching.
<https://ieeexplore.ieee.org/document/9043551>, 2020.
- [XYX⁺16] Bowen Xu, Deheng Ye, Zhenchang Xing, Xin Xia, Guibin Chen, and Shanping Li. Predicting semantically linkable knowledge in developer online forums via convolutional neural network.
<https://ieeexplore.ieee.org/document/7582745>, 8 2016.
- [YDC⁺18] Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. Learning to mine aligned code and natural language pairs from stack overflow.
<https://dl.acm.org/doi/10.1145/3196398.3196408>, 2018.
- [YYM15] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering.
<https://aclanthology.org/D15-1237/>, 2015.
- [YZG⁺19] Runqi Yang, Jianhai Zhang, Xing Gao, Feng Ji, and Haiqing Chen. Simple and effective text matching with richer alignment features.
<https://arxiv.org/abs/1908.00300>, 2019.
- [ZLXS15] Yun Zhang, David Lo, Xin Xia, and Jian-Ling Sun. Multi-factor duplicate question detection in stack overflow.
<https://link.springer.com/article/10.1007/s11390-015-1576-4>, 2015.
- [ZRB⁺16] Ye Zhang, Md Mustafizur Rahman, Alex Braylan, et al. Neural information retrieval: A literature review.
<https://arxiv.org/abs/1611.06792>, 2016.
- [ZSL⁺18] Wei Emma Zhang, Quan Z. Sheng, Jey Han Lau, Ermyas Abebe, and Wenjie Ruan. Duplicate detection in programming question answering communities.
<https://dl.acm.org/doi/10.1145/3169795>, 4 2018.

- [ZSLA17] Wei Emma Zhang, Quan Z. Sheng, Jey Han Lau, and Ermyas Abebe. Detecting duplicate posts in programming Q&A communities via latent semantics and association rules.
<https://dl.acm.org/doi/10.1145/3038912.3052701>, 2017.