УДК: 004.852:005.94
DOI: 10.31732/2663-2209-2024-74-178-184

IMPROVING KNOWLEDGE MANAGEMENT IN AN ORGANIZATION WITH NLP MODELS

Oleksandr Bogolii[1]
[1]PhD Student, "KROK" University, Kyiv, Ukraine, email: boholiiom@krok.edu.ua, ORCID: https://orcid.org/0000-0003-0253-667X

ВДОСКОНАЛЕННЯ УПРАВЛІННЯ ЗНАННЯМИ В ОРГАНІЗАЦІЇ ЗА ДОПОМОГОЮ NLP МОДЕЛЕЙ

Олександр Боголій[1]
[1]Аспірант, ВНЗ «Університет економіки та права «КРОК», м. Київ, Україна, e-mail: boholiiom@krok.edu.ua, ORCID: https://orcid.org/0000-0003-0253-667X

Abstract: *In recent years, it has become prevalent for software development to be held from different places in different time zones. In such a setup, to manage projects efficiently it is crucial to pay special attention to knowledge management inside the organization. The purpose of the article is to provide suggestions for improving knowledge management in the organization using NLP models. In this paper, we use a hybrid approach to build a chatbot integrated into a company's messaging system, so employees can use it almost the same way as if they ask a question to their colleagues. Questions are processed by the proposed framework, and the answer is posted next to the message of the question. While companies often have a lot of documentation in different kinds of wikis and documents, it may be challenging to find an answer to a question on time. Often, employees have to wait hours to get help from colleagues in other time zones. Having an intelligent chatbot inside an organization capable of answering questions using internal docs as a source could be beneficial. In this paper, we proposed an approach to building an intelligent chatbot, utilizing pre-build NLP models for efficient retrieval of documents, similar to the employee's query, and further extracting answers from these documents. We combine the Sentence BERT document retrieval model with the RoBERTa-based answer extraction model. Apache Beam project wiki was used as an example of the company's internal wiki. The proposed approach was evaluated on a test dataset containing 63 question-answer pairs in SQuAD format, and showed good performance, with F1 and Exact Match scores being ~0.73 and ~0.53 respectively. Thus, enterprises can use the same technique to build internal chatbots to facilitate knowledge management.*

Keywords: *Chatbots, NLP models, Sentence BERT, RoBERTa, Knowledge Management*
Formulas: 0, fig.: 1, tabl.: 2, bibl.: 8

Анотація. *Останнім часом розробка програмного забезпечення дедалі частіше відбувається за участі команд, розподілених по різних місцях та часових поясах. За таких умов, для ефективного управління проектами вкрай важливо приділяти особливу увагу управлінню знаннями всередині організації. Метою статті є надання пропозицій щодо вдосконалення управління знаннями в організації за допомогою NLP моделей. У цій статті ми використовуємо гібридний підхід для створення чат-бота, інтегрованого в систему обміну повідомленнями компанії, щоб співробітники могли використовувати його майже так само, як якщо б вони ставили запитання своїм колегам. Питання опрацьовуються запропонованим фреймворком, а відповідь розміщується поруч із повідомленням запитання. Незважаючи на те, що компанії зазвичай мають досить багато документації в різноманітних wiki-сторінках та документах, знайти вчасно відповідь на запитання буває досить складно. Часто співробітникам доводиться годинами чекати, щоб отримати допомогу від колег, що працюють в інших часових зонах. Для організації може бути корисним використання інтелектуального чат-бота, здатного відповідати на запитання, використовуючи внутрішню документацію як джерело. У цій статті ми пропонуємо підхід до створення чат-бота, що використовує попередньо створені NLP моделі для пошуку документів, релевантних запиту співробітника, та подальшого вилучення відповідей із цих документів. Запропоновано поєднувати модель пошуку документів Sentence BERT із моделлю вилучення відповідей на основі моделі RoBERTa. Документація проекту Apache Beam була використана як приклад внутрішньої документації компанії. Запропонований підхід було оцінено на тестовому наборі даних, що містив 63 пари запитань-відповідей у форматі SQuAD, і показав хорошу продуктивність, з оцінками F1 та Exact Match ~ 0,73 та ~ 0,53 відповідно. Таким чином, підприємства можуть використовувати схожий підхід для створення внутрішніх чат-ботів для полегшення управління знаннями.*

Ключові слова: *Чат-бот, NLP моделі, Sentence BERT, RoBERTa, Управління Знаннями*
Формул: 0, рис.: 1, табл.: 2, бібл.: 8

Introduction. Nowadays, the software is developed in a multi-site, multicultural, globally d-istributed environment. Along with the benefits obtained through globally distributed development, there are many challenges faced by various companies as well. The main challenges for efficient project management in a remote distributed context are lack of communication, coordination, and control, complicated knowledge sharing, and lack of knowledge about Agile practices among team members (Bogolii, 2023).

Companies use different strategies and techniques to mitigate certain challenges. Still, the efficiency of these strategies may vary in different teams and organizations and may not always be possible to apply. It is especially complicated to tackle communication, collaboration, and knowledge-sharing issues in a geographically distributed team where members work in different time zones.

With development held from various places in different timezones, it can be strongly advantageous to have employees unblocked with answering on time different sorts of questions that they may have. Intelligent chatbots that use NLP models can be a valuable part of this support process. Using chatbots to answer common questions and queries could allow team members in different time zones to find answers to their questions more easily and not be blocked by a lack of information. On the other hand, it reduces the number of questions to be checked and replied to, unloading the staff from too many interruptions and giving the possibility to answer remaining requests better.

While chatbots are being widely used as an essential element of the customer management process, no case studies are available that describe the experience of using chatbots internally inside organizations to facilitate knowledge management and mitigate communication challenges.

Besides the presence of well-known platforms such as Dialogflow from Google, Microsoft Bot Framework, IBM Watson, and others, organizations often do not want to expose internal documentation to these external providers. That is why building in-house solutions is often considered the only possible solution.

This paper will examine an approach to knowledge management inside the organization that uses natural language processing models to build a chatbot, integrated into an existing communication tool, to help employees find the answers to their questions. We will examine the accuracy of answers of the proposed system on a test dataset.

Literature review. Currently, there exist different types of chatbots. According to the knowledge domain, chatbots can be classified into two types: open-domain and closed-domain. Open-domain chatbots can chat about any topic, while closed-domain chatbots can only chat about a specific topic.

Also, chatbots differ in the learning methods they use and how the answers are provided. As for learning methods, chatbots may be rule-based, retrieval-based, or machine learning-based. The provided answers may be extracted or generated from documents stored in the knowledge base.

The traditional Rule-based chatbots operate on a set of predefined rules and patterns to generate responses. The chatbot matches user input against these rules and provides the appropriate response. However, these chatbots struggle to understand context, handle ambiguous queries, or adapt to new scenarios without manual rule updates.

Retrieval-based chatbots are another type of chatbot that relies on predefined responses, but they differ from rule-based chatbots in how they select the appropriate response. Instead of relying on explicit rules, retrieval-based chatbots use a retrieval mechanism to find the most suitable response from a predefined set of responses. Retrieval-based chatbots are effective for handling specific domains or FAQs where the training data covers a wide range of possible user inputs and corresponding responses. They can provide accurate responses based on similar inputs they have seen during training. However, retrieval-based models may struggle to handle out-of-domain queries or generate creative and contextually rich responses.

Recently, machine learning, particularly neural approaches to building chatbots, gained popularity. Neural approaches typically employ the transformers architecture (Vaswani et al., 2017), which uses two recurrent neural networks (RNNs) called Encoder and Decoder. The encoder encodes the input sentences into a semantic representation by consuming the words from left to right, one by one. Then, the decoder decodes this fixed-length representation to generate the target sequence. This particular family of ML approaches is called sequence-to-sequence (seq2seq). Seq2seq models are often used to build generative chatbots.

Neural approaches can also be used to build extractive chatbots. Neural question-answering (QA) models are trained on question-answer pairs and can be used for extractive chatbots. Based on the question, these models learn to identify the answer span within a given context. Typically, such models employ only Encoder from the transformers architecture.

Scaling up such language models with billions of parameters gives impressive results for a broad set of NLP tasks. For example, OpenAI researchers trained a GPT-3 model with 175 billion parameters. Often, such models are trained on enormous amounts of data and called Large Language Models. Some examples of large language models include flan-t5-base, flan-paLM, chinchilla, and GPT-3 variants, such as text-davinci-003.

Despite the performance of Large Language Models, their usage may be challenging. Popular models have scaled exponentially concerning the number of parameters and the size of the pre-training data. Unfortunately, existing compressing techniques cannot compress super large models (e.g., GPT-3) to a suitable degree for deployment on a single GPU or terminal device such as a laptop or cell phone.

Thus being said, hybrid approaches can be used to provide decent results using smaller-sized language models (GPT-2, RoBERTa, etc.).

For example, a combination of information retrieval (IR)-based chatbots and neural chatbot technology. It often outperforms IR and neural chatbots used alone. The method uses IR to extract QA pair candidates and then does a close analysis of documents and performs the core task of question answering using the attentive seq2seq model. Information retrieval is usually implemented using a TF-IDF index variant, like BM25 (Robertson et al., 1994).

It is worth mentioning, a popular framework - SentenceBERT (SBERT), proposed by Reimers & Gurevych (2019). SBERT is a modification of the pretrained BERT model (Devlin et al., 2019), that uses a siamese network to derive semantic sentence embeddings that can be used to calculate cosine similarity between sentences, which potentially improves text retrieval by understanding the content of the supplied text. While the words may be different, they may be semantically similar in meaning. For this reason, it often outperforms the TF-IDF index.

These hybrid approaches seem especially promising for this research, as often internal company documentation represents semi-structured content (FAQ, Wiki, etc.) spread among multiple documents, so combing the search of relative documents with efficient tools for extraction of answers looks very advantageous.

Methodology. In this paper, we use a hybrid approach to build a chatbot integrated into a company's messaging system, so employees can use it almost the same way as if they ask a question to their colleagues. Questions are processed by the proposed framework, and the answer is posted next to the message of the question.

Results. The implementation in this study uses Python 3.7 with the PyTorch library and is based on the Haystack framework from Deepset. We constructed a custom pipeline that retrieves the most relevant documents for search queries from the document index and then processes them to return the answer. See Figure 1 for the pipeline structure.

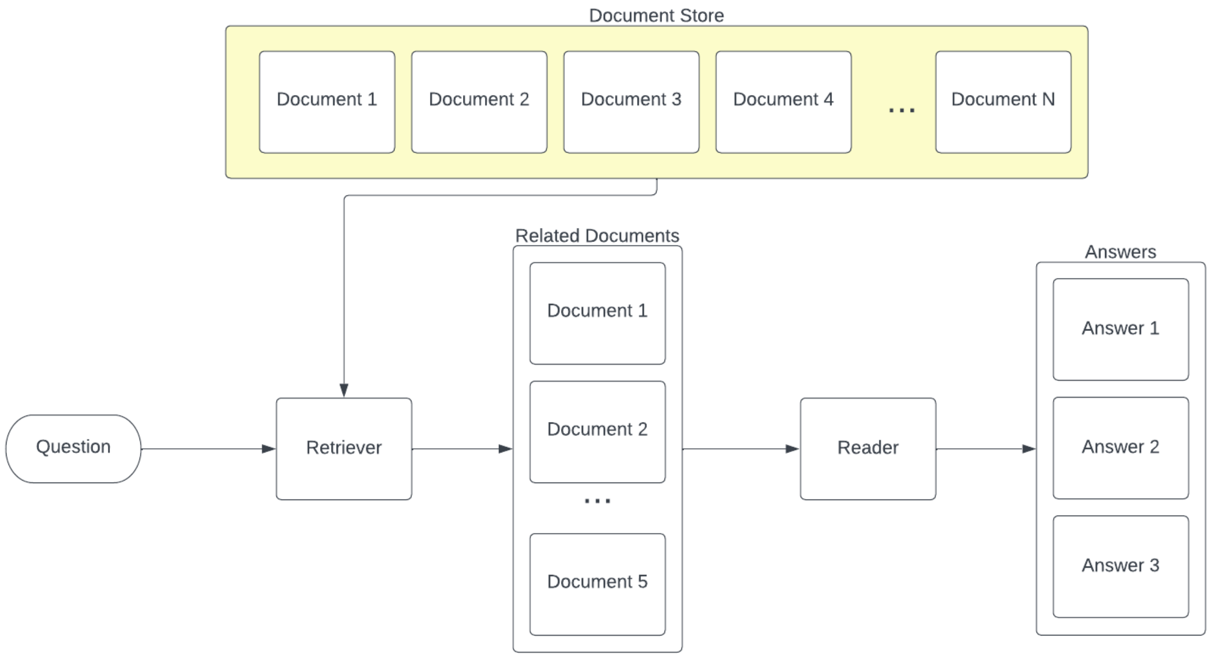

Figure 1. Retriever-Reader pipeline

Source: Figure created by authors

To find the most relevant documents for a given search query, we use the Sentence-BERT method with the multi-qa-mpnet-base-cos-v1 model, which has been trained on 215M (question, answer) pairs from diverse sources and can be used to retrieve documents based on a short query. The documents with the highest similarity are then passed to the next stage.

Next, relevant documents are analyzed to extract the best answer to the query. We used the transformer-based language model deepset/deberta-v3-large-squad2 from HuggingFace. This is the RoBERTa base model (Liu et al., 2019) that was fine-tuned on question-answer pairs, including unanswerable questions. The benchmark (Question Answering on Squad_v2, n.d.) shows that this model has the highest performance for Question Answering, still being able to run on average developer setup.

All setup and testing for the platforms and models outlined here were conducted on a MacBook Pro with an Apple M1 Pro chip and 32 GB RAM.

Dataset. Organizations often use enterprise wiki tools to store internal documentation. For this research, we used open wiki documentation for the Apache Beam project (*Apache Beam*, n.d.).

Documentation is written in English and spread among multiple pages. Pages often use different formatting options to make them more readable and accessible, including headings, tables, code snippets, etc. We wrote a Python script to scrape the pages from Wiki, each page in its file. Next, data from pages were preprocessed, using the following steps:

1. Extract the text from HTML tags
2. Filter only sentences that have Nouns and Verbs.
3. Remove consecutive empty lines and whitespaces at the beginning or end of each line in the text
4. Split into smaller documents of around 100 words each, with a sliding window of 20, to improve the document retrieval performance.

As shown in Figure 1, the proposed pipeline consists of retrieval and reader stages. In the retrieval stage, we look for relevant documents in the document index. In the reader stage, we use the top 5 most relevant documents as a context for answer generation to the query. In the next paragraphs, we will examine the results of the work of each stage and will evaluate their accuracy.

Retrieval. At this stage, Sentence-BERT was used to calculate cosine similarity between the question embedding and each paragraph embedding. The top 5 paragraphs with high scores are retrieved. Table 2 presents an example of paragraph retrieval for a test query: *"How do I get started contributing to Apache Beam?"*.

Table 1. Top 5 similar paragraphs to the example query

Paragraphs	Score
This is the Apache Beam Wiki, with tips, tricks, and detailed guides for contributors. If you want to learn about how to use Apache Beam, start with https://beam.apache.org Browse the page tree in the sidebar for IDE tips, technical documentation, howtos, etc.	0.86603
Frequently asked questions about contributing to Apache Beam. About contributing What should I do after the committer reviews my PR? The reviewer should give the Looks Good to Me (LGMT) in the PR and then you (the author of the pull request) should rebase, squash, or split, the commits so that the history is useful. How do I get started contributing to Apache Beam? See https://beam.apache.org/contribute/ Whom should I ask if I am stuck? How can I opt-in to reviewing pull requests? Go to https://github.com/apache/ repo.	0.84717
NOTE: Beam no longer uses Jira for issue tracking. Instead, please see https://beam.apache.org/contribute/ for information on getting started with GitHub Issues. Jira may still be needed to interact with other Apache projects or Infra, but will not be needed for most contributors. This guide aims to introduce Beam contributors to the basics of using Apache Jira. How to get started on Jira To start collaboration in Apache Jira: Access the Apache Jira dashboard. Read the introduction to see what permissions are available to users and to create an account.	0.84552
Clone the Apache beam project from https://github.com/apache/beam using main branch.	0.83945
If you have any questions, do not hesitate to reach out to us in the Apache Beam mailing list at dev@beam.apache.org (you will need to subscribe first by emailing dev-subscribe@beam.apache.org). Our team of mentors will be happy to answer any of your questions, and we are looking forward to hearing from you	0.83062

Source: Table created by authors

Reader. Most relevant paragraphs, like those presented in Table 1, are then passed to the reader model for answer extraction. Table 2 shows the top 3 pipeline answers and the corresponding context.

Table 2. Top 3 answers to the example query

Answer	Context	Score
Roadmap	A great place to start is the user-facing Roadmap. But there's a lot going on that isn't necessarily listed there.	0.87417
Clone	Clone the Apache beam project from https://github.com/apache/beam using main branch.	0.72821
Fork the github.com/apache/beam repo	To create a new pull request, follow the next steps: Fork the github.com/apache/beam repo Clone your fork, for example $ git clone git@github.	0.72534

Source: Table created by authors

Evaluation of results. To be able to make a statement about the quality of results in a question-answering pipeline, it is important to evaluate it. Two metrics used to evaluate the performance are F1-score and Exact match:

- *Exact match* measures the proportion of cases where the predicted Answer is identical to the correct Answer. For example, for the annotated question-answer pair "What is Chatbot?" + "intelligent conversational agent," even a predicted answer like "conversational intelligent agent" would yield a zero score because it does not match the expected answer 100%.
- The *F1 score* is more forgiving and measures the word overlap between the labeled

and the predicted answer. Whenever the EM is 1, F1 will also be 1.

Annotated datasets are crucial for evaluating the retrieval and the question-answering capabilities of the system. For this reason, we prepared the evaluation dataset with 63 question-answering pairs in SQuAD format (Rajpurkar et al., 2018). Evaluation of the pipeline returned an F1-Score of **0.73204** and an Exact Match Score of **0.53448**.

Discussion. As per our findings, data preprocessing steps can significantly impact the system's performance, and effective handling of data is key to getting the most out of pipeline performance. In particular, splitting large documents into smaller documents of around 100 words was significantly beneficial.

To achieve the highest performance, we tested the pipeline with different sets of models used for retrieval and reader. For the retrieval part, we tested following models used for Sentence Similarity tasks - multi-qa-mpnet-base-dot-v1, all-mpnet-base-v2 and all-distilroberta-v1. multi-qa-mpnet-base-dot-v1 had the highest precision and recall among them, probably because it was fine-tuned to retrieve documents based on short queries, similar to the questions in testset. We assume that even better results could be achieved by using OpenAI or Cohere embeddings, but we limited our tests to only models that have free access. In addition, to tests with different models, we also tried to use the TF-IDF index instead of SBERT for document retrieval. However, after manual verification of retrieval results on test queries, we saw that in plenty of cases where TF-IDF was failing, Sentence-BERT was performing well.

For the reader, at first we experimented with the most popular deepset/roberta-base-squad2 model, which gave decent results, with the F1 score being around 0.63. However, after we replaced it with deepset/deberta-v3-large-squad2 we got around a 15% percent increase. We also experimented with several seq2seq models, like google/flan-t5-large, but during manual evaluation, we found the results were worse. More fine-tuning is needed to efficiently use these models, which could be a subject for subsequent research.

For companies that have documentation wikis with similar characteristics under similar experimental conditions, similar results can be expected.

In addition to reassuring results, worst to mention the limitations of the investigated approach. The proposed chatbot was trained on a closed-domain dataset that contains a limited number of topics. In our case, we tested the dataset covering rephrased questions and answers about the Apache Beam framework. Thus, such a chatbot cannot answer unrelated questions.

Conclusions. In this paper, we discussed the approach to knowledge management inside the organization that helps employees find the answers to their questions on time. We proposed a methodology that uses natural language processing models to build an intelligent chatbot that uses the company's internal wiki to answer employees' queries.

The system could run on a standard developer machine and accurately answer certain questions that would otherwise require colleagues' interruption. We found that the preprocessing of data, data cleanup, and the NLP models chosen significantly impact chatbot performance.

As the performance of the proposed approach is encouraging, further investigation would be required in several distinct areas. Natural language processing is a very fast-evolving field, with new pre-trained models appearing very often, significantly improving the quality of results.

In the future, we would like to incorporate messages from the company’s communication tools into the chatbot’s document store. Frequent questions were already asked at some time in messengers; together with replies, they could be used as an additional content source for the chatbot. Lastly, we want to do an extensive evaluation of the pipeline utilizing semantic similarity of the two answers rather than their lexical overlap (Semantic Answer Similarity).

References:

1. Bogolii, O. (2023). Agile Software Development in a Remotely Working Geographically Distributed Team: A Systematic Review. European Project Management Journal, 13(1), 23–36. https://doi.org/10.56889/idnv2224
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need (arXiv:1706.03762). arXiv. http://arxiv.org/abs/1706.03762
3. Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1994). Okapi at TREC-Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (arXiv:1908.10084). arXiv. http://arxiv.org/abs/1908.10084
4. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (arXiv:1810.04805). arXiv. https://doi.org/10.48550/arXiv.1810.04805
5. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/ARXIV.1907.11692
6. Question Answering on squad_v2. (n.d.). Papers with Code. Retrieved March 8, 2024, from https://paperswithcode.com/sota/question-answering-on-squad-v2
7. Apache Beam. (n.d.). Apache Beam Wiki. Retrieved March 8, 2024, from https://cwiki.apache.org/confluence/display/BEAM/Apache+Beam
8. Rajpurkar, P., Jia, R., & Liang, P. (2018). Know What You Don’t Know: Unanswerable Questions for SQuAD (arXiv:1806.03822). arXiv. http://arxiv.org/abs/1806.03822