PROBEXPERT: An Enhanced Q&A Platform for Reducing Time Spent on Learning and Finding Answers

Dinuka R. Wijendra
Department of Computer Science & Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
dinuka.w@sliit.lk

Anjalie Gamage
Department of Computer Science & Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
anjalie.g@sliit.lk

Kamal Thennakoon
Department of Computer Science & Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it18004564@my.sliit.lk

Ashen Ranasinghe
Department of Computer Science & Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it18011012@my.sliit.lk

Dasun Ekanayake
Department of Computer Science & Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it18013610@my.sliit.lk

Thanura Marapana
Department of Computer Science & Software Engineering
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
it18078992@my.sliit.com

***Abstract*—The World Wide Web contains a wide range of material from a variety of fields. However, when concerns towards the computer science domain, information users find on the internet may not be up-to-date due to the rapid pace of change and having to spend less time on the internet for researching and debugging tasks is an added luxury. Having an expertise level while providing answers through a platform is convenient for users, yet when a user signs into a platform, the user must start from the beginning, regardless of the level of competence in the field. Moreover, not having a proper way to evaluate the existing programming knowledge is another obstacle. To address mentioned complications, researchers of this paper have introduced a new e-learning platform- 'ProbExpert'. The platform has been constructed with machine learning and deep learning approaches such as NLP, keyword extraction, semantic information analysis, cosine similarity, and information summarization. With aforesaid technologies, ProbExpert provides systems in automated answering, optimized answer generation, structured question-based quiz evaluation together with a fully automated portfolio generation with a novel user ranking algorithm based on the bell curve.**

keywords — answer automation, portfolio generation, quiz scoring model, user ranking

I. INTRODUCTION

In recent years, answers on Q&A sites such as Stack Overflow[1] and Quora have become a key source of information for developers to address their technical issues.[2] In most cases, developers submit their inquiries to a search engine, returning a list of relevant postings that may contain answers. Then, developers must review the answers and acquire knowledge to find the solutions to their issues. However, the sheer volume of questions and answers on the mentioned Q&A websites makes it difficult to find information.

As developers face such problems, it may take a long time to get the exact response they require. It is also difficult to locate answers in long articles since there are vast amounts of noise and repeated material online, and the answer they find may only fix part of the problem. 'ProbExpert' is developed to address these concerns, which is more advanced and more equipped to handle challenges than the current Q&A systems available.

ProbExpert consists of some advanced and novel concepts that help users to get more than answers. One of the main functions of ProbExpert is that it will offer developers a fully fledge auto-generated portfolio out of the box. It will track and highlight all the contributions committed to leading Dev communities on the internet. Therefore, users do not need to update them with their recent works. The proposed platform is smart enough to identify and update itself according to the user's recent contributions. Since all the information is precisely optimized for search engines, these portfolios will appear in the search results of major search engines such as Google, Bing, etc. Finally, the developer's Coding, Q&A, and Professional scores will be displayed next to their profile picture.

ProbExpert also includes the functionality of the Question-and-Answer system as a crucial component of the system for sharing information and receiving answers. In addition to receiving answers from people, the system will construct an answer on its own using publicly available data. Stack Overflow, GitHub, Medium, Dev, and YouTube are information resources for the automatic answer. The information gathered will be processed and will go through numerous phases before the response is finalized. To limit the number of duplicate questions, the platform will have a similar question-finding mechanism. As a public dev platform, anyone can answer the questions posted here, but the downside of this is that one question can get several duplicate answers, which inevitably leads to wasting the time of the person, and that is an area of focus with this research, 'reducing the time spent on finding the solution.' Therefore, to reduce this type of redundancy, an optimized answer is displayed. This process is focused on bringing all the alternative solutions into one optimized answer without harming the semantic information.

By considering only the above-stated specifications, it is clear that ProbExpert is not a traditional Q&A website. While it helps achieve the primary cause, another out-of-the-box functionality is that the platform can act as a medium to

improve individuals' programming knowledge. Being such a platform leads to having an enormous dataset of day-to-day programmer's questions and received reliable, optimized answers, which can be easily turned into a platform for theoretical knowledge checking. In this scenario, structured-type online quizzes are considered an effective method[4] and have proven a positive influence on academics[4]. The questions are presented based on the user level, therefore enabling the adaptiveness to everyone for increased engagement and motivation[5]. Presented questions will be generated by the platform's questions. Users can type the answer for each question, and they will receive an unbiased score to assess their current knowledge level.

Overall, this paper covers how ProbExpert helps users develop user profiles, acquire optimized responses, generate quizzes based on relevant areas, and generate automatically summarized answers from the publicly collected information.

II. Literature review

Within the last two decades, plenty of research have been carried out to detect experts in question-answer communities and social coding platforms. In 2010, Wei-Chen et al. proposed a novel hybrid approach [6] that considers user subject relevance, user reputation, and authority of a category in finding experts. It is the first model up to that time that considers both users' reputation and the users' domain knowledge to the target questions. In 2014, Zhao proposed a topic-sensitive probabilistic model [7] for detect experts in the question-answer communities. On the other hand, evaluating developers in social coding platforms such as GitHub and GitLab are useful because they have the real-world software development activities of developers. Also, employers and HR managers often scan candidates' GitHub profiles in the interview process to learn more about their skills and interests. In reference [8], Capiluppi et al. identified four main insights that employers can derive from developers' GitHub profiles by doing an interview-based study with several IT employers. Those are 1. Shared open-source values, 2. Community acceptance of works and contribution quality, 3. Project management skills, and 4. Passion for coding. Since GitHub provides a REST API to its full data set, more researchers tend to use it as a fine resource to evaluate developers and their activities. In 2013, Saya et al. [9] and in 2016, Constantinou et al. [10] studied the characteristics of developer's activities and the expertise level. In 2019, Eduardo et al. proposed an unsupervised machine learning approach to identify experts in software libraries and frameworks among GitHub users [11]. Nevertheless, these studies are platform-dependent, and they are solely focused on identifying the experts. However, previous works did not classify users according to their skill level (e.g., beginner, intermediate and advanced).

Different technologies, such as keyword extraction, text embedding, similarity analysis, and web scraping, generate automated answers and similar question findings. Numerous studies on the mentioned technologies have been conducted in recent years. Almost all of the research has been done on individual technology. Because the ProbExpert question and answering platform uses different technologies, this paper contains studies on the bindings of those specific studies.

Automatic text summarization is a complex subject that is of increasing importance these days. When an input is applied, an automatically summarized result is produced. Although research on the Automated Text Summary began in IBM Research Laboratories in the 1950s,[12] the area of Text Summarization has grown exponentially in recent years due to the internet. Because there are ample amounts of material on the internet, manually summarizing large text pages is quite challenging. As a result, it is critical to sift through many available records to find the proper document. The goal of the text review is to condense the source text into a condensed version that keeps the material's content and overall meaning[13]. To make it easier for users to locate the solutions, the platform is equipped with finding a summarized answer utilizing the overall answers to the ProbExpert platform. A great deal of research has been done on answer summarizing[12]–[18]. A summary is a text derived from one or more texts that contain an approximate amount of the information contained in the original text but not more than half of the original text[13]. According to another guide, a text summary is a process of distilling the most relevant details from a source to create abbreviated versions for a particular customer and objective [12].

With modern-day knowledge shifting towards the internet, all learners can access vast information with several clicks. However, there are times that they cannot rely on the internet alone because, as learners, all the essential facts should be stored in memory. The best way to retain information is by taking quizzes in spaced repetition[19]. According to research[20], the quizzes method is more effective. According to the paper, reading information and then consolidating and testing the quiz form's knowledge helps retain the information[20]. Also, regarding a study based on students which spans five years, the conclusion was the online quizzes are a proven positive influence on students' academics[4]. With the above facts in mind, our platform ProbExpert has implemented a quiz system utilizing the platform's user questions and brilliant answers such questions received. Individuals will be exposed to everyday programming problems, preferably according to the user's knowledge level, which will be helpful in upcoming interviews or exams, and an unbiased score to evaluate the user's knowledge over their desired study area.

III. Methodology

ProbExpert platform is mainly composed of 4 components. In the upcoming sections above technologies have been discussed thoroughly.

A. Data Gathering

a) Developers' activity extraction: Three different scrapping tools have been developed from the ground up to collect the developers' data from GitHub, Stack Overflow, and LinkedIn. Since Stack Overflow and LinkedIn do not provide a public API to gather their users' data, BeautifulSoup4 and Selenium were used to build these two scrappers. However, unlike others, GitHub provides a GraphQL API to query all of its public data. Since it has a strict rating limit (5000 requests for an hour), and these response query results not changing frequently, A tool called

Git-Data-Miner (GDM) has been developed from scratch to retrive users' data and to stop duplicate requests within 24hrs in order to keep the request count below the github rate limit.

b) Stackoverflow answer scraping: for StackOverflow answer scraping, a custom model has been built with Beautifulsoup4 and Lxml. The model will first try to scrape a verified answer if there is one. If the model finds a verified answer, it will process and filter relevant content out. If the model cannot find a verified answer, it will search for the most voted answer of the StackOverflow question. Then it will perform the filtering process. Another application of mentioned StackOverflow answer scraping is applied in quiz generation. If users' desired questions are not on our platform, they can search and obtain from StackOverflow.

c) Other data related to automated answers: Aside from StackOverflow response data, automated answers include information from YouTube, GitHub, Medium, and Dev blogs. A custom Google search will be performed first to gather up-to-date information to acquire data from the above resources. The scraped search results will then be filtered to locate relevant links to the above resources. Finally, the filtered links will be used to collect data from other sources. As an added step to check and maximize relevancy, information from all resources will be processed through numerous phases explained later in this paper.

B. Data Processing

A large amount of raw data was used to generate information in ProbExpert. As the first task, the data must be processed by removing unnecessary data and converting the data into a numerical representation. There are three steps in the data processing.

a) Tokenization: In this process, the text is first tokenized into small individual tokens such as words, punctuation. This process is done by the implementation of rules specific to each language. Based on the specified pattern, the strings are broken into tokens using regular expressions, The patterns used in this work remove the punctuation in the

b) Stop word removal: The stopwords are a group of often used words in the language. Like in English, having several stop words such as "the," "a," "is," "are,". The perception of using these kinds of stopwords is that removing common informative words from the text could lead to focusing more on the crucial words (technology-related words in this paper).

c) Text Embedding: The text form of a sentence cannot be utilized in natural language processing to verify the similarity of a sentence to another sentence (question, according to this study). Sentences must be translated into numerical form. Text data is converted into numerical vector representation to the algorithm to generalize the data and perform further steps.

C. Keyword Extraction

Keyword Extraction (KE) is the automated extraction of single or multiple-token phrases from a textual document that best expresses all critical aspects of its content and can be seen as the automated generation of a short document summary[21]. In the ProbExpert platform, those extracted keywords will be used to calculate the relevancy of the information.

In this study, the Embedding-based keyword extraction method has been used to extract keywords. This exploits document embeddings [*B*] and cosine similarity [*D*] to identify candidate keywords. First, a document embedding is computed, then word n-grams of different sizes are generated, which are subsequently ranked along with their similarity to the embedding of the document. We can effectively achieve this task using pre-trained BERT models' transformer-based sentence embeddings[22][23]. KeyBERT is a minimal and easy-to-use keyword extraction technique that leverages BERT embeddings made by Maarten Grootendorst[21][24]. Once the desired question/answer is applied to KeyBERT, the following operations will be carried out to extract keywords.

- Extraction of candidate Keywords.
- Convert the documents/texts to numerical data. [*B*]
- Calculate similarity. [*D*]
- Diversification using Max Sum Similarity and Maximum Marginal Relevance algorithms.

D. Similarity Checking With Cosine Similarity

To find similar questions, the similarity between answers, and the relevancy of the information on this platform, the similarity value between information needed to be calculated. This has been accomplished through the usage of cosine similarity[25].

a) Similar question finding: Cosine similarity was used to calculate the similarity between two questions (vectorized values). The user's question will be embedded first as described in [*a*], followed by the question list. Then, for each question, cosine similarity will be determined against the user's question. Questions with more than 80% cosine similarity will be classified as similar in the last phase.

b) The similarity between answers: Cosine similarity is used to calculate the similarity between user answers against the platform's verified answer. We used the BERT sentence transformers model to map sentences and paragraphs to a 768-dimensional dense vector space, enabling tasks like clustering and semantic search. After tokenizing [*B*], embeddings, and performing mean pooling to take attention mask into account for correct averaging, applying the cosine similarity between the two vectors was done, which generated a similarity score between the above-mentioned two answers.

A text document is represented as a vector of terms in this metric. The similarity between two documents can be calculated using this model by calculating the cosine value between the term vectors of the two documents[26]. Given two N dimension vectors $\vec{v}$ and $\vec{w}$ the cosine similarity between them can be calculated as follows (1):

$$Cosine(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\ \sqrt{\sum_{i=1}^{N} w_i^2}} \quad (1)$$

E. Generating the automated answer

Material scraped from web resources will be preprocessed before being utilized to calculate the relevancy to the question using cosine similarity[26]. If the scraped information from a resource does not have a similarity value greater than 80%, it is considered irrelevant to the user's question. Information with a more than 80% similarity value will be kept in a document database with the question id as the reference to the user's question. Keyword extraction and weighted keyword analysis will be used to evaluate resources that do not provide text information to generate the similarity value, such as github and YouTube resources. If the Youtube resource contain a description, it will go through the same process as text resource. Answer will contain one Stackoverflow answer, two YouTube and github resources and maximum of ten blog posts.After the resources have been reviewed. If there are any further resources with a high similarity value, will be provided as links within the answer.

F. Summarization

The Q&A portion of ProbExpert is primarily focused on obtaining a semantically optimized answer to be more efficient and save time when using the platform. So, it is critical to use an appropriate summarizing technique for this. The most often utilized methods of summary in the current era are extractive and abstractive summarization. We discovered that abstractive approaches are superior since they are more human-friendly than reading a series of lines directly from the selected answers[16]. In order to do abstractive summarization, ProbExpert employed a mix of Transformers and BERT libraries, which are commonly used in machine learning. BERT (Bidirectional transformer) is used to overcome RNN and other neural network restrictions, such as long-term dependencies. It is a bidirectional model that has been pre-trained.

G. Quiz Evaluation Technique

The platform uses the returning score of the calculated cosine similarity [D] method for the quizzes' evaluation method. The process consists of calculating the above score for three scenarios.

- The cosine similarity score between the platform's optimal answer and the user's answer. (C_o)
- The cosine similarity score of the summarized user answer and the platform's optimal answer. (C_s)
- Cosine similarity of user answer's extracted keywords and platform's answer's extracted keywords. (C_k)

ProbExpert is equipped with three different BERT models for answer evaluation (distilroberta-base, bert-base-nli-mean-tokens, t5-base), which eliminates the dependency of a single model and eliminates any single model BERT transformer model related inconsistencies[27]. Once all the similarity scores are obtained, the mean score will be presented as the final score. If the calculated cosine similarity score can be represented in C, the formula is as follows (2):

$$Score = \frac{C_o + C_s + C_k}{3} \quad (2)$$

H. User Ranking Technique

The ProbExpert developer ranking technique consists of three individual scorer models to evaluate developers' skills in three different major categories: Coding, Q&A, and developer professional achievements. GitHub, Stack Overflow, and LinkedIn are used respectively to identify and evaluate the areas mentioned above of a developer. In these three communities, GitHub is the only platform that offers a public REST API to its complete data set. Furthermore, when evaluating a developer, coding skills and contributed projects play a significant role. Therefore, the ProbExpert ranking algorithm gave the 80% of the total weight to the GitHub scorer model while giving only 20% to the other two models when finalizing the total score of a developer.

I. Scorer Models

The primary Scorer model, which is GitHub scorer consists of 4 stages. In stage one, it gathers all required features from GitHub's GraphQL API for the given username. These gathered features are briefly described below. Then it calculates the penalty and reward score for each feature. According to the formula (3), High Contributions (HC): HC measures the developer's contribution count to the repositories, with over 1000 stargazers. U/K gives the total commit percentage of the developer while S gives the total stargazers of the repository, and n gives the total repository count, which has over 1000 stargazers.

$$\mathrm{HC} = \sum_{i=1}^{n} \left(s \, \frac{u}{k} \right) \quad (3)$$

Commit Frequency (CF): Here, CF measures the current commit frequency of a developer by dividing the total number of commits of the last 365 days by 365. This value helps to determine how active the developer is currently.
Low Contributions (LC), Number of stargazers (TS), total followers (TF), total repository count (TR), number of used languages (TL), and the number of issues (NI) opened by the developer are the other activities that are taken into account as the rewarding factors for the scorer model. When determining a developer's programming skill, each factor is multiplied by a specific weight according to its hardness [11]. The method of finding suitable weights for the selected features will be discussed later in this paper.
Since This scorer model is based on the Bell Curve graph [28], considerations were applied in some activities as penalty factors to balance the distribution of the scores. Therefore, if profile age, total commit count, and the number of used languages is under a specific limit, they will be considered penalty activities. These features also get multiplied by the computed weights based on their significance. After generating the score for each feature, it sums up all scores into one score called the total dev score (TDS). Total rank weights (TR) and assigned weight for each feature were calculated by consuming the data of 60 GitHub profiles extracted using the GDM tool. Since feature weights do not relate with the population [10], a balanced data set which is based on stratified balanced sampling that included undergraduates, software engineers, senior

software engineers, open-source contributors, GitHub star developers, and beginner profiles aged less than one year, which are not included to any of above stratum, was used as a balanced sample. Two annotators were used for labelling the profiles into the classes, and then by plotting each feature against their class, weights were calculated.

Then to determine the rank of the selected developer, the normalized score is calculated based on the Bell-curve by consuming the derived TDS value from the previous step. The normal cumulative distribution function (normalCDF) has been used to derive the normalized dev score from the generated score instead of using empirical rule. since it does not give the precise value for the result, ranking users based on those values will not be accurate. However, it is impossible to calculate the cumulative distribution directly in JavaScript because it does not provide any built-in method to calculate the error function (erf) [29]. Therefore, by using the erf method of math.js opensource library, the normalCDF function was implemented.

$$NDS = \frac{1}{2}\, erf\left(\frac{TDS - TW}{\sqrt{2}\ TR}\right) \tag{4}$$

According to the formula (4), NDS gives the normalized developer score using TDS, summed values of assigned weight for each factor (TW), and the summed value of the weight of each rank (TR). Finally, by considering the left tailed p-value from the generated bell curve, a user is classified into the related class according to the values of TABLE *I*.

TABLE I : USER CLASSES DISTRIBUTION

Class	Range
Newbie	100-81
Beginner	80-56
Intermediate	55-46
Advanced	45-26
Expert	25-5
Guru	5-0

Stack Overflow and LinkedIn scorer models have also been developed using the same manner used to develop the GitHub scorer model. User Reputation, Total answered tags count, Total Number of Badges with their types, the number of endorsements, and the number of peer connections have been used as the features for these two models. Finally, the developer's overall score is calculated by giving 80% of the weight to the GitHub score and 10% each for the other two scorers.

J. Automated Portfolio Generation

By consuming the given usernames of GitHub, Stack Overflow, and LinkedIn of a developer, ProbExpert's portfolio system will first scrape and collect the developer activities and calculate the developer score as explained in the previous sections. Then it generates the developer portfolio by showcasing the developer score and other valuable and significant developer activities using timeline progress bars and graphs.

Furthermore, it includes a leaderboard that assists recruiters and other users in finding out the experts on the platform. Since these portfolios needed to be optimized for search engines, Next.js [30] has been used on the top React.js library to enable static rendering. Therefore, all the generated developer portfolios can be cached on all the major content delivery networks (CDNs), and also, they are 100% accessible by web spiders to rank them on major search engines. Also, Next.js hybrid-static page generation method is used for loading the portfolios without any server-side computation by serving the pre-rendered pages stored in the server.

IV. RESULTS AND DISCUSSION

This section performs a qualitative comparison of the proposed ProbExpert Q&A platform with the currently available other platforms.

a) Expert Identification: Since all the developers are listed on the leaderboard according to their rankings, hiring managers and potential buyers can find the unemployed or free-lancing developers in no time by filtering the leaderboard by using the available to hire tag.

b) CV Shortlisting: Recruiters can utilize ProbExpert's portfolio generating section to quickly generate and determine developers' scores and activities. This will save a recruiter a significant amount of time, which they would otherwise spend manually going through a developer's social networks such as LinkedIn, GitHub, and Stack Overflow while evaluating a candidate.

c) As guidance for newbies: Since developer portfolios include significant developer activities, viewing an expert's portfolio will give novices a solid idea of how they should shape their path accordingly to pursue their dream profession or career goals.

d) Automated answer: The system will identify multiple resources from all five separate resource websites (StackOverflow, GitHub, YouTube, medium, dev) in the best-case scenario of automated answer generation. If the user were to visit those resource websites manually, they would need to view 14 web pages, which would take 9.4 minutes (5). The average loading time for a desktop webpage is 10.3 seconds [31] (lt = 10.3). The average loading time for a desktop webpage is 10.3 seconds [31] (lt = 10.3). The user's time to determine the relevance of the resource has been calculated to be 30 seconds (rt = 30).

$$total\ time\ = \sum_{k=0}^{n} (lt\ +\ rt) \tag{5}$$

e) Optimized answer: Instead of checking each answer individually, users may easily apply the optimized answer solution to receive the most accurate and effective answer for their problems. This process will save a significant amount of time while also ensuring a working solution to the problem.

f) Structured type theoretical Quizzes from platform questions: Since the platform contains questions raised by general programmers, the quizzes made by platform's questions can be used as material to update and check the current knowledge. This facility can be helpful to individuals as well as larger organizations to evaluate or train their current employees or recruits.

V. FUTURE WORK

Due to the scarcity of data, questions with cosine similarity values greater than 80% are now considered similar; however, as data sets grow, cosine similarity values can be increased to greater than 90%. A higher cosine similarity score between questions indicates that the system will find nearly identical questions. Furthermore, a blockchain system will be applied at a later stage. The primary purpose of this implementation is to introduce a digital currency for transactions within the platform, which can be used to do various tasks, such as hiring an expert for a dedicated session.

REFERENCES

[1] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," in *2013 International Conference on Social Computing*, 2013, pp. 188–195, doi: 10.1109/SocialCom.2013.35.

[2] K. Mao, Y. Yang, Q. Wang, Y. Jia, and M. Harman, "Developer recommendation for crowdsourced software development tasks," in *Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015*, Jun. 2015, vol. 30, pp. 347–356, doi: 10.1109/SOSE.2015.46.

[3] C.-M. A. Yeung, N. Gibbins, and N. Shadbolt, "A Study of User Profile Generation from Folksonomies."

[4] L. Salas-Morera, A. Arauzo-Azofra, and L. García-Hernández, "Analysis of Online Quizzes As a Teaching and Assessment Tool," *J. Technol. Sci. Educ.*, vol. 2, no. 1, 2012, doi: 10.3926/jotse.30.

[5] B. Ross, A. M. Chase, D. Robbie, G. Oates, and Y. Absalom, "Adaptive quizzes to increase motivation, engagement and learning outcomes in a first year accounting unit," *Int. J. Educ. Technol. High. Educ.*, vol. 15, no. 1, pp. 1–14, 2018, doi: 10.1186/s41239-018-0113-2.

[6] W. C. Kao, D. R. Liu, and S. W. Wang, "Expert finding in question-answering websites: A novel hybrid approach," in *Proceedings of the ACM Symposium on Applied Computing*, 2010, pp. 867–871, doi: 10.1145/1774088.1774266.

[7] G. Zhou, J. Zhao, T. He, and W. Wu, "An empirical study of topic-sensitive probabilistic model for expert finding in question answer communities," *Knowledge-Based Syst.*, vol. 66, pp. 136–145, 2014, doi: 10.1016/j.knosys.2014.04.032.

[8] A. Capiluppi, A. Serebrenik, and L. Singer, "Assessing technical candidates on the social web," *IEEE Softw.*, vol. 30, no. 1, pp. 45–51, 2013, doi: 10.1109/MS.2012.169.

[9] S. Onoue, H. Hata, and K. I. Matsumoto, "A study of the characteristics of developers' activities in GitHub," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2, pp. 7–12, 2013, doi: 10.1109/APSEC.2013.104.

[10] E. Constantinou and G. M. Kapitsaki, "Identifying Developers' Expertise in Social Coding Platforms," in *Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, Oct. 2016, pp. 63–67, doi: 10.1109/SEAA.2016.18.

[11] J. E. Montandon, L. L. Silva, and M. T. Valente, "Identifying Experts in Software Libraries and Frameworks among GitHub Users Software Developers Expertise View project Software Defects View project Identifying Experts in Software Libraries and Frameworks among GitHub Users."

[12] H. P. Luhn, "The Automatic Creation of Literature Abstracts," *IBM J. Res. Dev.*, vol. 2, no. 2, 2010, doi: 10.1147/rd.22.0159.

[13] F. Kiyani and O. Tas, "A survey automatic text summarization," *Pressacademia*, vol. 5, no. 1, 2017, doi: 10.17261/pressacademia.2017.591.

[14] E. Yulianti, R. C. Chen, F. Scholer, W. B. Croft, and M. Sanderson, "Document summarization for answering non-factoid queries," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 1, 2018, doi: 10.1109/TKDE.2017.2754373.

[15] R. M. Alguliev, R. M. Aliguliyev, and N. R. Isazade, "Multiple documents summarization based on evolutionary optimization algorithm," *Expert Syst. Appl.*, vol. 40, no. 5, 2013, doi: 10.1016/j.eswa.2012.09.014.

[16] D. Demner-Fushman and J. Lin, "Answer extraction, semantic clustering, and extractive summarization for clinical question answering," in *COLING/ACL 2006 - 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 2006, vol. 1, doi: 10.3115/1220175.1220281.

[17] W. Chan, X. Zhou, W. Wang, and T. S. Chua, "Community answer summarization for multi-sentence question with group L1 regularization," in *50th Annual Meeting of the Association for Computational Linguistics, ACL 2012 - Proceedings of the Conference*, 2012, vol. 1.

[18] M. Keikha, J. H. Park, and W. B. Croft, "Evaluating answer passages using summarization measures," 2014, doi: 10.1145/2600428.2609485.

[19] L. Averell and A. Heathcote, "The form of the forgetting curve and the fate of memories," *J. Math. Psychol.*, vol. 55, no. 1, pp. 25–35, 2011, doi: 10.1016/j.jmp.2010.08.009.

[20] H. L. Roediger, A. L. Putnam, and M. A. Smith, *Ten Benefits of Testing and Their Applications to Educational Practice*, vol. 55. 2011.

[21] J. Piskorski, N. Stefanovitch, G. Jacquet, and A. Podavini, "Exploring Linguistically-Lightweight Keyword Extraction Techniques for Indexing News Articles in a Multilingual Set-up," *Proc. EACL Hackashop News Media Content Anal. Autom. Rep. Gener.*, pp. 35–44, 2021.

[22] S. Desai and G. Durrett, "Calibration of Pre-trained Transformers," pp. 295–302, 2020, doi: 10.18653/v1/2020.emnlp-main.21.

[23] T. Wolf *et al.*, "Transformers: State-of-the-Art Natural Language Processing," pp. 38–45, 2020, doi: 10.18653/v1/2020.emnlp-demos.6.

[24] M. Grootendorst, "Keyword Extraction with BERT," 2020. .

[25] D. Gunawan, C. A. Sembiring, and M. A. Budiman, "The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents," *J. Phys. Conf. Ser.*, vol. 978, no. 1, 2018, doi: 10.1088/1742-6596/978/1/012120.

[26] B. Li and L. Han, "Distance weighted cosine similarity measure for text classification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8206 LNCS, pp. 611–618, 2013, doi: 10.1007/978-3-642-41278-3_74.

[27] A. Chernyavskiy, D. Ilvovsky, and P. Nakov, "Transformers: 'The End of History' for NLP?," 2021.

[28] "The Bell Curve: Intelligence and Class Structure in American Life - Richard J. Herrnstein, Charles Murray - Google Books." .

[29] A. Soranzo and E. Epure, "Simply Explicitly Invertible Approximations to 4 Decimals of Error Function and Normal Cumulative Distribution Function," no. 2, pp. 3–5, Jan. 2012.

[30] L. Hussein and S. Jubell, "Hur rendering påverkar prestanda och sökmotoroptimering : – NextJS vs React," 2021.

[31] Backlinko, "web page speed analysis - 5.2 million desktop and mobile pages," *Backlinko*, 2019. https://backlinko.com/page-speed-stats (accessed Aug. 28, 2021).