Improving Content Quality for Online Professional Activities using Domain Specific Learning and Knowledge

By

Nidhi Goyal

under the supervision of

Prof. Ponnurangam Kumaraguru
Dr. V. Raghava Mutharaju
Dr. Niharika Sachdeva

Indraprastha Institute of Information Technology, New Delhi

June, 2024

Improving Content Quality for Online Professional Activities using Domain Specific Learning and Knowledge

By

Nidhi Goyal

submitted

in partial fulfillment of the requirements

for the award of the degree of

Doctor of Philosophy

to the

Indraprastha Institute of Information Technology, New Delhi

Okhla Industrial Estate, Phase III

New Delhi, India - 110020

June, 2024

Indraprastha Institute of Information Technology-Delhi
Okhla Industrial Estate, Phase III
New Delhi-110020, India
June-2024

Certificate

This is to certify that the thesis titled **Improving Content Quality for Online Professional Activities using Domain Specific Learning and Knowledge** being submitted by **Nidhi Goyal** to the Indraprastha Institute of Information Technology-Delhi, for the award of the degree of **Doctor of Philosophy**, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Prof. Ponnurangam Kumaraguru, Dr. V. Raghava Mutharaju, Dr. Niharika Sachdeva
Professor, Assistant Professor, Associate Vice President
IIIT-Hyderabad, IIIT-Delhi, InfoEdge India Limited
New Delhi-110020
June, 2024

Indraprastha Institute of Information Technology-Delhi
Okhla Industrial Estate, Phase III
New Delhi-110020, India
June-2024

Declaration

This is to be certified that the dissertation entitled **Improving Content Quality for Online Professional Activities using Domain Specific Learning and Knowledge** being submitted by **Nidhi Goyal** to the **Indraprastha Institute of Information Technology-Delhi**, for the award of degree of **Doctor of Philosophy**, is a bonafide work carried out by me. This research work has been carried out under the supervision of **Prof. Ponnurangam Kumaraguru, Dr. V. Raghava Mutharaju, and Dr. Niharika Sachdeva**.

The study pertains to this dissertation has not been submitted in part or in full, to any other University or Institution for the award of any other degree.

Nidhi Goyal
PhD student
IIIT-Delhi, India

Acknowledgements

I would like to express my appreciation to my academic supervisors Prof. Ponnurangam Kumaraguru ("PK") and Dr. V. Raghava Mutharaju for giving me the opportunity to work on this thesis and enlighten my PhD journey through their continuous support, active monitoring, and timely feedback. PK played an instrumental role in shaping my career to academia, giving me enough opportunities and freedom to work in different research areas of interest. He provided me with a perfect platform and ecosystem to showcase my abilities, providing unwavering encouragement and support at every stage. I want to thank him for keeping track of every milestone achieved throughout my PhD journey.

Dr. Raghava accepted the role of being my mentor and guided me in my research. I would thank him for his patience and for making it easy to ask the dumb questions. I really appreciate his openness and support to my research topic. He always encourages me to think deeper and go beyond architecture engineering in research. I am very grateful to KRaCR lab activities and projects for giving me a clear understanding of the domain-specific things which were essential for completing this work. I thank him for his co-operation, enthusiasm, highest priority to research, and patience throughout my PhD journey.

I would like to express my deepest appreciation to my industry mentor, Dr. Niharika Sachdevawho made an impact in moulding my professional path. Her guidance, support and advice encouraged me throughout my professional journey. She has consistently challenged me to think critically, work hard, and provided me with opportunities to grow and develop my skills. Her belief in my abilities has inspired me to aim higher and pursue my goals with passion and dedication. I have gained confidence in my abilities (especially how to write, read papers, conduct literature review), expanded my knowledge (about technical aspects), and built valuable connections in research domain. I will always be grateful for her passionate research discussions and paper draft reviews.

The work in this thesis wouldn't have been possible without the continuous support of advisors. At the end, without their valuable thoughts and recommendations, It would have been difficult for me to complete this dissertation. I would like to thank them for providing excellent research facilities. I am grateful to the Indraprastha Institute of Information Technology-Delhi for pro-

viding excellent infrastructure, research environment and research-oriented course work.
I want to direct a big thank you to **InfoEdge** India Limited and Science Engineering and Research Board (SERB) for providing me with a research fellowship under **Prime Minister fellowship for Doctoral Research Scheme** that helped me financially throughout my PhD. This Public-Private partnership and working with industry has been a special part of my PhD time which nourished my research skills and brought a big positive change in me. Special thanks to amazing collaborators Nitendra Singh Rajput and analytics team members.
My sincere thanks to **Dr. Radhika Mamidi** and **ACM Anveshan Setu Fellowship** that offered me the internship at LTRC, IIITH. I would like to say thanks for her guidance, motivation, positive words, and continuous technical/non-technical discussions.
I am grateful to **Dr. Siddhartha Asthana** who played a crucial role in helping me understand the fundamentals of core research. His guidance, support, and patience is invaluable in my learning process. Without his help, I would not have been able to grasp the hidden nuances of writing and technical complexities. He helped me especially during the low phases of my PhD with long talks which cleared up my thoughts not only about research but also in general about life. I am truly grateful for his time and effort, and I feel fortunate to have had such a knowledgeable and generous mentor.

I would also like to extend my thanks to my professional colleagues from **PreCog Research Group** at IIIT-H and **KRaCR** lab at IIIT-D. I would like to specially thank all my fellow colleagues namely Anmol Goel and Jushaan Singh Kalra I would also like to express my sincere thanks to Vijay Chaudhary, Rijula Kar, Nausheen Fatma, Raghav Mehta and Udit Arora for our discussions, shepharding, and discuss necessary technicalities which enhanced my deeper understanding of concepts. I would like to thank Saurabh and Avinash Tulasi for discussing several computer and server-related issues. I would like to thank Shivangi Singhal, Hitkul, Prashant, Dr. Geetika Arora, Kumari Neha, Ritwik, Saurabh Gupta, Monika Jain, Mayank, and Gunjan Singh for discussing several courses and research work.

I would like to thank all my spiritual teachers namely BK Shivani, BK Kuldeep, BK Vamishi, Prof. EV Girish, BK Sujata, BK Prabha, and Dr. Mohit Gupta for motivating me through their angelic lives and make me a divine being with good management skills, positivity, emotionally independent and energetic through their spiritual guidance and meditation sessions. I would

also like to thank my yoga teacher Laxshmi for his yoga classes and for encouraging me to keep myself physically fit and mentally confident.

I would also like to gratefully acknowledge Dr. Kajal Kansal, Dr. Neelam Sharma, Dr. Tania Sidana, Dr. Anshul bhatia, Dr. Mitali Sinha, Pushpita Majumdar, Priti Kumari, Anuradha Gupta for their invaluable friendship and emotional support. I am also grateful to my Yuktahar mess and people (Siddhant Bansal, Chirag Jain, Mangal, Praguna, Anapoorani, Ayappa, Raghav Mittal, Arhant Jain and many more) whom I met in Yuktahar for having cheerful and fruitful discussions over the food table.

I am extremely grateful to my committee members and collaborators : Dr. Rajiv Ratan Shah, Dr. Charu Sharma, and Dr. Radhika Mamidi for their valuable comments. My gratitude also extends to the technical and admin support staff of IIIT-D and IIIT-H, especially Mrs. Priti, Mrs. Anshu, Mr. Raju Biswas, Imran, and Mr. Krishna Kishore for being extremely helpful in the fast resolution of all technical and admin related matters.

My family's support, love and cooperation have been enormous throughout my journey. I am thankful to my parents for their blessings, my brother, Sai Amrit Patnaik, my sister, Karuna Aggarwal and all my relatives for their full support under all conditions, not just during PhD but throughout my life.
During my PhD especially deadlines and meetings, my family supported me, and I recall having a small mandir in the corner of my room. Especially during the COVID-19 pandemic, when my mom stayed with me, she took care to ring the holy bell (ghanti) slowly and enter the room quietly, ensuring everything remain undisturbed.
To my Supreme father, Shivbaba, Words cannot adequately express the gratitude I feel towards you, the vital force of my life. I would like to dedicate my deepest emotions to thank you for the companionship, empowerment, blessings, inspiration, enlightenment, and positive energy you have brought into my life. You hold a special place in my soul, and I dedicate this thesis to you.

Finally, I would like to dedicate this thesis to my late Father, who could not be here with me today.

Nidhi Goyal

Contents

List of Figures

List of Tables

Abstract

Online professional platforms such as Indeed, LinkedIn, Naukri, StackOverflow, and Blind serve as digital ecosystems to connect professionals, employers, and job seekers. These platforms witness online user activities, including finding jobs, candidate matching, and posting a job, which result in a voluminous amount of User Generated Content (UGC). UGC content primarily includes rich information about job postings, CVs, candidate posts, recruiter profiles, etc. The quality of this content varies from meaningful information to misleading content, depending on the expertise, reliability, and intention of the users. Even though information helps job seekers find the right jobs, the unmonitored nature of the content (including ambiguous, redundant, missing, off-topic, scams, misleading, or irrelevant information) makes it difficult to assess the content quality, thereby affecting the platform's trustworthiness, reducing the value to its customers, and, in turn, hampering the user experience. For instance, the content comes with multiple variations of each entity name (e.g., 'economictimes.com'; 'eco.times'; 'the economic times'; 'economic times'; 'ET'). These multiple non-standardized variations (noisy, redundant, and ambiguous), when directly incorporated into downstream applications such as semantic search, question answering, and recommender systems, result in poor system performance. Similarly, statistics from the Singaporean recruitment platform shows that 65% of the job descriptions (JDs) do not include relevant and popular skills. At the same time, 40% of JDs miss listing 20% or more explicitly-stated skills in the prose description. It reduces the number of relevant applications for the job posting and affects the performance of significant recruitment tasks such as job-to-resume matching. With millions of job seekers per month, these candidates often come across dishonest, money-seeking, intentionally and verifiably false information in jobs such as offering more wages, flexible working hours, and appealing career growth opportunities. The proliferation of these jobs not only hampers candidate's experience but also acts as a repressing factor in an enterprise's reputation. Given the platforms are open-source and anyone ranging from novices and experts can upload content to these platforms, low-quality questions (lack of clarity, off-topic content, primarily opinion-based, too broad) also come up. Therefore, it is crucial to maintain the quality of content posted on these platforms. The thesis adopts a four-fold approach to address content quality issues for online professional platforms. The first phase of this work centres around normalizing content on online professional platforms. The second phase aims to predict missing skills to enhance job quality over these platforms. The third phase involves modeling a framework to detect misleading content on recruitment platforms. This requires mining unstructured recruitment data from various sources to obtain structured information and creating domain-specific knowledge graphs. We also delve into understanding employment scam complaints to help platforms continuously refine their advisories based on user complaint base and feedback to ensure they stay updated with the dynamically evolving tactics used by scammers. The fourth phase focuses on identifying low-quality information for question-answering services. In conclusion, we contribute by building automated solutions to improve content quality for online professional activities using domain-specific learning and knowledge.

Chapter 1

Introduction

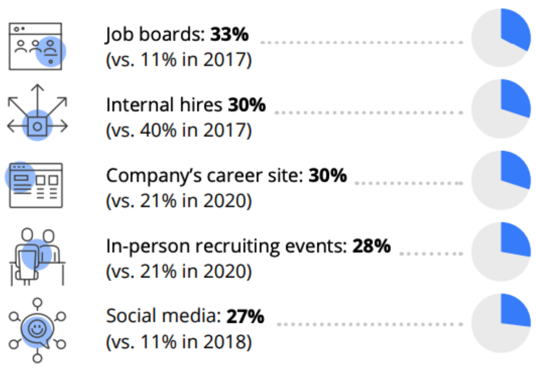
"You can't build a great building on a weak foundation. You must have a solid foundation for a strong superstructure."

- Gordon B. Hinckley

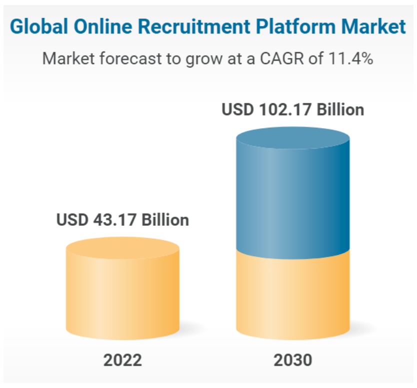
Over the years, technological advancements have enhanced professional activities, transforming the way professionals communicate and share knowledge. According to Bureau of Labor Statistics[1], there is a massive increase in employment opportunities for professional, scientific, and technical services in the decade (2021-2031). The rapid growth of professional activities has led to a parallel rise in the development and expansion of online professional platforms such as Indeed [5], Simply Hired [6], Monster [7], Glassdoor [8], LinkedIn [9], CareerBuilder [10], Naukri [11], StackExchange [12], StackOverflow [13], Blind [14]. Simply Hired has more than 8 million job postings and 30 million monthly users globally. CareerBuilder [10] contains over 45 million candidate resumes and 1.6 million jobs. Naukri.com [11] has 2-3 million job applications daily and 85 million job seekers available for recruiters [15]. Monster.com [7] has 7.9K jobs searched and 2.8K jobs viewed every minute. Stack Overflow (SO), a job search service[2] and a popular Community Question Answering (CQA) has 22.1 million questions and 16.6 million users as of 2022 [16]. Due to the extensive reach and popularity of online professional platforms (Figure 1.1(a)), the global online recruitment market size [17] is expected to reach USD 102.17 billion by 2030 with more than billions of monthly active users (Figure 1.1(b)). Such activities generate an enormous amount of user-generated content [13, 18, 19] including job postings, CVs, skills, company names, institute names, technical questions, posts, blogs, experience, and qualifications.

[1]https://www.bls.gov/iag/tgs/iag54.htm#about

[2]https://stackoverflow.com/jobs/companies



(a)



(b)

Figure 1.1: (a) Jobvite generated the 2021 Indian Recruiter Nation Report to determine current hiring trends and the industry's priorities. According to statistics, Job boards as the top hiring sources for high-quality candidates. Image source: Top hiring sources for high-quality candidates, (b) The Global Online recruitment platform market is forecasted to grow significantly, at a CAGR of 11.36%, to reach a staggering USD 102.17 billion by 2030. Image source: Global Online recruitment platform cumulative impact of high inflation and forecast from 2023-2030.

1.1 Content quality

There is large volume, velocity, and variety of content on professional recruitment platforms. Their quality ranges from meaningful information to misleading content depending on the expertise, reliability, and intention of the users [20]. According to Gartner Research's data quality market survey[3], poor content quality resulted in an annual financial burden of approximately $15 million dollars for the organization. Organizations such as IBM also declared that businesses in the United States lose about 3.1 trillion dollars annually due to data quality issues.
An industry report [21] shows that more than 60% of the surveyed firms (500 medium-size corporations with annual sales of more than $20 million) have problems with content quality. Poor quality can have substantial social and economic impacts. Hence, the quality of content/data has always been a critical concern for organizations.
Existing content/data quality literature [21] adopt two general types of strategies, namely data-driven and process-driven. Data-driven strategies improve the quality of data by directly modifying its value. For example, obsolete data values are updated by refreshing a database with data from a more current database. On the other hand, process-driven strategies improve quality by redesigning the processes that create data or knowledge. For example, a framework can be redesigned that can control the format of data before storage. Research [22] says that quality is not inherent in data but follows when someone can use it successfully. 'Quality' follows from

[3]https://www.forbes.com/sites/forbestechcouncil/2021/10/14/flying-blind-how-bad-data-undermines-business/?sh=65ff61f529e8

the value content create after being put to use.
The concept of "*fitness for use*" is widely adopted in the quality literature [22, 23]. Fitness refers to the suitability of content for its intended purpose. For instance, a job posted by a recruiter should have multiple verticals of heterogeneous information. These verticals include position, skills, shift timings, company profile, salary, job roles and responsibilities. This job information is crucial for potential candidates in making an informed decision about applying. On the other hand an unclear job posting lacks important information. Recruiters would potentially miss out on relevant candidates. Likewise, questions from candidates sparking on-topic discussions enrich the collective knowledge of the professional network. Since most of the content on online professional platforms is user-generated, understanding content quality is crucial due to the enormous volume of user generated content.
A large percentage rely on these platforms for their careers, business, and networking. Such user-generated content carries the potential for a range of quality issues, including inconsistency, inaccuracies, grammatical errors, lack of clarity, missing information, off-topic elements and misleading information.
Towards this end, we observe various content quality issues from an online professional platform user's (recruiters, candidates, job seekers) point of view. These issues are broadly categorized into (a) non-standardized (inconsistent, non-standard, noisy, redundant, and ambiguous) content, (b) incomplete and missing information, (c) misleading information (mismatch in skills, industries), and (d) low-quality content (off-topic, unclear, too-broad, opinion-based).
Non-standardized and inconsistent entities: Online professional content typically contains non-standard user-generated entities. These diverse entities (company name, institute name, designation, or skills) could have multiple variations for the same entity. For example, entities may have multiple representations named *'warner bros.'*, *'warner brothers pictures'*, and *'warner bros. entertainment inc.'* for the same real world entity *'Warner Brothers Studios'*. A popular recruitment domain dataset [2] has 497 variations of *'ICICI Prudential Life Insurance'*, 1145 variations of *'Dr. Babasaheb Ambedkar Marathwada University Aurangabad'*, and 123 variations of *'Senior Software Engineer'*. Such name variations lack standard nomenclature or representation, thereby resulting in many impediments (a) Inaccurate or inconsistent representations of job-related information affect the accuracy of statistical analysis, impacting decision-making processes for both job seekers and recruiters, (b) hinders the efficiency of job matching processes leading to mismatch between suitable candidates and relevant job positions, (c) requires additional computational resources resulting in slower system performance, (d) organizations can possibly face legal implications if hiring decisions are influenced by misrepresentation of entities leading to discriminatory practices. Considering these challenges, standardization (i.e., mapping multiple references of the same entities to a real-world entity) in the recruitment process is crucial to ensure consistency, accuracy, and efficiency in the job

search for job seekers and recruiters.

Incomplete and missing information: Online professional platforms often receive millions of job postings on their platforms. In reviewing extensive job postings, job seekers predominantly assess the prerequisites of the job by searching for important skills. Despite the extensive reach of job boards, the recruiters sometimes miss the relevant information such as skills [24] while posting the job on the platform. Missing skills problem may have occurred due to the communication gap between the recruiters (who may have little in-domain expertise) and the target employer (who are domain experts). Around 78.86% of explicit skills are missing in job descriptions [25]. The incompleteness diminishes the platform's overall utility for job seekers and employers, particularly for skill-based job matching algorithms. This skill gap can lead to an influx of applications from candidates who do not meet the necessary requirements, complicating the candidate selection process and potentially causing delays in hiring. Therefore, it is imperative to assist recruiters in identifying missing information and crafting high-quality job postings to attract relevant candidates, increase the volume of applications, and filter the recruiting process's best talent.

Misleading information: In addition to inconsistent variations and incomplete job postings, candidates often come across jobs offering competitive salaries, flexible working hours, and appealing career growth opportunities. Such job postings could contain many untenable facts. For instance, data entry-level positions offer \$1000-\$2500 per week to a fresher candidate without specialized skills. Such content mislead users [26], displays or promotes false jobs [27], and fraudulent representations [28]. Such jobs are dishonest, money-seeking, intentionally and verifiably false, and mislead job seekers. Despite having a high-performance immune system [9] in the recruitment domain, these platforms need to consistently uproot content that misleads users and create an avenue for fabricated content. Typically, misleading content detection has two components: a) understanding and extracting domain-specific facts and b) building an automated pipeline to flag such misleading documents. The first component involves extracting relevant domain-specific information (skills, qualifications, roles, location, etc.) from documents. Enterprises invest considerable time and effort to extract, evaluate, and organize domain-specific information in semi-structured or structured format [29]. These organizations need well-designed data extraction pipelines to automatically leverage the information in unstructured data and integrate it with existing structured databases [30]. In this way, there is better control over unstructured data that can facilitate efficient decision-making capabilities.
This is a challenging task due to a) dependency upon the style of writing (it is subjective) and b) ambiguous entities (such as '*Microsoft*' as a skill or company name) c) heterogeneous information from various sources. For instance, a job description mentioning *'Bachelors from an IIIT'* in the unstructured field and *'B.Tech'* as a qualification in the structured field needs integration to draw accurate conclusions. Thus, there is a need to develop a recruitment domain

knowledge framework to pull out all necessary information from unstructured documents that machines can understand. Finally, it is desirable to detect and take off misleading information by exploiting the structured facts present in these documents.

Low-quality content: Another aspect of content quality comes when recruiters review online professional communication of candidates, which helps them assess their technical skills, problem-solving approach, and communication abilities, determining their suitability for a specific job role. When candidates ask clear, well-researched, and relevant questions on community Q&A platforms, they are more likely to receive helpful responses and upvotes from the community. This, in turn, boosts their reputation and credibility as a knowledgeable and engaged community member. Therefore, how they ask questions directly impacts their profile (reputation) on online professional platforms. Sharing insights, asking relevant questions, and providing constructive feedback are crucial to understanding a particular topic and concept. However, research [31] suggests that unanswered questions on knowledge-sharing platforms are still being viewed 139 times on average and this motivated enterprises to investigate why such questions remain answered. The primary reasons for unanswered questions typically include being too short, unclear, vague, difficult to follow, irrelevant, not constituting a proper question, or being a duplicate. Such low-quality content (off-topic discussions, opinion-based statements, or excessive breadth) not only diminishes the quality of interactions but also undermines trust and engagement among customers. Therefore, it is imperative to identify such low-quality content and enhance the quality of information that will encourage others to participate actively, ultimately enriching the platform's knowledge pool and promoting meaningful learning and exchange of ideas.

1.2 Existing Challenges in Content quality

Despite being a content quality problem studied for over a decade, existing techniques use handcrafted, stylometric, and linguistic features. Below, we discuss some challenges in conducting content quality research for online professional activities.

- **Unavailability of domain-specific knowledge:** To extract information from unstructured data, various natural language processing techniques such as Named-Entity Recognizer tools [32], POS taggers [33] require domain-specific knowledge. Online professional platforms do not have publicly available domain-specific knowledge bases. Additionally, recruiters find it challenging to engage in meaningful discussions regarding job requirements, candidate qualifications, and technical assessments which hinders effective collaboration and decision-making without domain-specific knowledge.

- **Extracting knowledge from heterogeneous data:** Given the diversity of unstructured content offered by different users such as job seekers, recruiters, etc. (*who*) and users of different levels produce or submit content (*what*) and understanding user intent (*why*) vary from one platform to another. Therefore, knowledge extraction and understanding from voluminous unstructured user-generated content from heterogeneous sources on professional networks is a big challenge.
- **Validity of Data:** Given the dynamic nature of data, it is difficult to predict the correctness and validity of the facts [34], particularly temporally changing relationships. Valid data serves as the foundation for fact-based content. When content is based on reliable and valid data, it increases its credibility and trustworthiness. This enhances the quality of the content and provides value to the users by presenting accurate and verifiable information.
- **Data Consistency:** Maintaining consistency across various information present in jobs, posts, or CVs is a challenging task in the recruitment domain. To enable accurate computation, besides a domain-specific Knowledge Base (KB) of employers, we need a system that can do name normalization, i.e., linking the employer names in the job postings and the resumes to the employer entities in the KB. Jobs contain context-specific information that needs standardization [35], provenance information [36], entity alignment [37], etc. based on underlying schema and representation.
- **Leverage the location context:** Job postings and entities (company names, institute names) are often associated with location information. The location information associated with these entities is helpful for normalization, but it could be empty or inaccurate.

To overcome the above challenges and enhance the quality of information, researchers have analyzed, studied and experimented on the content and formulated new techniques and methodologies for other platforms. Research on online professional websites is still relatively unexplored and warrants further investigation [27].

1.3 Core Thesis Question

Given the above-mentioned challenges, we address the issue of standardization, dealing with missing content, and identifying low-quality and misleading information to improve content quality for online professional activities. We define the core research question of this thesis as:

How can we improve the quality of online professional content by leveraging domain-specific learning and knowledge?

1.4 Thesis Contributions

This thesis makes the following contributions towards the core research question. In this work, we leverage domain-specific knowledge to address the above research goal. This thesis helps to enhance, build automated and improved content quality-based systems for online professional platforms. In this thesis, we concentrate on four different aspects of content quality (Figure 1.2).

- The first contribution focuses on how to standardize or normalize the entities (such as skills, institutes, companies, and designations) on online professional platforms.
- The second contribution deals with finding missing entities (job skills) to improve the quality of jobs on online recruitment platforms.
- The third contribution presented in this thesis focus on identifying misleading content (Fraudulent jobs) on online recruitment platforms.
- In the final contribution, we address the problem of identifying low-quality (off-topic, unclear, opinion-based) content for question-answering services.

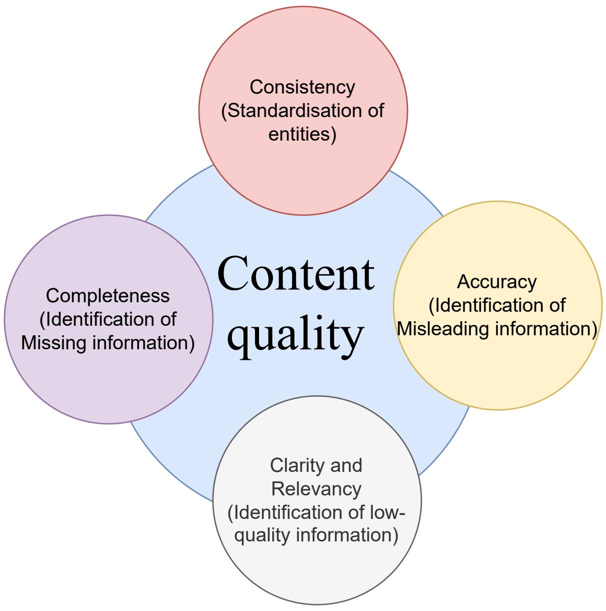

Consistency (Standardisation of entities)
Completeness (Identification of Missing information)
Content quality
Accuracy (Identification of Misleading information)
Clarity and Relevancy (Identification of low-quality information)

Figure 1.2: shows an overview of contributions (content quality issues) in online professional activities.

1.4.1 Standardization of entities in online professional content

To build an effective normalization system, the system should be (a) able to handle employer names from both job postings and resumes, which are in semi-structured format and could come from different sources, (b) leverage the location context, (c) handle name variations (e.g., 'TCS' vs. 'Tata Consultancy services'), (d) handle irrelevant or unlinkable input data. Normalization/Canonicalization cannot be directly applied to domain-specific content due to: (a) low coverage of domain-specific entities in Open KB. For example, only 10% of entities in recruitment domain data were present in Freebase [38]. So research that relies on using Open knowledge bases to help in disambiguation [38] cannot be used for the recruitment domain; (b) Handcrafted features (e.g. lexical, semantic, etc.) are limited as they do not scale to capture the volume of variations. Such approaches have been known to be cost-in-efficient and sub-optimal [39]; (c) Deep learning approaches incur high computational costs when deployed as real-world systems. These limitations warrant new solutions enabling entity canonicalization in the recruitment domain. To address these challenges, we propose a novel Canonicalization framework that outperforms all the known statistical and deep learning methods and uses side information such as industry type, URL of websites, etc. further enhancing the performance of the proposed method.

1.4.2 Deal with missing entities in online professional content

Solving the issue of missing entities on platforms is important because it ensures complete and accurate information, improves user experience, and promotes inclusivity and representation. Among these entities, skill is the most important parameter to determine whether the candidate will be selected for the position. To find missing skills, existing works [40, 41] explored missing skill recommendation tasks using large-scale pre-trained language models. However, these approaches fail to exploit the structural relationships between jobs and skills across the dataset. Current deep extreme classifiers [42, 43] find it hard to model such implicit relationships and give equal importance to every skill corresponding to the job. Similarly, language models bring high computational costs at massive scales as a task not only involves predicting multiple missing skills but also requires precisely organising the most relevant skills specific to the job posting. To address these challenges, we construct a novel job-skill graph that allow flexible integration of textual features and various pre-trained language representation models. We cast our problem as an EXtreme Multi-label Classification (XMLC) using a job-skill graph and propose a graph neural network-based framework with skill attention to learn multi-resolution graph neighborhoods with the sampling method.

1.4.3 Identify misleading information in online professional content

To accomplish this objective, we have partitioned the problem into two components.
Extract domain-specific information and provide structure to online professional content: This component focuses on the crucial part of the intelligence for enterprise applications, i.e., to extract domain-specific information from millions of unstructured documents and integrate them to draw useful insights. Taking the recruitment industry as an example, services focused toward the user's experience are an essential part of value creation and are becoming increasingly more important; however, it is not just the job postings by recruiters but also the valuable information that can be automatically extracted from job seeker profiles and posts that can help enterprises improve applications and serve users. In the recruitment domain, the extracted information can have various attributes. For example, the characteristics of a job include job type, date posted, job requirements, job role, skills required, etc. Furthermore, the attributes can have hidden relationships; e.g. two jobs should have a relationship if they require a common skill of 'Python' in their unstructured content. Hence, extracting useful information from unstructured content and converting it into structured form is imperative for these platforms. Towards this end, our goal is to structure this domain-specific recruitment knowledge to map hidden properties such as type of recruiter, shift timings, interview dates, etc. To accomplish this, we introduce a large-scale domain-specific knowledge extraction end-to-end pipeline, transforming unstructured text into a structured format.
Build a framework to classify misleading information using domain knowledge and deep learning-based approaches: In this part, we aim to build systems or frameworks that leverage the extracted information to assist various downstream tasks. We utilize datasets from popular online recruitment platforms to develop frameworks to classify misleading data present in documents based on the domain-specific information extracted in the previous section. To accomplish this, we introduce a multi-tier hybrid framework, which leverages domain-specific recruitment knowledge and deep learning-based approaches to classify fraudulent and non-fraudulent job postings in the domain.

1.4.4 Identify low-quality content

In this objective, we study online professional activities including questions, answers, and posts from knowledge-sharing platforms. Since low-quality activities increase the workload for community moderators and highly experienced users who spend time engaging with content leading to the waste of their efforts.[4] Such low-quality content falls under the umbrella (*'off-topic'*, *'unclear what you're asking'*, *'primarily opinion-based'*, and *'too broad'*). Therefore, we propose a multi-tier hybrid framework for low-quality content detection that incorporates the content-related information associated with each question using BERT. We also introduce a novel way to exploit the tag-related information to capture implicit relationships between a question and tag using Graph Convolutional Networks (GCNs) [44]. Furthermore, we conduct survival analysis [45] to provide various insights on questions (closed) by comparing the median survival time and the mean time till the Event of Interest (the time elapsed from the question's creation to closure). Our framework addresses the cold-start problem using only pre-submission information of questions posted over the platform. To achieve these objectives, we construct a framework that can serve as an early assessment mechanism for content quality. Such a framework would help moderators to identify the potential low-quality content and uproot them by taking actionable measures.

1.5 Thesis Organization

The rest of the work is structured as follows.

- Chapter 2 discusses the background of content quality and domain-specific knowledge along with research gaps.
- Chapter 3 describes our proposed multi-tier approach for data standardization of recruitment domain entities.
- Chapter 4 focus on finding missing skills in job postings to enhance the quality of job postings.
- Chapter 5 presents the proposed framework for information extraction and construction of domain-specific knowledge graphs.
- Chapter 6 presents the hybrid framework for the identification of misleading job postings.
- Chapter 7 focuses upon identifying low-quality content on knowledge-sharing platforms.
- Chapter 8 concludes the thesis by summarizing the contributions and discussing future research work on content quality.

[4]https://meta.stackexchange.com/questions/166623/what-is-a-day-in-the-life-of-a-stack-exchange-moderator-like/166630#166630

- Chapter 9 explores the future directions and preliminary work done in the context of content quality.

1.6 Publications

The work in the thesis is primarily related to the following peer-reviewed articles:

- **Goyal, N.**, Kalra, J., Sharma, C., Mutharaju, R., Sachdeva, N., & Kumaraguru, P. (2023, May). JobXMLC: EXtreme Multi-Label Classification of Job Skills with Graph Neural Networks. In Findings of the Association for Computational Linguistics: EACL 2023 (pp. 2136-2146).

- **Goyal, N.**, Mamidi, R., Sachdeva, N., & Kumaraguru, P. (2023, January). Warning: It'sa scam!! Towards understanding the Employment Scams using Knowledge Graphs. In Proceedings of the 6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD) (pp. 303-304).

- **Goyal, N.**, Arora, U., Goel, A., Sachdeva, N., & Kumaraguru, P. (2022, July). Ask it right! Identifying low-quality questions on community question answering services. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

- **Goyal, N.**, Sachdeva, N., Goel, A., Kalra, J. S., & Kumaraguru, P. (2021, September). KCNet: Kernel-based canonicalization network for entities in recruitment domain. In Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part II 30 (pp. 157-169). Springer International Publishing.

- **Goyal, N.**, Sachdeva, N., & Kumaraguru, P. (2021, August). Spy the lie: fraudulent jobs detection in recruitment domain using knowledge graphs. In Proceedings of Knowledge Science, Engineering and Management: 14th International Conference, KSEM 2021, Part II 14 (pp. 612-623). Springer International Publishing.

- **Goyal, N.**, Sachdeva, N., Choudhary, V., Kar, R., Kumaraguru, P., & Rajput, N. (2019, September). Con2kg-a large-scale domain-specific knowledge graph. In Proceedings of the 30th ACM Conference on Hypertext and Social Media (pp. 287-288).

The following publications have been completed over the course of the PhD but are not discussed in the thesis:

- Jaglan, K., Pindiprolu, M. C., Sharma, T., Singam, A. R., **Goyal, N.**, Kumaraguru, P., & Brandes, U. (2024, May). Tight Sampling in Unbounded Networks. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 18, pp. 704-716).

- **Goyal, N.**, Goel, A., Garg, T., Sachdeva, Kumaraguru, P. (2023, December). Efficient Knowledge Graph embeddings via Kernelized Random Projections. In Proceedings of Big Data Analytics in Astronomy, Science, and Engineering: 11th International Conference on Big Data Analytics, BDA 2023 (pp. 198-209).

Chapter 2

Background

"Develop an interest in life as you see it; the people, things, literature, music-the world is so rich, simply throbbing with rich treasures, beautiful souls, and interesting people. Forget yourself."

- Henry Miller

In this chapter, we lay the foundations and present related work in all the contributions of this thesis. We first provide an overview of the relevant literature in the broader area of content quality on online platforms (Section 2.1). Through this thesis, we aim to make significant contributions to the understanding and improvement of content quality in the context of professional interactions using domain-specific learning and knowledge (Section 2.2). We delve into the various dimensions of literature on content quality, domain-specific knowledge, and tools and techniques commonly used in content quality literature (Section 2.3). Finally, as AI becomes more competent, the community is looking towards downstream tasks requiring these solutions be deployed in real-world ecosystems, such as search systems, content moderation systems, flagging posts, recommendations, and nudging users. In conclusion, we identify research gaps (Section 2.4) for content quality improvement in online professional networks.

2.1 Content quality on Online platforms

Content quality [23] refers to the overall goodness, reliability, and usefulness of the information presented in a piece of content, whether a document, post, blog, or database. It determines how valuable, accurate, relevant, complete, and well-presented the content is for intended applications or use cases. The content quality problem is exacerbated by potential issues such as informativeness, bias, trustworthiness, consistency, diversity, etc., with user-generated content.

Despite the growing research on related areas, content quality remains an ill-posed concept. Most notably, no uniform definition of content quality or quality criteria exists. This is primarily attributed to the fact that what is considered high-quality content is highly subjective and is appropriate for specific use cases, but its quality may be insufficient for other purposes [23].

Datasets	**Related work**	**Techniques used**
Twitter	[46–50]	Statistical methods, User behavior
Facebook	[51–53]	Content, network, and User features
StackOverflow	[54–56]	Network and User features
Quora	[57–59]	User behavior and Deep Learning
Reddit	[60–62]	Statistical methods, Quality metrics
ReVerb45K	[63, 64]	Knowledge Graph Representation features
Recruitment domain datasets	[2, 27, 40, 65–68]	Handcrafted features and Deep Learning methods

Table 2.1: Distribution of prior works based on combating content quality across various online platforms. These research studies address the challenges of maintaining high-quality content in the diverse online ecosystem.

Online platforms such as Twitter [50], Facebook [51], and Reddit [60] generate large volumes of user-generated content, more specifically, the presence of spam, compromised accounts, malware, and phishing attacks are significant concerns with respect to the quality of information on online social platforms. One significant drawback of the current solutions that predominantly utilize generic approaches is their lack of generalizability to other domains. These methods struggle to adapt to new contexts, enterprises, and other domains. This limitation hinders their broader applicability and reusability, making it challenging to transfer knowledge and insights gained from one domain to another. Table 2.1 shows the distribution of prior works and datasets focusing on combating content quality across various online platforms. The last column shows the different techniques used to solve content quality issues on online platforms. The most common methods existing research uses are based on content or user features.

2.2 Content quality and Online professional activities

This section investigates five aspects of content quality on online professional networks. The most critical aspects of content quality [23, 69, 70] include:

- Consistency [69]: When content follows standardized practices, it becomes easier to effortlessly navigate, comprehend, collaborate, compare, and share information. Enterprise applications can integrate information from various sources seamlessly by adopting standardized data formats and structures. This streamlined user experience enhances productivity and proficiency when performing tasks with the application.

- Completeness [23]: The complete content must include crucial details by filling in the gaps in provided knowledge. For instance, incomplete job postings deter potential candidates from applying, and recruiters miss out on talented applicants.

- Accuracy [70]: The information presented should be factually correct and free from misleading information. For instance, misleading job descriptions cause humongous losses to an enterprise, making accuracy critical in all recruitment efforts to attract talent, build a competent workforce, and maintain a positive employer brand.

- Relevance [23]: The content should be directly related to its intended topic or subject, meeting the needs and expectations of the target audience for online professional platforms. For instance, the post's content should be relevant to the professional platform to gain value and visibility over the platform.

- Clarity [23]: The content should be well-organized, easy to understand, and effectively communicate its primary objective. For instance, the questions should be straightforward (what do you want to ask) before posting on any knowledge-sharing platform.

Maintaining high content quality is essential for building customer trust, establishing authority, and encouraging long-term interactions. Furthermore, this can positively impact search engine rankings, question answering, user engagement, and recommendations for businesses and online platforms. All the above-mentioned aspects focus on enhancing the overall quality of professional content with respect to the content point of view. We focus on these dimensions from a use-case point of view to improve the quality of downstream tasks in domain-specific scenarios. In the following section, we discuss some preliminaries commonly used to enhance content quality.

2.3 Preliminaries

This section discusses domain-specific knowledge extraction from unstructured text, tools and techniques including graph-based approaches and knowledge representation learning. We also explore their applications on large-scale social collaborative platforms such as Stack Overflow [13], CarrerBuilder [66], and Wikipedia [71].

Domain-specific knowledge for online professional platforms includes job roles, responsibilities, industry-specific terms, relevant skills, qualifications, job postings, resumes, and attributes valued by employers in specific industries [66]. Extracting domain-specific information is crucial in tailoring solutions to the unique needs of different industries and their real-world applications.

Since most of the content on these platforms is user-generated, the first class of approaches uses Open Information Extraction Techniques [3] without an ontology to extract information from unstructured text. Schmitz et al. [72] focuses on automatically identifying relation phrases from the text. Abebe et al. [73] proposed a semi-automated approach to construct domain concept ontology from source code identifiers. Some works extract fixed relations using Information Extraction (IE) techniques. Pujara et al. [74] use IE techniques to identify candidate facts that include a set and attributes of these entities and the relationships between these entities. Mausam et al. [3, 75] proposes different Open Information Extraction techniques to extract triples from unstructured text. Some IE tools such as StuffIE [76], include semantic tagging of unlabeled facets using fine-grained information extraction. This extracted knowledge can be organized by structuring information within a specific domain using knowledge graphs [74]. Knowledge graphs (KGs), a popular type of heterogeneous graph, organize and represent knowledge to model complex relationships, enabling advanced querying and reasoning [77]. These are structured ways to store facts as subject, predicate, and object. Another class of work includes the construction and utilization of Knowledge Graphs [29] to solve any challenging and well-researched problem in the generic domain. Most works focus on the construction of Open Domain Knowledge graphs such as Wikipedia [71], Freebase [38], NELL [78], YAGO [34], and DBpedia [36]. Research works like T2KG [79], LODifier [80] focused on constructing KG from unstructured data by leveraging information of predefined knowledge bases like DBpedia [36]. Microsoft Academic graph [81] and AceKG [82] focused on building knowledge graph of academic papers. Once the structured data is available, the objective is to effectively represent and extract meaningful insights from it. Now, we discuss a range of methods used to learn representations from structured domain-specific knowledge.

Graph representation learning refers to the process of learning embeddings or vector

representations for the components of a graph, such as nodes, edges, or entire subgraphs. Techniques for graph representation learning include embedding methods like Node2vec [83] and DeepWalk [84], Graph Neural Networks (GNNs), and knowledge graph embedding approaches such as TransE [85], TransH [86], and RotatE [87].
Traditional embedding methods [83, 84] calculate embeddings from each node to capture the graph's structure. They primarily capture local neighborhood information, potentially missing out on global structural properties of the graph. Later, graph-based approaches like LINE [88] and SDNE [89] were used for first and second-order structural information of the graph. These embeddings were then used for training supervised algorithms for node classification tasks. These embedding creation algorithms only considered the topology of the graph and did not consider the properties of individual nodes due to which the expressivity of the models was impacted. Generating embeddings in real-time is difficult since it would mean retraining on the whole graph.
This problem was solved by the use of Graph Convolutional networks [90], which considered graph topology as well as the node properties. Through operating deep learning-based methods in the graph domain, graph neural networks [91] are expected to demonstrate substantial expressive power and learning capability for the content/applications that can be represented as graphs.
Graph Neural Networks (GNNs) have emerged as a powerful tool for analyzing graph-structured data [92]. They are designed to leverage the rich relational information inherent in graphs, making them particularly effective for domain-specific applications where relationships and interactions between entities are crucial. GNNs extend the capabilities of traditional neural networks to graph data structures. The core idea is to perform convolution operations over the nodes and edges of a graph, aggregating information from a node's neighborhood to update its representations.

- Node Embedding Initialization: Each node in the graph is initially represented by a feature vector, which can include attributes such as textual, numerical, or categorical data. Each node v in the graph is initially represented by a feature vector $h_v^{(0)}$.

- Neighborhood Aggregation: Nodes update their representations by aggregating feature information from their neighbors. This process can be formalized as Equation 2.1.

$$h_v^{(k+1)} = \sigma\left(W^{(k)} \cdot \text{AGGREGATE}\left(\{h_u^{(k)} : u \in \mathcal{N}(v)\}\right)\right) \tag{2.1}$$

  where $h_v^{(k)}$ is the feature vector of node v at layer k, $\mathcal{N}(v)$ denotes the neighbors of node v, $W^{(k)}$ is the learnable weight matrix at layer k, σ is a non-linear activation function.

- Aggregation Function: After multiple layers of neighborhood aggregation, an aggregation function is used to produce a fixed-size representation of the entire graph or specific nodes. This can be done through global pooling operations as described in Equation 2.2.

$$\text{Aggregation} = \sum_{v \in \mathcal{V}} h_v^{(K)} \tag{2.2}$$

where $\mathcal{V}$ denotes the set of all nodes in the graph, and K is the final layer.
After the introduction of Graph Neural Networks (GNNs), several GNN architectures have gained popularity due to their unique approaches and effectiveness in various applications. Among the most common graph neural network architectures are GraphSAGE [93], GraphSAINT [94], and GIN [95].
Graph Sample and Aggregation: GraphSAGE [93] proposed the computation of node representations inductively (previously unseen data) by sampling and by recursively aggregating over fixed-sized neighbourhoods on large graphs (Equation 2.3).

$$h_v^{(k+1)} = \sigma\left(W \cdot \text{AGGREGATE}\left(\{h_v^{(k)}\} \cup \{h_u^{(k)}, \forall u \in \mathcal{N}(v)\}\right)\right) \tag{2.3}$$

where $h_v^{(k)}$ is the feature vector of node v at layer k, $\mathcal{N}(v)$ denotes the neighbors of node v, W is a learnable weight matrix, σ is a non-linear activation function, AGGREGATE is a function that aggregates the feature vectors of a node and its neighbors.
Graph Sampling-Based Inductive Learning Method: GraphSAINT [94] enhances scalability and efficiency by using subgraph sampling techniques to train the GNN on large-scale graphs while maintaining high performance. The core idea involves sampling a subgraph and performing aggregation within this subgraph. The Equation 2.4 can be represented as:

$$h_v^{(k+1)} = \sigma\left(W \cdot \text{AGGREGATE}\left(\{h_u^{(k)}, \forall u \in \mathcal{N}(v) \cap \mathcal{S}\}\right)\right) \tag{2.4}$$

where $\mathcal{S}$ denotes the sampled subgraph.
Graph Isomorphism Network (GIN): Keyulu et al. [95] provided theoretical foundations for GNNs based on the Weisfeiler Lehman (WL) test and proposed the Graph Isomorphism Network (GIN) with discriminative power equal to that of the WL test. GIN ensures maximal discriminative power by distinguishing between different graph structures with greater precision through its injective aggregation function (Equation 2.5).

$$h_v^{(k+1)} = \text{MLP}^{(k)}\left((1+\epsilon) \cdot h_v^{(k)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k)}\right) \tag{2.5}$$

where $\text{MLP}^{(k)}$ is a multi-layer perceptron at layer k, ϵ is a learnable or fixed scalar.

While GNNs provide a broader framework for learning representations from various types of graph-structured data, another class of **knowledge representation learning methods** [91] primarily focuses on encoding semantic relationships in knowledge graphs. It involves automatically learning representations (embeddings) of entities and relationships in a knowledge graph or a structured knowledge base. Such representations [91] in knowledge graphs are widely used for **fact checking** applications [96, 97]. The set of approaches finds streams in knowledge graphs to support fact checking [98]. Pan et al. [99] suggests that translational knowledge graph representation methods such as TransE [85], TransH [86] are used to predict the plausibility of the facts using external KGs (DBpedia [36], Freebase [38]) for fake news detection. Some works [100] leverage unstructured and structured sources [101] for automatic fact-checking. In all domain-specific applications, improving content quality involves prioritizing completeness, accuracy, consistency, relevance, and clarity. These factors are essential for maintaining credibility and reliability across professional platforms. For consistency, entity normalization [66] is a critical process in data management and analysis, particularly for organizations like CompanyDepot that rely on consistent data for many downstream tasks. For clarity and relevance aspects, Baltadzhieva et al. [56, 102] studied and analyzed the linguistic terms and user's behaviour in guiding the feature engineering process to determine the question's quality. Research works [103–106] identify redundant (duplicate) questions and estimate the relative difficulty of questions to enhance user experience over **Question Answering forums**. Roy et al. [107] contributes with a multi-class classification of 'closed' questions along with 'closing' reasons for questions. Jan et al. [108] focused on identifying unclear questions as they have low quality. For completeness, Khaouja et al. [109] identify the skill bases used for analyzing the job market, the type of extracted skills, the skill identification methods, the studied sector and their granularity. Tong et al. [110] build a job-skill network to measure the popularity of the skills by exploring a large corpus of job postings. Generative models [111] use a multi-attention deep neural network model and skill graph to generate job postings. Bhola et al. [112] employs an Extreme Multi-label Classification method that utilizes the Transformer model to predict the missing skills from a textual job description. Despite existing efforts, the evolving landscape of user-generated content necessitates further exploration of domain-specific approaches to fill research gaps that will enhance content quality.

2.4 Research Gaps

- Domain-specific information is multi-faceted and contains heterogeneous information such as skills, experience, designations, companies, and job type, which need to have associated knowledge of the polarities and context of the entities, which are still unexplored in the content quality literature.
- Existing frameworks for knowledge graph construction are specific to general concepts from day-to-day life and lack domain-specific knowledge. Despite the utility of KGs, most of the existing KBs such as Freebase [38], NELL [78], YAGO [34], and DBpedia [36] provide limited information about domain-specific facts, essential entities such as evolving skills, designations, and hidden properties of the job such as type of recruiter, shift timings, interview dates, etc.
- Most fact-checking methods rely on experts, such as journalists or scientists, to assess the content and the crowd's wisdom [113]. Expert-based methods are expensive as they need the hiring of experts, are limited in number, and may not treat all the content being produced.
- The side information used in canonicalization literature [63] is rudimentary and lack domain-specific information. These methods have little or no information about external sources such as Wikipedia Infobox [71] and Google search API [29], providing additional knowledge for noisy entities. Some methods fail to generalize and have a limited understanding of more complex and emerging entities.
- Most of the state-of-the-art entity canonicalization approaches [39, 63, 64] achieve below-par performance for real-world domain-specific datasets (company, skills, designation, institutes) due to noise, sparseness, and context-sensitive information in the user-generated text.
- Despite significant efforts, graph-based approaches and collaborative learning are still relatively unexplored for addressing issues related to completeness (missing skills).

Chapter 3

Canonicalization (Standardization) of entities in online professional content

"We are only as strong as we are united, as weak as we are divided."

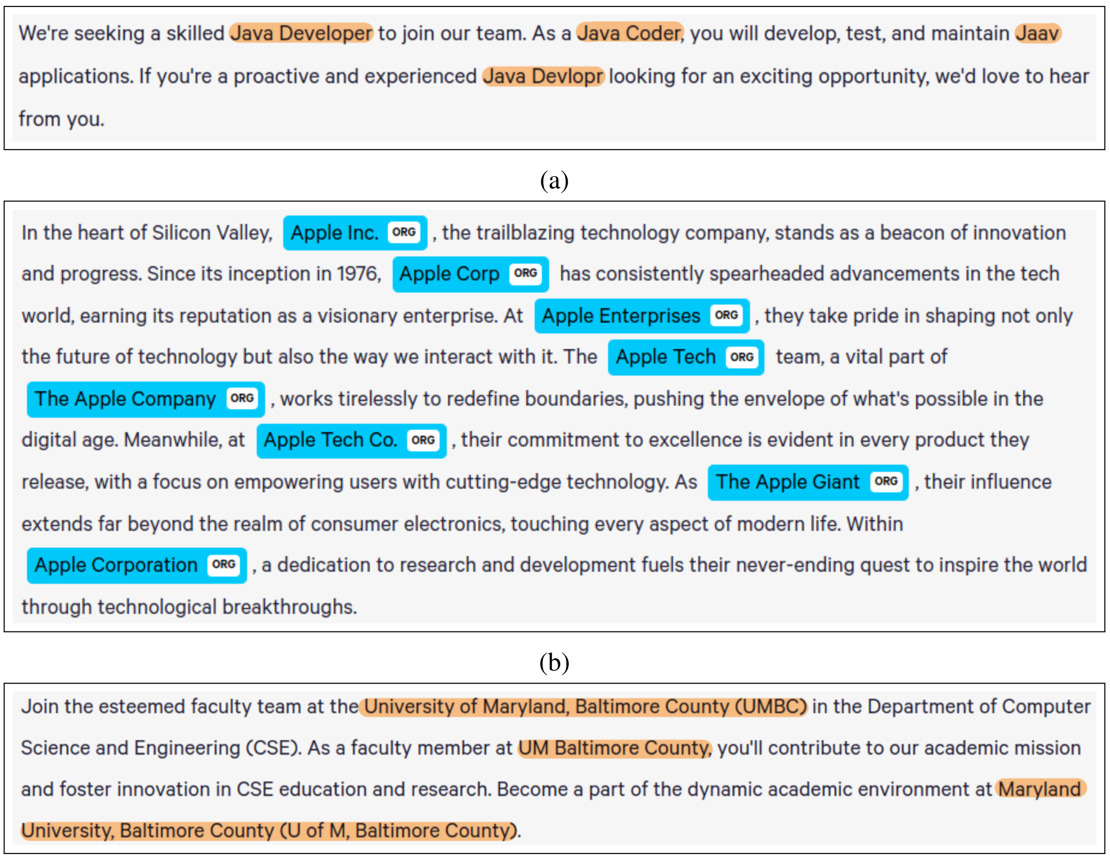
-J. K. Rowling, 'Harry Potter and the Goblet of Fire'

This chapter focuses on how to standardize or normalize the content (such as skills, institutes, companies, and designations) on online professional platforms. In this[1] chapter, we present our first solution to deal with noise, inconsistencies, and ambiguities in abundance of user-generated content found in online job postings, candidate profiles, and company profiles. Figure 3.1 demonstrates many diverse variations of these entities (company) present in documents such as company profiles.

3.1 Introduction

Online recruitment platforms have abundant user-generated content in the form of job postings, company and candidate profiles. This content includes diverse recruitment domain entities such as company names, skills, institutes, and designations that become part of the knowledge base. As the content is user-generated, prevalent ambiguities [2], prolific variations [114], noisy, and redundant entities also come up in the KB. Thus, employing these domain-specific entities directly for downstream applications such as question answering [115], job search [19], and recommendations [60] results in poor system performance. Therefore, canonicalization of the entities, i.e., mapping various references of a unique entity into a representative cluster is im-

[1] Work presented in this chapter has been accepted in ICANN 2021. **Goyal, N.**, Sachdeva, N., Goel, A., Kalra, J., and Kumaraguru, P. KCNet: Kernel-based Canonicalization Network for entities in Recruitment Domain. In 30th International Conference on Artificial Neural Networks (ICANN). 2021.



We're seeking a skilled Java Developer to join our team. As a Java Coder, you will develop, test, and maintain Jaav applications. If you're a proactive and experienced Java Devlopr looking for an exciting opportunity, we'd love to hear from you.

(a)

In the heart of Silicon Valley, Apple Inc. ORG , the trailblazing technology company, stands as a beacon of innovation and progress. Since its inception in 1976, Apple Corp ORG has consistently spearheaded advancements in the tech world, earning its reputation as a visionary enterprise. At Apple Enterprises ORG , they take pride in shaping not only the future of technology but also the way we interact with it. The Apple Tech ORG team, a vital part of The Apple Company ORG , works tirelessly to redefine boundaries, pushing the envelope of what's possible in the digital age. Meanwhile, at Apple Tech Co. ORG , their commitment to excellence is evident in every product they release, with a focus on empowering users with cutting-edge technology. As The Apple Giant ORG , their influence extends far beyond the realm of consumer electronics, touching every aspect of modern life. Within Apple Corporation ORG , a dedication to research and development fuels their never-ending quest to inspire the world through technological breakthroughs.

(b)

Join the esteemed faculty team at the University of Maryland, Baltimore County (UMBC) in the Department of Computer Science and Engineering (CSE). As a faculty member at UM Baltimore County, you'll contribute to our academic mission and foster innovation in CSE education and research. Become a part of the dynamic academic environment at Maryland University, Baltimore County (U of M, Baltimore County).

(c)

Figure 3.1: Various challenges included in the canonicalization of the same entity, represented in diverse forms. a) An entity *'Java developer'* having different variations due to misspelling. b) 'Apple Corp' variations present in a company profile description. c) UMBC can have different variations, including abbreviations, that can lead it to be identified as different entities.

perative for recruitment platforms. Canonicalizing named entities involves various challenges (Figure 3.1), including spelling mistakes and variations (*java developer* & *java deveoper*), overlapping but different entities (*Emerald Bikes pvt limited* & *Emerald Jewellery Retail Limited*), hierarchical variations (*Oracle Financial Services Software* & *Oracle Corporation*), domain-specific concepts (*SOAP* & *REST*), short forms (*umbc* & *University of Maryland, Baltimore*), and semantically similar variations (*Accel Frontline* & *Inspirisys*).

Canonicalizing semantically similar and domain-specific entities can be difficult, even with human experts, because it demands domain-specific knowledge rather than depending solely on information associated with entity variations. For instance, *'Accel Frontline'* and *'Inspirisys'* are different semantic variations of the same entity which is challenging to canonicalize without any additional context in form of side information (the knowledge associated with an entity such as location, website, etc.)

Vashishth et al. [63] showed the use of external side information (morphological, IDF token overlap, PPDB [116]), however, it is rudimentary and has limited utility in domain-specific settings. Another challenge arises from the fact that these entities, which already include location-specific contexts such as *'MIT India'*, *'MIT'* or *'Madras Institute of Technology'*, often originate from heterogeneous sources such as job postings, CVs, and candidate profiles, leading to varying degrees of noise in their representations. In such cases, side information from publicly available sources such as Wikipedia [71], WordNet [117], Google Knowledge Graph [29] in the form of affiliations or textual descriptions would significantly aid in establishing canonical forms for these entities. For instance, the side information of MIT India from Google KG is*'Madras Institute of Technology is an engineering institute located in Chromepet, Chennai, India. It is one of the four autonomous constituent colleges of Anna University.'* This Side Information can complement the entities by providing additional context related to domain-specific entities. Therefore, we leverage side information from external sources [29, 36, 71] to improve the entity canonicalization task (Side Information Section 3.6.2).

Previous approaches [39, 66] focus on statistical methods that use handcrafted features for entity canonicalization. Fatma et al. [2] employ a deep learning method that overcomes the challenges of statistical methods by eliminating the need for explicit feature engineering and using character-based word embeddings for unknown and emerging entities. Le et al. [118] shows that deep learning methods often minimize training errors but fail to generalize. One important family of algorithms [118–120] that implicitly explores a much larger feature space via computing a similarity (or kernel) function between entities. These approaches suggest introducing learnable kernels can effectively capture intricate patterns in data, potentially requiring fewer parameters compared to deep learning models designed for similarity tasks and improving generalizability.

We study a kernel-based neural network (Section 3.5.1) designed for entity canonicalization in the recruitment domain and enhance the performance using side-information which is underexplored in the literature. This work proposes a novel multi-tier framework using a learnable kernel network [118, 120], which implicitly maps the data into high-dimensional feature space. Our framework captures the non-linear mapping between contextual, meta, and semantic representations through a learning objective to output the pairwise similarity between recruitment domain entities. Furthermore, we generate the canonicalized clusters for each entity. We demonstrate and validate the efficacy of our approach on proprietary as well as open source datasets, including DBpedia and ESCO [121, 122] for the generalizability of our solution.

3.2 Related Work

KB Canonicalization: Galarraga et al. [39] uses manually defined feature spaces to perform the canonicalization task. This approach encodes limited similarity between different semantic representations. Hence, it results in a degradation of performance for real-time applications. Vashishth et al. [63] jointly handle noun and relation phrases using knowledge graph embedding models [123] by optimizing its objective function along with using information from external sources called *'side information'*. However, these state-of-the-art knowledge graph embedding methods [85] achieve below-par performance for real-world recruitment domain datasets due to noise, sparseness [64], and context-sensitive information present in triples. Additionally, the side information methods used in literature [63] are rudimentary and lack domain-specific information. Considering these limitations, we leverage external knowledge sources such as Wikipedia Infobox and Google search API, which provide additional knowledge for noisy entities.

Domain-Specific methods: Despite the importance of named-entity canonicalization in the recruitment domain, only a few recent studies have explored the problem for unique domain challenges [35, 66]. Yan et al. [24] propose a company name normalization system that employs LinkedIn social graphs and a binary classification approach. In this work, the authors use complete profile information as the context. However, this information will be hard for new and emerging entities. Lin et al. [114] use side information and learn domain knowledge from the source text based on the type of entities. Popular state-of-the-art entity linking tools [124] are probabilistic and requires sufficient contextual information to connect to candidate entity and perform well when standard surface forms are available. For example, recruitment domain-specific documents may contain *'Python'* or *'Python Programming'* while the former is linked to a different type of entity with a high confidence score using these tools. Similarly, Fatma et al. [2] utilizes word and character-based representations based similarity model along with the attention mechanism to cluster similar entities. However, these works fail to generalize and

have a limited understanding of more complex and emerging entities.
Kernel-based architectures: Kernel methods have proven effective in exploring larger feature space implicitly in deep learning architectures [125]. Customized kernel [126] based deep learning architectures enhance the performance of the model and map data to an optimized high-level feature space where data may have desirable features toward the application. Recent works utilize deep embedding kernel architectures for identity detection, transfer learning, classification and other tasks [118]. We use kernel-infused neural networks to capture the latent semantic relationships and non-linearity between different pairs of entities in KBs. These kernel methods are robust for collaboration with neural networks and less expensive than training deep learning architectures.
Clustering methods: Research works have used various clustering techniques for the canonicalization task. Among these methods, Hierarchical Agglomerative Clustering (HAC) is the most extensively used in the literature [2, 39, 63].
Our research is uniquely placed at the intersection of the vast literature on kernel-based neural network learning and clustering approaches for the canonicalization of domain-specific KBs.

3.3 Contributions

The contributions of this work are:

- We propose a Kernel-based Canonicalization Network (KCNet), which induces a non-linear mapping between the contextual vector representations while capturing fine-granular and high-dimensional relationships among vectors. To the best of our knowledge, this is the first approach towards exploring kernel features for canonicalizing Knowledge Base entities in the recruitment domain.

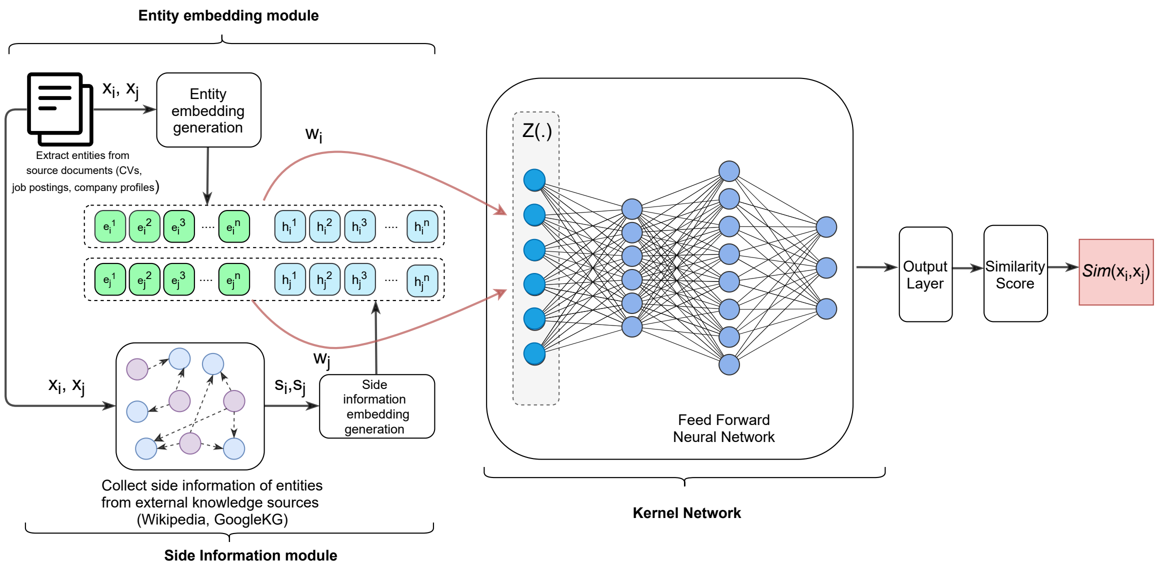
- KCNet efficiently models more prosperous semantic and meta-side information from external knowledge sources to canonicalize domain-specific entities.

- We perform extensive experiments on real-world proprietary and publicly available datasets in the recruitment domain to show the effectiveness of our proposed approach as compared to baselines.

3.4 Problem Definition

Consider $\mathcal{E}$ as the set of entities extracted from job postings, CVs, and company profiles (Entity Extraction process is discussed in Chapter 5, Section 5.3.3). For each entity x_i, we consider its side information $s_i \in \mathcal{S}\ \forall x_i \in \mathcal{E}$ acquired from heterogeneous sources (elaborated in detail in section 3.6.2).

Given a pair of entities x_1 and x_2 and their corresponding side information s_1 and s_2 where $x_1, x_2 \in \mathcal{E}$ and $s_1, s_2 \in \mathcal{S}$, the main objective is to find a function $f(x_1, s_1, x_2, s_2) \rightarrow sim(x_1, x_2)$. A pairwise similarity matrix ($\mathcal{M}_{sim}$) is formed by applying f over the set of all entity pairs and then a clustering algorithm is used to form unique canonical clusters of similar entities.



Entity embedding module
x_i, x_j
Entity embedding generation
Extract entities from source documents (CVs, job postings, company profiles)
w_i
e_i^1 e_i^2 e_i^3 .... e_i^n h_i^1 h_i^2 h_i^3 .... h_i^n
e_j^1 e_j^2 e_j^3 .... e_j^n h_j^1 h_j^2 h_j^3 .... h_j^n
w_j
x_i, x_j
s_i, s_j
Side information embedding generation
Collect side information of entities from external knowledge sources (Wikipedia, GoogleKG)
Side Information module
Z(.)
Feed Forward Neural Network
Kernel Network
Output Layer
Similarity Score
$Sim(x_i, x_j)$

Figure 3.2: Kernel-based Canonicalization Network for entities in recruitment domain. We first extract entities and combine it with side information. The combination (concat) is passed through the Kernel network. The output is a pairwise similarity matrix.

3.5 Kernel-based Canonicalization Network (KCNet)

In this section, we introduce the proposed KCNet approach. We elaborate on the problem definition and each component of our network architecture in detail.

3.5.1 Network Architecture

We propose a multi-tier novel architecture consisting of three modules: Entity embedding generation, Side Information embedding generation, and Kernel network. We apply the clustering technique after obtaining output on our proposed architecture. Fig. 3.2 shows the overall architecture of our proposed approach (KCNet).

- **Entity embedding generation.** We obtain an m-dimensional (m=768) vector for each entity pair (x_i, x_j) producing (e_i, e_j) in the space $C \in \mathcal{R}^m$. We use a pre-trained distilled version (fewer parameters, less space and time complexity) of S-BERT [127] to generate initial contextual† [2] embeddings for all entities.

- **Side Information embedding generation.** We represent (h_i, h_j) as n-dimensional vector (n=768) for the side information acquired for each entity pair (x_i, x_j) in the space $H \in \mathcal{R}^n$. The formation of side information vector is described in Section 3.6.2. These representations h_i and h_j are concatenated with the corresponding entity representations e_i and e_j to obtain infused vector representations w_i and w_j for the input pair (x_i, x_j). Here, $w_i = e_i \odot h_i$ and $w_j = e_j \odot h_j$.
  Note that, $\odot$ is the concatenation function of two vectors producing (w_i,w_j) in the space $W \in \mathcal{R}^{m+n}$. The $(m+n)$ -dimensional vector representation is fed into the kernel network to learn the similarity function, $sim(x_i,x_j)$.

- **Kernel Network.** We introduce a kernel network to learn the similarity function f to model complex relationships between the data representations of input pairs in an optimized space. The input to this network is denoted as Z, formed by the combined representation w in the equation (1).

  $$Z = (w_i \circ w_j) \odot \left| w_i - w_j \right| \tag{3.1}$$

  Here, $\circ$ is a Hadamard (element-wise) product which exploits interactions between two vectors at each dimension. We also determine the $\mathcal{L}_1$ distance for each dimension w_i and w_j and concatenate the interactions of both the components as shown in equation (2).

  $$Z = \left\{ w_i^1 * w_j^1, \dots, w_i^{m+n} * w_j^{m+n}, |w_i^1 - w_j^1|, \dots, |w_i^{m+n} - w_j^{m+n}| \right\} \tag{3.2}$$

  where w_i^k represents the k^{th} dimension of w_i. The dimensionality of Z is $2 * (m+n)$. The kernel function takes into account both element-wise product (design of polynomial

[2]† specifies that an entity name such as '*University of Maryland, Baltimore*' contains the location-specific context, i.e., '*Baltimore*'. The representation of the entire entity is termed as contextual embedding.

kernel) and differences (design of RBF kernel) over each dimension of the original entity. This configuration of inputs allows the network to learn a non-linear relationship between the input pairs and symmetric representations at a fine granular level over each input dimension. Therefore, the learned kernel can map a more robust similarity function over the input space compared to traditional methods such as RBF and polynomial [126]. Similar observations for the customized kernel have been made in [118]. The newly obtained vector Z captures the latent semantic relationships between the two input entities. This vector is fed into a multi-layer feed-forward neural network with *sigmoid* output, facilitating the learning of a highly non-linear mapping f to predict similarity over entity pairs. The size of hidden layers (number of neurons) in the kernel network is chosen using a hyperparameter k. We define $k = \alpha * d$ where d is the dimensionality of Z and typically, $\alpha = \left\{1,2,3\right\}$, say, $f(x_i, x_j) = f(x_j, x_i) > 0$. The kernel network outputs the probability that input pair (x_i, x_j) belong to the same cluster. Therefore,

$$f(x_i, x_j) = P(y^i = y^j | x_i, x_j) \tag{3.3}$$

- **Clustering using pairwise similarity scores.** We generate a pairwise similarity matrix $\mathcal{M}_{sim}$ using the pairwise similarity scores obtained from the Kernel Network for all the entity pairs (x_i, x_j). For every entity pair in a dataset (for instance, institute–institute pairs have pairwise similarity scores in similarity matrix). Once the pairwise similarity matrix $\mathcal{M}_{sim}$ is obtained, we apply Hierarchical Agglomerative Clustering (HAC) to form a unique cluster of entities. HAC is one of the widely preferred clustering techniques used in literature [2, 39, 63] for canonicalization tasks. Each entity is finally mapped to a unique cluster. We choose the number of clusters k using the silhouette index [128]. We repeat the same process for all the datasets (skills, designations, institutes, companies).

3.6 Datasets

This section presents a comprehensive overview of the datasets and the side information details to enhance the understanding and context of the data.

3.6.1 Dataset Description

Proprietary Datasets. We use real-world datasets from one of the largest recruitment platforms in India. The dataset, i.e., Recruitment Domain Entities (RDE), consists of company clusters (RDE (C)), institute clusters (RDE (I)), skill clusters, and designation clusters (RDE

(D)). Table 3.1 reports the statistics (ground truth clusters and side information) of four proprietary and three public datasets. The ground truth clusters are manually annotated by domain experts [2] with a kappa agreement of 0.83. Next, we generate the variation pairs- positive and negative samples. Each name variation of entity $e_x \in \mathcal{E}$ is defined as $\left\{e_x^1, e_x^2, e_x^3, \ldots e_x^n\right\}$, which belong to same annotated cluster. We remove non-ASCII characters to form a set of all unique name variations of e_x. For each entity pair, (e_x^i, e_y^j), training data is prepared using the mapping function g, such that, $g(e_x^i, e_y^j) = 1, \forall(x, y)$ where $i \neq j$ and $x = y$ belongs to same annotated cluster (positive pairs). Similarly, $g(e_x^i, e_y^j) = 0$ where $x \neq y$ belongs to different clusters (negative pairs) using a random sampling approach [129].

Open Datasets. We test the effectiveness of our framework (KCNet) using open-source datasets i.e., DBpedia (C) and ESCO. DBpedia (C) [2] dataset is prepared by querying DBpedia for company names to extract *dbo:Company*. ESCO [122] i.e., ESCO (S) and ESCO (D) are open-source recruitment domain datasets for ESCO-skills and ESCO-designations. Fatma et al. [2] prepared and released [3] these datasets for the research community.

Dataset	**Ground truth (Clusters)**	**Entity variation pairs**	**Side Information (Wikipedia & Google KG)**
		Open Datasets	
DBpedia (C)	2,944	22,829	100%
ESCO (D)	2,903	62,969	80%
ESCO (S)	2,644	35,554	73%
		Proprietary Datasets	
RDE (C)	25,602	13,31,814	†
RDE (I)	23,690	12,88,090	†
RDE (D)	3,894	32,114	†
RDE (S)	607	3,197	†

Table 3.1: The first three rows shows the dataset statistics for Open Datasets (DBpedia (C), ESCO (S), ESCO (D)) and the last four rows are Proprietary Datasets (RDE (C), RDE (I), RDE (S), RDE (D)). A † sign denotes that the information was redacted for proprietary datasets.

3.6.2 Side Information

We leverage two sources for side information extraction, *Wikipedia Infoboxes* and *Google KG*.

[3] https://github.com/anonymous-warrior/EntityCanonicalizationDataset.

- **Wikipedia Infobox.** We query Wikipedia using its advanced search options[4] and extracted knowledge from Wikipedia infoboxes for different datasets. While querying the advanced search using phrases, we include the entity types such as institute, skill, designation, or company along with their variations to aid in disambiguating information. If two variations are linked to the same Wikipedia entity, we assume them to be equivalent as per this information. For instance, *'GIT'* and *'Gandhinagar Institute of Technology'* can be linked to the same Wikipedia entity. Wikipedia [71] contains heterogeneous side information (provided below) about different entities and sometimes an empty list in case the information is unavailable for that particular entity.
  {*'title_wikis', 'websites', 'types'*} for RDE (S)
  {*'Names', 'websites', 'title_wikis'*} for RDE (D)
  {*'title_wikis', 'websites', 'affiliation'*} for RDE (I)
  {*'Names', 'websites', 'title_wikis','types'}* for ESCO (S)
  'Names', 'websites', {'title_wikis'} for ESCO (D)
  {*'types', 'industries', 'websites', 'native names', 'title_wikis'*} for DBpedia (C).

- **Google KG.** For some entities with short forms, noisy variations, etc. we are unable to fetch knowledge using Wikipedia search. Therefore, we leverage Google KG Search API [5] to extract supplementary information for each entity. For instance, the Google API response for an institute entity *'loreto college canada'* yields the attributes such as {*title*, *type*, *description* ,*website*, *local map*, *image*, *map*, *latitude*, and *longitude*.} from knowledge graph. Among these attributes, we found that the rich semantic textual *description* and *type* attributes of entities is available for 90% of them. Therefore, we utilize this attribute as side information from Google KG to supplement the model with semantic knowledge.

Finally, we combine the side information extracted from Google KG and Wikipedia infoboxes. For example, an institute entity *'loreto college canada'* and its side information is defined as *'title_wikis': 'Loretto College School',* ; *'description': 'Loretto College School, formerly the Loretto Abbey Day School and Loretto Abbey Day School and College, is a Catholic high school for girls in Toronto, Ontario, Canada. The school is operated by the Toronto Catholic District School Board, formerly the Metropolitan Separate School Board.'* ;*'website':'tcdsb.org/o/lorettocollege'*;*'affiliations':'Roman Catholic (Loretto Sisters)*'; *'type':'School in Toronto, Ontario'*.
Using this information, we create side information embeddings s_i, a concatenated sequence of side information vector representations $\{s_i^1, s_i^2, \dots s_i^p\}$, where p is the number of attributes

[4]https://www.mediawiki.org/wiki/MediaWiki
[5]https://serpapi.com/

obtained from external sources. The advantage of employing this approach (concatenated sequence) is that, despite the heterogeneity of side information and its variation based on search location and other attributes, our approach supports the flexible integration of this information into the model, when the number of attributes changes. Finally, we generate the side information embeddings using a pre-trained distilled version of the S-BERT [127] model. We follow the same process across all the entity types. The given list can be further extended based on the availability of other side information.

3.7 Experimental Setup

This section describes our baselines, model configurations, and evaluation metrics.

3.7.1 Baselines

We compare our approach of finding pairwise similarity using KCNet with the following baseline approaches:

- **Galarraga-IDF** [39]: The approach suggests that when two noun phrases share a common word, they are often perceived as similar unless that word is widely shared among many other mentions. For instance, in the context of company name canonicalization, the similarity between *"Tata Consultancy Services"* and *"Infosys Consulting Services"* due to the shared word *"Services"* may not indicate significant differentiation. This is because "Services" is commonly found in various company names, such as *"Accenture Services"* or *"IBM Global Services"*, which are distinct entities despite sharing the same word. We use a weighted word overlap-based similarity measure proposed by Galarraga [39] in which a word is given more importance if it appears in fewer mentions.

- **Entity embeddings (Distilled S-BERT(*))+cosine:** We generate our entity embeddings using a pre-trained model (Distilled S-BERT [130]) using the methodology described in Section 3.5.1 to obtain the vector representation for the entity pair (x_i, x_j). Subsequently, we find a pairwise similarity matrix by applying cosine similarity measures on these vector representations obtained for every entity pair (x_i, x_j).

- **Entity and Side Information embeddings (Distilled S-BERT(**)) +cosine:** We obtain entity embedding of (x_i, x_j) and side information embedding of (s_i, s_j) to get combined (w_i, w_j) using the methodology described in Section 3.5.1. Subsequently, we find a pairwise similarity matrix by applying cosine similarity measures on these vector representations obtained for every entity pair (x_i, x_j).

- **Char-BiLSTM+A [2]:** The approach utilizes a siamese network that takes characters [2] as input and passes it through the pair of BiLSTM layers enhanced by the attention layer. To accomplish this, the authors first standardize all name variations to lowercase and remove rarely occurring non-informative characters (e.g., '*'). Following pre-processing, the character vocabulary size for company variations is 64. The character input vector is a concatenated sequence of 64-dimensional one-hot vectors for each character. Each entity variation with a character vocabulary of 64 is represented by a 43 x 64-dimensional character input vector (authors limit the characters to 43 for each entity). This sub-network comprises two Bi-LSTM layers. This character input vector, as detailed in [2] is fed into these LSTM layers and followed by sigmoid activation on the output layer, representing the similarity score between entities.

- **Word-BiLSTM+A [2]:** The previous baseline (Char-BiLSTM+A) is modified by replacing character-based representations with word-based representations followed by attention layer. The word input vector for an entity name consists of a concatenated sequence of 100-dimensional word embeddings using FastText [131] for each word token. Authors limit this to the first 7 words with padding, resulting in a 7 x 100-dimensional word input vector. The word input vectors are fed into Bi-LSTM layers and followed by sigmoid activation on the output layer, representing the similarity score between entities.

- **Char-BiLSTM+A+Word+A [2]:** This approach learns a siamese model (SM) architecture for Char Bi-LSTM + A+ Word + A. Character level representation learned from Bi-LSTM followed by attention layer (Char-BiLSTM+A) and word level representation learned from Bi-LSTM layer followed by attention layer (Word-BiLSTM+A). The architecture details are described by authors in [2].

3.7.2 Model Configurations

We learn pairwise similarity models using the proposed architecture for different datasets (companies, institutes, designations, skills). The training and testing dataset split is taken as (80, 20). The optimal value of hyperparameters (*size of hidden layer,* α) for companies, designations, and skills is (1536, 2), whereas for institutes, (768, 2). Batch-size is 512, and the number of fully connected layers is 3. Rectified linear units (*ReLU*) are used as activation functions, and the dropout rate is 0.3. *Binary cross-entropy loss* and *Adam* as an optimizer is used to train the kernel network and learn the pairwise similarity matrix ($\mathcal{M}_{sim}$).

3.8 Evaluation metrics

Following [2, 39, 63], we use evaluation metrics (Macro, Micro) across Entity Canonicalization methods. We evaluate clusters using Macro Precision ($\mathrm{P_{macro}}$), Macro Recall ($\mathrm{R_{macro}}$), Micro Precision ($\mathrm{P_{micro}}$) and Micro Recall ($\mathrm{R_{micro}}$) For all evaluations, F1-score is defined as the harmonic average of precision and recall. To demonstrate the evaluation of clusters, $\mathcal{GE}$ denotes the set of ground-truth clusters, and $\mathcal{PE}$ denotes the set of predicted clusters by HAC algorithm [2]. Table 3.2 shows the sample of ground truth/gold standard clusters ($\mathcal{GE}$) and predicted clusters ($\mathcal{PE}$) to explain the evaluation metrics for clustering. We use Precision (P), Recall (R), and F1-scores (F) [132] for pairwise similarity results (Table 3.3 and Table 3.4).

- **Macro Precision ($\mathbf{P_{macro}}$)** is defined as the fraction of pure clusters in $\mathcal{PE}$, i.e., clusters in which all the entities are linked to the same gold entity (Equation 3.4).

$$P_{\mathrm{macro}}(\mathcal{PE},\mathcal{GE}) = \frac{|\mathcal{PE}_{\mathrm{i}} \in \mathcal{PE} : \exists \mathcal{GE}_{\mathrm{i}} \in \mathcal{GE} : \mathcal{GE}_{\mathrm{i}} \supseteq \mathcal{PE}_{\mathrm{i}}|}{|\mathcal{PE}|} \tag{3.4}$$

  Explanation: Macro Precision evaluates the extent to which predicted clusters are pure, indicating that all mentions within a cluster refer consistently to the same gold entity. In the example, two out of three predicted clusters are pure (i.e., if it is fully contained in any ground truth cluster) ($PE2$, $PE3$), so $P_{\mathrm{macro}} = \frac{2}{3} = 0.6$.

- **Macro Recall ($\mathbf{R_{macro}}$)** Macro Recall measures the fraction of ground truth clusters $\mathcal{GE}$ that are fully captured by the predicted clusters $\mathcal{PE}$. It is calculated in a similar fashion as macro precision but with the roles of $\mathcal{GE}$ and $\mathcal{PE}$ interchanged (Equation 3.5).

$$R_{\mathrm{macro}}(\mathcal{PE},\mathcal{GE}) = \frac{|\mathcal{GE}_{\mathrm{i}} \in \mathcal{GE} : \exists \mathcal{PE}_{\mathrm{i}} \in \mathcal{PE} : \mathcal{PE}_{\mathrm{i}} \supseteq \mathcal{GE}_{\mathrm{i}}|}{|\mathcal{GE}|} \tag{3.5}$$

  Explanation: In the example in Table 3.2, only one out of three ground truth clusters GE3 is fully captured by the predicted clusters (PE3), so $R_{\mathrm{macro}} = \frac{1}{3} = 0.33$.

Gold Entity	**Ground truth-Gold standard Clusters ($\mathcal{GE}$)**	**Predicted Clusters ($\mathcal{PE}$)**
Apple Inc.	GE1: [Apple, Apple Inc., AAPL]	PE1: [Apple , Apple Inc., JNJ]
Johnson & Johnson	GE2: [Johnson & Johnson, Johnson and Johnson, Johnson Inc., J&J, JNJ]	PE2: [Johnson & Johnson, Johnson and Johnson, Johnson Inc., J&J]
Microsoft Corporation	GE3: [Microsoft Corporation, MSFT]	PE3: [Microsoft Corporation, MSFT]

Table 3.2: A sample of ground truth/gold standard clusters (GE) and predicted clusters (PE) to explain the evaluation metrics for clustering. The metrics (Section 3.8 for details) have been explained using these samples. Note that the list of entity variations used here {*AAPL*, *Apple Inc.*, *Apple*, *Microsoft Corporation*, *MSFT*, *Johnson & Johnson*, *Johnson and Johnson*, *Johnson Inc.*, *J&J*, *JNJ.*}

- **Micro Precision ($\mathbf{P_{micro}}$)** focuses on the precision of clusters for each individual entity within the dataset irrespective of the overall cluster purity. It measures the degree of purity of $\mathcal{PE}$ clusters based on the assumption that the most frequently occurring ground-truth entity is correct (Equation 3.6).

$$P_{\text{micro}}(\mathcal{PE},\mathcal{GE}) = \frac{1}{\mathcal{N}} \sum_{\mathcal{PE}_\text{i}\in\mathcal{PE}} \max_{\mathcal{GE}_\text{i}\in\mathcal{GE}} |\mathcal{PE}_\text{i} \cap \mathcal{GE}_\text{i}| \tag{3.6}$$

where $\mathcal{N}$ denotes the number of variations provided as input to be canonicalized.

Explanation: Micro Precision calculates the average purity of predicted clusters, assuming the most frequent entity in each cluster is correct. In Table 3.2, each cluster has a maximum overlap with the corresponding ground truth cluster as follows: $PE1$ with $GE1$: 2 entities, $PE2$ with $GE2$: 4 entities, $PE3$ with $GE3$: 2 entities. $P_{\text{micro}} = \frac{(2+4+2)}{10} = 0.8$.

- **Micro Recall ($\mathbf{R_{micro}}$)** is calculated like micro precision but with the roles of $\mathcal{GE}$ and $\mathcal{PE}$ interchanged (Equation 3.7).

$$R_{\text{micro}}(\mathcal{PE},\mathcal{GE}) = \frac{1}{\mathcal{N}} \sum_{\mathcal{GE}_\text{i}\in\mathcal{GE}} \max_{\mathcal{PE}_\text{i}\in\mathcal{PE}} |\mathcal{GE}_\text{i} \cap \mathcal{PE}_\text{i}| \tag{3.7}$$

Explanation: Micro Recall measures the average coverage of gold entities by the predicted clusters. In the example, each gold entity has a maximum overlap with the corresponding predicted cluster: GE1 with PE1: 2 entities, GE2 with PE2: 4 entities, and

GE3 with PE3: 2 entities. $R_{\text{micro}} = \frac{(2+4+2)}{10} = 0.8$.

- **Precision** is a measure of correctly identified similar pairs (true positives) out of all the pairs identified as similar by the architecture (Equation 3.8).

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.8}$$

- **Recall** is the correctly identified similar pairs of entity variations out of all similar pairs in the dataset (True positives and false negatives) (Equation 3.9).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.9}$$

- **F1-score** is defined as the harmonic average of precision and recall. The F1 score also serves as a balanced measurement of the precision and recall. (Equation 3.10).

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.10}$$

3.9 Results and Discussion

We evaluate the performance of pairwise similarity scores (Table 3.3 and Table 3.4) from our proposed architecture, KCNet, and compared them to other baselines in the literature.

Pairwise Similarity: Table 3.3 summarizes the test results of the pairwise similarity of KCNet along with other baseline approaches. Galarraga-IDF [39] is a weighted word overlap similarity measure based on the assumption that frequently occurring words should be given a lower weight. We observe that the method achieves 22.6 precision and 23.6 F1-scores for DBpedia (C). It suffers low F1-scores because giving lower weight to popular terms such as *Allianz* from the variations of same entity called *'Allianz Cornhill'*,*'Allianz Global Investors'*, and *'Allianz Group'* makes the final canonical form not meaningful. *Distilled S-BERT(*)+cosine* gives higher F1-score of 45.3 as compared to Galarraga-IDF [39] as these are contextual embeddings and also offers some support for out of vocabulary variations. Char-BiLSTM+A+Word+A performed better than Distilled S-BERT(*)+cosine with an F1-score of 71.3 on DBpedia (C), which shows the architecture captures overlapping patterns of characters and words with attention module. KCNet (without side info) improves average F1-scores relatively by 11.6% over Char-BiLSTM+A+Word+A for ESCO (S), ESCO (D), and DBpedia (C), respectively, in measuring the pairwise similarity. Similarly, KCNet achieves a significantly bigger F1-scores 90.6, 90.9, 89.3, and 98.8 (Table 3.4) for RDE (S), RDE (D), RDE (I), and RDE (C), respectively) in measuring the pairwise similarity improvement over the baseline approaches.

Model	**Performance on Open datasets**					
	ESCO (S)		ESCO (D)		DBpedia (C)	
	P	F	P	F	P	F
Galarraga-IDF†	50.8	32.8	61.7	38.9	22.6	23.6
Distilled S-BERT(*)+cosine	49.3	44.4	49.3	39.0	49.6	45.3
Distilled S-BERT(**)+ cosine	49.5	50.0	49.4	49.7	50.0	49.8
Char-BiLSTM+A†	85.9	86.9	76.3	75.1	72.1	59.7
Word-BiLSTM+A†	85.6	89.6	83.1	83.7	77.6	70.7
Char-BiLSTM+A+Word+A†	87.3	90.7	84.2	85.4	78.0	71.3
KCNet (without sideinfo)	99.0	95.1	98.8	86.9	99.0	92.5
KCNet (with sideinfo)	99.2	98.3	98.8	89.4	99.1	97.0

Table 3.3: Test Results of pairwise similarity using our proposed model compared to different baselines. Here, ESCO (S), ESCO (D), and DBpedia (C) refer to Open Datasets of Skills, Designations, and Companies, respectively. Results of † are taken from [2]. P and F refer to Precision and F1-scores. Distilled S-BERT (*, **) refers to (entity, entity $\odot$ side information) embedding using Distilled S-BERT model.

Model	**Performance on Proprietary datasets**							
	RDE (S)		RDE (D)		RDE (I)		RDE (C)	
	P	F	P	F	P	F	P	F
Galarraga-IDF†	33.2	12.5	63.0	60.3	64.3	66.5	75.8	71.2
Distilled S-BERT(*)+cosine	47.8	47.5	49.7	48.8	49.7	49.1	49.2	49.1
Distilled S-BERT(**)+ cosine	47.5	48.8	49.8	49.9	34.6	41.5	56.2	48.4
Char-BiLSTM+A†	81.8	86.9	72.6	77.2	84.5	84.8	99.3	98.9
Word-BiLSTM+A†	80.1	86.5	90.5	94.8	80.6	83.3	95.3	95.6
Char-BiLSTM+A+Word+A†	82.7	88.5	94.4	96.3	86.7	86.7	99.5	99.2
KCNet (without sideinfo)	96.7	90.6	99.6	90.9	92.4	89.3	99.4	98.8
KCNet (with sideinfo)	99.5	99.4	99.7	99.6	99.5	99.5	99.5	99.3

Table 3.4: Test Results of pairwise similarity using our proposed model compared to different baselines. Here, RDE (S), RDE (D), RDE (I), and RDE (C) refer to Proprietary datasets of Skills, Designations, Institutes, and Companies, respectively. Results of † are taken from [2]. P and F refer to Precision and F1-scores. Distilled S-BERT (*, **) refers to (entity, entity $\odot$ side information) embedding using distilled S-BERT model.

Contribution of Side information in KCNet: We evaluate the performances of different versions of *KCNet* (with and without side info). In Table 3.3 and Table 3.5, we observe that the side information provides a gain upto 4.5% and 10% across different open and proprietary datasets as compared to KCNet (without sideinfo). We also observe for skill entity variation pair *'mdx'*,*'MultiDimensional eXpressions'*), KCNet (without sideinfo) gives the similarity of 0.84 which shows it learns the structure and non-linear mapping in latent space even for pairs having non-overlapping words or characters. For same entity pair, Distilled S-BERT(*)+cosine approach results in low pairwise similarity of 0.73. With side information, *KCNet* returns a high similarity score of 0.99 indicating that it learns the latent semantic relationships between these two entities using the side information. For entity pairs (*uplholstery fillings, upholstery paddings*), *Distilled S-BERT(*)+cosine* returns a pairwise similarity score of 0.86, whereas *KCNet* learns a better representation and achieves a pairwise similarity score of 0.99 making it more confident for overlapping variations. Additionally, side information significantly helped in capturing semantic entity variations such as (*mycology, fungi studies*) where the best baseline *Char-BiLSTM+A+Word+A* captures limited patterns and is unable to model them as there is no overlap of characters or words in these pairs. In contrast, *KCNet* is able to capture pairwise similarity score of $>=$ 0.90 for such variations. This shows that *KCNet* can effectively model these variations well when supplemented with side information. Despite the absence of side information for some entities (Table 3.1), KCNet shows a significant improvement for the entity canonicalization task. Next, we discuss the use of pairwise similarity scores for clustering (Table 3.5).

Datasets	**Model**	**Metrics**					
		Micro			**Macro**		
		P	R	F	P	R	F
RDE (S)	β	0.71	0.64	0.67	0.94	0.31	0.47
	γ	**0.99**	**0.97**	**0.98**	**0.96**	**0.97**	**0.96**
RDE (D)	β	0.95	0.53	0.67	0.83	0.15	0.24
	γ	0.86	**0.78**	**0.82**	**0.85**	**0.54**	**0.66**
RDE (I)	β	0.84	0.75	0.79	0.96	0.48	0.64
	γ	0.83	**0.85**	**0.84**	0.74	**0.71**	**0.72**
RDE (C)	β	0.98	0.99	0.98	0.97	0.96	0.96
	γ	0.98	0.97	**0.98**	**0.98**	**0.97**	**0.98**
ESCO(S)	β	0.84	0.82	0.83	0.65	0.49	0.55
	γ	**0.93**	**0.92**	**0.92**	**0.89**	**0.75**	**0.81**
ESCO(D)	β	0.49	0.79	0.61	0.21	0.32	0.25
	γ	**0.91**	0.61	**0.73**	**0.81**	0.22	**0.34**
DBpedia(C)	β	0.88	0.52	0.65	0.92	0.25	0.39
	γ	**0.93**	**0.75**	**0.83**	0.86	**0.60**	**0.70**

Table 3.5: Test Results over HAC using pairwise similarity. Here, β:baseline (*Char-BiLSTM+A+Word+A*) and γ: proposed model (*KCNet*) with sideinfo. A one-way repeated measures ANOVA test was conducted to determine the significance for all evaluation metrics ($p < 0.00003$).

Clustering results: Table 3.5 presents the results for HAC clustering used to canonicalize entity variations in the test dataset. For DBpedia (C), we observe that our proposed approach gives higher (almost double) Macro recall (0.60) and Macro-F1 (0.70) as compared to β: baseline (Char-BiLSTM+A+Word+A). For ESCO (D), we observe that there is low Macro Recall. Due to low Macro recall, our results have generated poor Macro-F1 scores (0.34). Macro-F1 is sensitive to individual error examples in each cluster. Our low Macro-F1 may be because our error examples are distributed in many clusters. This suggests that missing or incorrectly mapping just one variation, even if all other variations are mapped correctly to different clusters compared to the gold standard, significantly affects its performance. Similarly, for RDE (D), Micro Precision (0.86) is lower than baseline (0.95) while Micro F1 (0.82) is higher than baseline (0.67). This suggests that while Micro precision may be slightly lower, our proposed approach achieves higher recall (KCNet is more inclusive of relevant variations) in a cluster. We believe that HAC is suitable for diverse datasets and can capture broad and fine-grained similarities among entities. We use a one-way repeated measures ANOVA test to determine the significance for all evaluation metrics ($p < 0.00003$) across datasets. Similar trends have been observed in past research [2].

Error analysis: Although KCNet gives promising results across all datasets, it wrongly clusters some entities; for example, some skills such as *bees wax* and *natural wax* signify the same concept but occur in different clusters. One possible reason could be that the representation of words *bees* and *natural* are far apart in the contextual vector representation space, so the model assigns a lower similarity score and hence, incorrectly classifies it. Similarly, *'packager'* is incorrectly placed in cluster of [*'dozer driver'*, *'dozer/crawler driver'*, *'packager'*]. The possible reason could be the complete absence of side information for three entities, which confuses KCNet with closer contextual vector representations. Despite this, KCNet addresses the challenge of handling abbreviations, short forms, and non-overlapping entities by learning vector representations of these entities in the kernel space.

3.10 Summary

This chapter focused on canonicalizing real-world entities from the recruitment domain such as companies, designations, institutes, and skills by designing a novel multi-tier framework Kernel-based Canonicalization Network. KCNet induces a non-linear mapping between the contextual vector representations while capturing fine-granular and high-dimensional relationships among vectors. KCNet efficiently models more prosperous semantic and meta side information from external knowledge towards exploring kernel features for canonicalizing entities in the recruitment domain. Furthermore, we applied Hierarchical Agglomerative Clustering

(HAC) using the pairwise similarity matrix $\mathcal{M}_{sim}$ to create unique clusters of entities. Experiments revealed that the Kernel-based neural network approach achieves significantly higher performance on both proprietary and open datasets. We demonstrate that our proposed methods are also generalizable to domain-specific entities in similar scenarios.

Chapter 4

Deal with missing entities in online professional content

"That's what we're missing. We're missing an argument. We're missing debate. We're missing colloquy. We're missing all sorts of things. Instead, we're accepting."

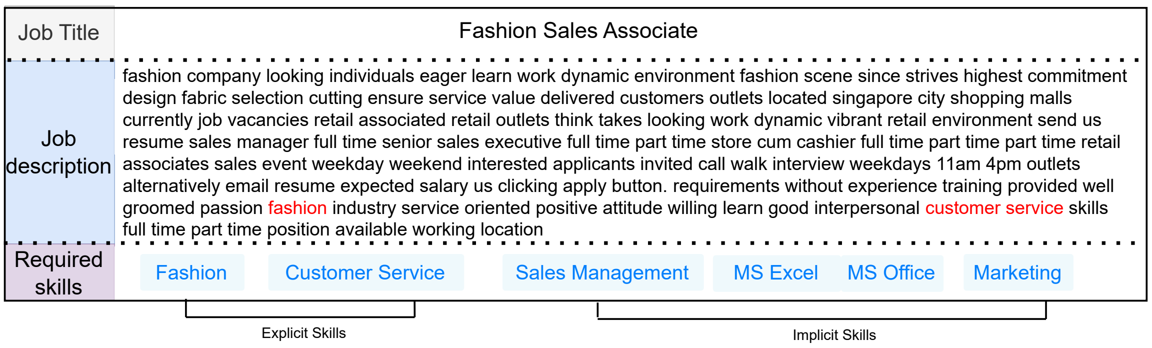
- Studs Terkel

4.1 Introduction

In this[1] chapter, we focus on the completeness (finding missing skills) of professional content, which plays an important role in determining whether a candidate will be selected or not. Recruiters create job positions mentioning skills, roles, and responsibilities to reach potential candidates. By posting high-quality jobs, recruiters bring job seekers closer to becoming applicants when they click on job postings. Unfortunately, recruiters miss out on the candidate if the job posting doesn't provide enough information or compel them to apply. Some recruiters often lack the expertise to determine what information should be included while posting a job, leading to missing information in the job and thus affecting content quality [109]. Recruiters often miss including crucial job skills due to a communication gap with domain experts. Among all these required fields, skills are crucial to determine whether a candidate is suitable for a job position [133]. According to statistics, 65% of the job descriptions (JDs) do not include relevant and popular skills, and 40% of JDs miss listing 20% or more explicitly-stated skills (present as substrings in the JD) in the prose description [25]. Most of this information is

[1]Work presented in this chapter, has been accepted in Findings of EACL 2023. **Goyal, N.**, Kalra, J., Sharma, C., Mutharaju, R., Sachdeva, N., and Kumaraguru, P. JobXMLC: EXtreme Multi-Label Classification of Job Skills with Graph Neural Networks. In Findings of the Association for Computational Linguistics: EACL 2023.

found in the job description, including the required skills field. Figure 4.1 shows a job from my career future.sg dataset consisting of the job title, job description, and required skills where some skills are missing from the job description. These missing skills affect the performance of recruitment tasks such as job suggestions, job search, candidate recommendations, person-job fit, etc. Therefore, it is imperative to recommend such missing skills to improve the quality of job postings. Despite considerable advances in missing data prediction techniques over the last three decades, missing entities remain largely unsolved for domain-specific scenarios.



fashion company looking individuals eager learn work dynamic environment fashion scene since strives highest commitment design fabric selection cutting ensure service value delivered customers outlets located singapore city shopping malls currently job vacancies retail associated retail outlets think takes looking work dynamic vibrant retail environment send us resume sales manager full time senior sales executive full time part time store cum cashier full time part time part time retail associates sales event weekday weekend interested applicants invited call walk interview weekdays 11am 4pm outlets alternatively email resume expected salary us clicking apply button. requirements without experience training provided well groomed passion fashion industry service oriented positive attitude willing learn good interpersonal customer service skills full time part time position available working location

Figure 4.1: An example of a job from mycareersfuture.sg dataset. The job description does not include implicit and job-specific skills (absent from job description) such as *'sales management'*, *'MS Excel'*, *'MS Office'*, and *'Marketing'*.

Traditional approaches [134, 135] often rely on sequence labelling techniques to annotate job descriptions, treating contiguous chunks of text as prediction targets (skills). However, these approaches need manually annotated labels from textual JDs, making it sub-optimal to train a sequence labelling task. Recent works [25, 41] have explored missing skill prediction tasks using large-scale pre-trained language models. Document embedding and Graph-based systems [136, 137] are used for skill extraction and recommendations. However, these approaches have a few shortcomings, a) language models are limited to contextual modelling and bring high computational costs at massive scales as a task not only involves predicting multiple missing skills but also requires precisely organising the most relevant skills specific to the job posting, b) do not exploit inter-relational structures like job-job and job-skill relationships. For example, assume jobs j_1 and j_2 share a common skill s_1. If another skill s_2 is relevant to j_2 and other similar jobs, we can infer that s_2 might also be relevant to j_1. Such transitive cues can be extremely useful for identifying missing skills. Current deep extreme classifiers [42, 138] find it hard to model such implicit relationships unless the training set explicitly contains a pair (j_1, s_2); c) associated relevant skill labels exhibit interdependence. For instance, '*Javascript*' and '*CSS*' labels frequently co-occur, while the simultaneous occurrence of '*Python*' and '*tally*' labels is rare. This means we can't give equal importance to every skill corresponding to the job. However, each skill in the skill label set has different weights based on the frequency of

their occurrence in job descriptions. Missing skills applications also face extreme skill label sparsity; using label co-occurrence alone yields fractured correlations.
Graphs are naturally suitable to make the relationships explicit, such as job-skill networks. Two nodes (jobs) are likely to have common neighbors (skills) if the jobs have overlapping skills. Although this task can be formulated as a link prediction task, which traditional approaches often tackle using proximity-preserving embeddings. These embeddings maintain the closeness of related data points, leading to many similar pairs. However, these approaches [83, 139] may not perform well in inductive settings where the model needs to predict unseen relationships during training. In recent developments, a Graph Neural Networks (GNNs) [93, 140] have emerged as a well-known architecture for modeling structural relationships between nodes (jobs and skills), enabling the representation of knowledge and additional information. Authors [141] observed that the performance of GraphSAGE [93] (which works in inductive settings) trained on a collaborative graph (document-label) for a link prediction task deteriorated with an increasing value of K. Similar works [142, 143] also observed that GNNs such as GraphSAGE [93] are unable to make optimal use of higher-order neighbourhoods. Over the past decade, Extreme Multi-label Classification (XMLC) techniques have been designed to handle scenarios where the task is to tag a document with its most relevant subset of labels from an extremely large label space, such as in the case of job-skill graphs in our work. We take the XMLC perspective as nodes (jobs) need to be associated with a large number of skill labels. In this context, predicting the links (edges) between jobs and skill labels in the bipartite graph essentially translates to predicting the relevant labels for a given job, which aligns with the XMLC objective. Therefore, solving it as an XMLC task is more suitable for our scenario (missing skill prediction). To this end, we propose a framework called JobXMLC, which uses a GNN to jointly model the jobs and skills in the same space with label (skill) attention module. Through the use of ubiquitous job description data, we aim to predict the missing skills using collaborative learning of jobs and skills and modeling our problem as an Extreme Multi-label Classification (XMLC) task.

4.2 Related Work

Several works [25, 144] have been done for skill prediction using Extreme Multi-label Classification. XMLC refers to the classification of text where the number of the set of labels is large, i.e., thousands or millions. One-vs-All (OVA) is a well-known method for text classification tasks with high accuracy [145]. The OVA approach is computationally efficient for the XMLC task for modest-sized label sets (up to a few thousand labels).
These methods are broadly classified as (a) Deep learning-based models, (b) Graph-based ap-

proaches, and (c) Domain-specific methods.

Deep learning-based methods. Deep learning models that use powerful text representation capabilities have also been explored for the XMLC problems [42, 146]. XML-CNN [147] applies a dynamic max-pooling scheme and a family of CNN models to learn text representations. AttentionXML [42] uses the attentional bidirectional long short-term memory (BiLSTM) networks to extract embeddings from raw text inputs. However, the CNN-based models cannot capture the most relevant parts of the information on each label. The RNN-based methods fail to model long-term dependencies due to vanishing gradients. Research [25] explores the language models such as ELMo, Transformer and BERT [148], and X-BERT [146] for XMLC task. These approaches model input language's syntactic and semantic structure to predict tokens based on the available contextual information. However, such models are computationally expensive and require a predefined meaning of labels. In addition, the difficulty of scaling to the extreme label space remains in deep learning methods as the output layer scales linearly with the product of label size and feature dimension. The research gaps with deep-extreme classifiers motivate us to explore alternative approaches and techniques that do not require explicit label representation or predefined semantic meaning of labels and are scalable for extensive datasets.

Graph-based approaches. In the realm of graph-based approaches, a more intuitive strategy for identifying missing links involves framing it as a link prediction task. This approach frequently utilizes proximity-preserving embeddings to capture the relationships among nodes and edges within graphs. However, traditional graph-based approaches [83, 84, 139] may not perform well in inductive settings (where the model needs to predict unseen links during training. The recent proliferation of graph neural networks [140] allows using node neighborhoods to learn more discriminative features collaboratively. GraphSAGE [93] proposed the computation of node representations inductively by recursively aggregating over fixed-sized neighborhoods. Authors [95] proposed the Graph Isomorphism Network (GIN) with discriminative power equal to that of the WL test. GraphSAINT [94], a graph sampling-based inductive learning method, compute node representations based on the local graph structure and node attributes. Previous studies [142, 143] have noted that GNNs like GraphSAGE [93] do not effectively leverage higher-order neighborhoods for link prediction tasks. This insight suggests opportunities for improving how GNNs handle complex graph structures.

Domain-specific methods. Research work identifies the skill bases used for analyzing the job market, the type of extracted skills [109], the skill identification methods, the studied sector and their granularity. Xu et al. [110] build a job-skill network to measure the popularity of the skills by exploring a large corpus of job postings. Bhola et al. [25] employs an Extreme Multi-label Classification method that utilizes the Transformer model to predict the required skills from textual job descriptions.

However, these approaches are computationally expensive and either predict frequent skills or miss rare crucial skills for recruiters. To address all the existing challenges and limitations, we propose JobXMLC that uses graph neural networks with skill attention to learn multi-hop job-skill networks. Inspired by [141], JobXMLC effectively covers multi-hop neighborhood embeddings which aggregate information from nodes that are reachable within several hops and allow flexible integration of features of job and skills into the graph. To the best of our knowledge, this work is the first to exploit GNNs for the job-skill prediction task.

4.3 Contributions

The contributions of this work are:

- We construct a novel job-skill graph consisting of $22,844$ (jobs and skills) and 650K relationships that allow flexible integration of textual features and various pre-trained language representation models.
- We propose JobXMLC comprising of graph neural networks with skill attention to learn multi-resolution graph neighborhoods with the sampling method.
- We also provide the performance comparison of JobXMLC, which outperforms by a margin of 6% from the best baselines.
- JobXMLC is lightweight, up to 18X faster in training and 634X in predicting than existing deep learning-based extreme classifiers to scale up to thousands of labels. We have made our code and dataset public at https://precog.iiit.ac.in/resources.html.

4.4 Problem Formulation

Consider the set of jobs $\mathcal{J} = \{j_1, j_2, \cdots\cdots, j_i\}$. A job $j_i \in \mathcal{J}$ corresponds to its textual description and $\mathcal{S}_i$ is the set of skill labels for the i^{th} job. The skill set is represented as, $\mathcal{S}_i = \{s_i^1, s_i^2, s_i^3, \cdots\cdots, s_i^k\} \ \forall\ 1 \leq k \leq n$, where n refers to total skills that vary differently for each dataset. The task of JobXMLC is to learn a function $f : \mathcal{J} \rightarrow 2^{\mathbb{S}}$ that maps a job $j_i \in \mathcal{J}$ to its target skill set $S_i \in \mathbb{S}$, where $\mathbb{S} = \{\mathcal{S}_1 \cup \mathcal{S}_2 \cup ... \cup \mathcal{S}_{|\mathcal{J}|}\}$.

4.5 JobXMLC: EXtreme Multi-label Classification of Job Skills

In this section, we introduce JobXMLC as shown in Figure 4.2. The architecture is inspired by the models proposed in [141]. The architecture comprises of three major components: 1) Job-skill graph 2) Graph Neural Network that learns multi-hop embeddings with neighbourhood selection approach on the job-skill graph 3) a scalable mechanism of extreme classifiers to predict skill labels in cold and warm-start scenarios.

4.5.1 Module I: Job-skill graph

We first formally define a job-skill graph, which is usually represented as a tuple $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ and $\mathbb{E}$ are the set of nodes and edges respectively. Here $\mathbb{V}$ consists of jobs belonging to $\mathcal{J}$ and skill set $\mathbb{S}$ (See Section 4.4). We construct an edge $e \in \mathbb{E}$ between j_i and s_i^k where $\mathbb{E} \subset \mathcal{J} \times \mathbb{S}$ iff s_i^k is relevant to j_i i.e., s_i^k is a positive label for j_i. Each node $v \in \mathbb{V}$, is initialized as a d-dimensional vector based on its textual features (job descriptions and skill labels). In the textual features of jobs, we leverage word-level information, including POS tagging [33] and word importance (TF-IDF) [149] to improve structure and relevancy of job descriptions. When we apply POS tagging to job descriptions, we focus on the most relevant words (nouns) that are likely to indicate skills. We found that around 60% of the words in job descriptions are nouns. The underlying assumption is that skill labels are mostly nouns [135], such as 'Java' and 'Python'. By filtering out less relevant words (verbs, adjectives, adverbs) which are not indicative of skills, POS tagging help in understanding the syntactic structure of the sentences in job. Similarly, common words like "requirements", "experience", "team", and "work" tend to have high frequencies across many job descriptions. We use TF-IDF, which assigns a lower weight to these common words because they do not provide meaningful information about the skills needed for a job. We enhance the relevance of the information extracted from job descriptions by prioritizing less common words with higher TF-IDF scores. Each word in the job description gets its initial [131] representation from a fasttext model fine-tuned on recruitment domain datasets (See details in Section 4.6). We average these representations to obtain a d-dimensional initial representation z_j^0 for each node (job) from its textual features. Similarly, we generate the initial representation t_s^0 for each node (skill) using fastText embeddings of the 'name of skills' such as 'python-3.x', 'asp.net-core'. For each node v, its initial representation is $h_v^{(0)}$ (with $h_v^{(0)} \equiv z_j^0$ if the node v is a job $j \in [\mathcal{J}]$ and $h_v^{(0)} \equiv t_s^0$ if node v is a skill label $s \in [\mathbb{S}]$).

4.5.2 Module II: Multi-hop job-skill graph neural network

To learn a job-skill graph, the module introduces the propagation network and neighborhood selection approach.

Propagation Network. This network captures higher-order job-skill graph structure using multiple layers of aggregation, each layer aggregating information from the previous layer's node representations. Therefore, we utilize Graph Isomorphism Network (GIN) [95] encoder for representations considering its outstanding expressive capacity and model simplicity. We chose Graph Isomorphism Network (GIN) because it has high representational power empirically which is crucial to capture graph structures. It consists of a convolution to aggregate information from a node's neighbors and update the node representation using convolved embeddings, $f_v^{(k)}$ to obtain transformed embeddings, $h_v^{(k)}$. We also learn multi-hop representations with k-hop neighbors of each node, where k is a hyperparameter. For instance, if k=1, the encoder would only consider the immediate neighbors (skills) of each node (job) in the graph. If k=2, it would consider the neighbors (jobs) of the neighbors (skills) and so on. Equation 4.1 shows the graph neural network layer that updates the node representation using a weighted sum of neighboring node features.

$$f_v^{(k)} = (1+\lambda_k)f_v^{(k-1)} + \sum_{j \in \mathbb{N}_v, j \neq v} f_j^{(k-1)} \tag{4.1}$$

where $\mathbb{N}_v$ be the set of neighboring nodes of an i^{th} node after sampling (Neighborhood selection 4.5.2); $f_v^{(k)}$ be the representation of the v^{th} node after layer k, and λ_k is a fixed scalar for layer k.

$$h_v^{(k)} = f_v^{(k)} + g(\delta(R_k * g(f_v^{(k)}))) \tag{4.2}$$

where $g(.)$ is ReLU activation, $\delta(.)$ is batch normalization and R_k is a parameter matrix for the residual layer and $f_v^{(k)}$ are the convolved embeddings obtained in Equation 4.1.

Equation 4.2 shows the transformed embeddings, $h_v^{(k)}$ after applying residual layer followed by batch normalization and ReLU. To avoid over-smoothing, we utilize the skip connection operation that gathers information from historical representations of nodes.

Neighborhood selection. Instead of considering all k-hop neighbors of each node, we sampled a subset of the neighbors at each layer of the network. The goal of selection is to reduce the computational cost of the network and ensure that our model is scalable in dense settings. Therefore, we select the top l neighbors based on their frequency of occurrence. Formally, for every node $v \in \mathbb{N}_v$, we accomplish frequency-based sorting where a set of fanouts $[k]$

neighbors (See details in Table 4.2) are sampled for every node to construct $\mathbb{N}_v$ as shown in Equation 4.1. From this module, we obtain transformed embeddings, $h_v^{(k)}$ at each successive hop/ multi-resolution embeddings (Figure 4.2). By default, $f_v^{(0)}$ was set to initial representation $h_v^{(0)}$ (with z_j^0 if the node v is a job $j \in [\mathcal{J}]$ and t_s^0 if node v is a skill label $s \in [\mathbb{S}]$).

4.5.3 Module III: Extreme multi-label classification

In this module, we employ One-vs-All (OvA) classifiers, $C = [c_1, c_2, ..., c_{\mathbb{S}}] \in \mathbb{R}^{\mathcal{J} \times \mathbb{S}}$ to exploit the multi-resolution embeddings obtained from the Module II. This involves training a separate binary classifier for each skill label s in the label space $\mathbb{R}^{\mathbb{S}}$. Inspired by [141], JobXMLC uses a skill label attention mechanism, so that every skill label s has a different view of the job-skill graph by weighing these multi-resolution embeddings differently while applying a skill label classifier. The skill attention module and prediction pipeline are explained in detail below.

Skill attention: We incorporate a skill attention mechanism for each skill $s \in [\mathcal{S}]$ and layer $k \in [\mathcal{K}]$. Given multi-resolution embeddings z_j^k when node v is a job $j \in [\mathcal{J}]$ for $k \in [\mathcal{K}]$, a skill attention mechanism allows the model to weigh these resolutions differently. Skill labels that appear frequently benefit from focused embeddings, which are achieved using a smaller k and rare skill labels, with their low connectivity in the graph, may benefit from embeddings using a larger k. Hence, we obtain attention weights α_{sk} using a softmax operation (Equation 4.3).

$$\alpha_{sk} = exp(e_{sk}) / \sum_{k' \in [K]} exp(e_{sk'}) \tag{4.3}$$

Here, real scalars, $exp(e_{sk}) \in \mathbb{R}$ are learnt for every skill label s and layer k in graph neural network.

$$z_j^{(s)} = \sum_{k \in K} \alpha_{sk}.z_j^k \tag{4.4}$$

Further, we calculate a skill label-specific embedding, $z_j^{(s)}$ for a skill label $s \in [S]$ using multi-resolution embeddings z_j^k for a job (Equation 4.4). Once, the label-specific embedding is obtained, we apply OvA classifier to generate a score for the skill label as $score_s = \left\langle c_s, z_j^k \right\rangle$. These scores are then used to determine the relevance of each skill label for the job.

Prediction Pipeline: In the JobXMLC framework, the prediction pipeline assigns relevant skill labels to job descriptions from a vast skill label space $\mathbb{R}^{\mathbb{S}}$. It is designed to handle various scenarios, including cold start (where the test job appears for the first time) and warm settings (where some skill labels are partially revealed).

Cold-start settings: We use fastText embeddings to obtain initial representation $\hat{z}_j^{(0)}$ of a new test job as described in Module I. We utilize initial representations of skill labels, $t_s^{(0)}$, and jobs, $z_j^{(0)}$ used in training to construct an Approximate Nearest Neighbors (ANNS) graph. This process introduces edges for the test job into the existing job-skill graph using the *prediction-introduce-edges* parameter (see Table 4.2). These neighbors may belong to either $\mathcal{J}$ or $\mathcal{S}$.

Warm-start settings: In XMLC literature [141, 150], warm-start refers to scenarios where a (small) subset of relevant skill labels for a (test) job is revealed before requiring further skill predictions by the framework/model. The warm-start setting occurs when the recruiter specifies some skills before writing the job description. Unlike the cold-start scenario, edges are introduced to the warm-start skill labels for the test job in the existing job-skill graph.
In both settings, JobXMLC is used to obtain multi-resolution embeddings $\hat{z}_j^k$ for the test job, $k \in [\mathcal{K}]$. Experiments for Warm and Cold start scenarios are shown in Table 4.10. Finally, we use a shortlisting mechanism that reduces complexity and enhances the efficiency of the prediction process.

Shortlister: For the test job, the skill label-specific embeddings z_j^s (Equation 4.4) are created with respect to skill labels $s \in \mathcal{S}$. While evaluating scores for all skill labels s would take $\Omega(\mathcal{J}\mathcal{S})$ time, therefore a shortlister is required to select a set of $\mathcal{O}(\log \mathcal{S})$ skill labels that appear most relevant to the job. To create the shortlister, multi-resolution embeddings of skill labels are averaged as shown in Equation 4.5 .

$$\bar{t}_s = \frac{1}{K} \sum_{k \in [K]} t_s^k \tag{4.5}$$

Similarly, for the test job, its multi-resolution embeddings obtained from GNN architecture are averaged as shown in Equation 4.6. We create an ANNS graph based on the cosine similarity between $\bar{t}_s$ and $\bar{\hat{z}}$ and rank the top *num_shortlist* (See Table 4.2) neighbors. This process identifies the set S of potential labels for which label-wise embeddings are calculated.

$$\bar{\hat{z}} = \frac{1}{K} \sum_{k \in [K]} \hat{z}^k \tag{4.6}$$

Once the label-specific embedding is obtained, OvA classifier are applied as described in Skill attention (Section 4.5.3) to determine the relevance of each skill label for the job.

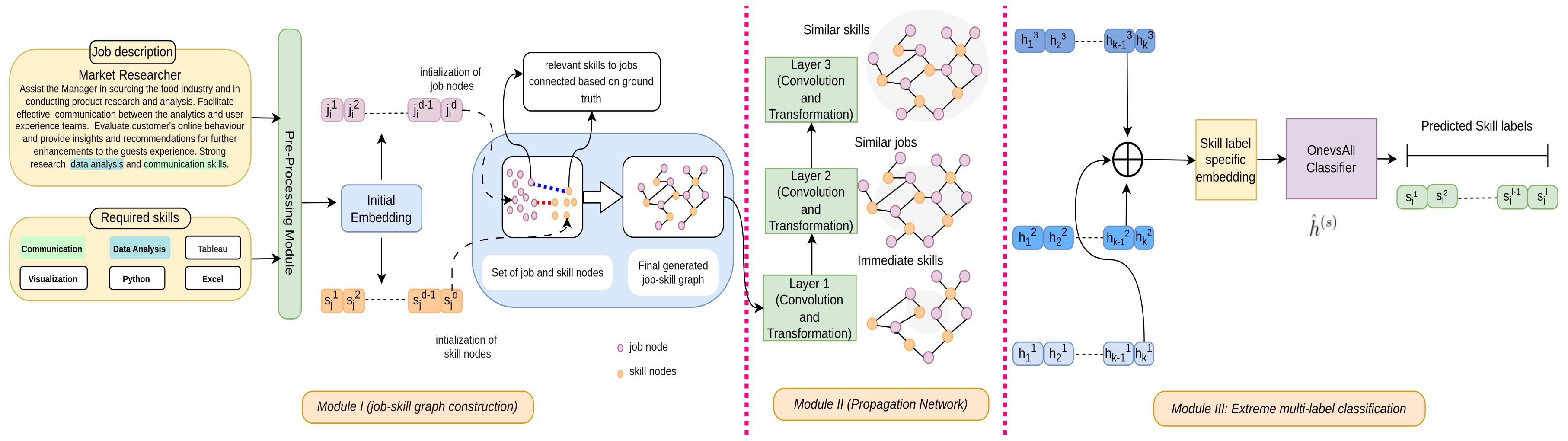

Job description
Market Researcher
Assist the Manager in sourcing the food industry and in conducting product research and analysis. Facilitate effective communication between the analytics and user experience teams. Evaluate customer's online behaviour and provide insights and recommendations for further enhancements to the guests experience. Strong research, data analysis and communication skills.
Required skills
Communication
Data Analysis
Tableau
Visualization
Python
Excel
Pre-Processing Module
Initial Embedding
intialization of job nodes
intialization of skill nodes
relevant skills to jobs connected based on ground truth
Set of job and skill nodes
Final generated job-skill graph
job node
skill nodes
Module I (job-skill graph construction)
Similar skills
Similar jobs
Immediate skills
Layer 3 (Convolution and Transformation)
Layer 2 (Convolution and Transformation)
Layer 1 (Convolution and Transformation)
Module II (Propagation Network)
Skill label specific embedding
OnevsAll Classifier
Predicted Skill labels
Module III: Extreme multi-label classification

Figure 4.2: **JobXMLC** consists of three components: Module I consists of a mechanism to construct a job-skill graph, and Module II consists of a graph neural network-based architecture that learns embeddings and multi-hop neighborhoods using a job-skill graph effectively. Module III uses a scalable mechanism of extreme classifiers to predict missing skills.

Element	mycareersfuture.sg	StackOverflow
No. of job posts	$20,298$	$20,320$
# of distinct skills	$2,548$	275
# of skills with 20 or more mentions	$1,209$	50
Average skill tags per job post	19.98	2.8
Average token count per job post	162.27	200.8
Maximum token count in a job post	$1,127$	800

Table 4.1: Dataset statistics for mycareersfuture.sg and StackOverflow.

4.6 Experimental Setup

4.6.1 Datasets

We utilize two real-world recruitment datasets, namely mycareersfuture.sg [25] and StackOverflow[2] collected from popular recruitment platforms. These datasets comprise over $20,000$ richly structured job posts with 23 informative fields about the advertisement details and current status. Table 4.1 reports the statistics for recruitment domain datasets. Small-scale datasets vary from 100 to 300, whereas large-scale ranges from 300 to millions of labels. Similar scales for the XMLC task are demonstrated in [147, 151].

Data Pre-processing. We filtered out *job descriptions*, *job title*, *required skills* corresponding to every job posting. From mycareersfuture.sg dataset, we consider concatenation of *'roles & responsibilities'* and *'job requirements'* fields as the *'job description'*, and *'required skills'* as the set of target discrete labels. Similarly, for StackOverflow dataset, we consider the *'job description'* and *'required skills'* sections. We filtered out the jobs with empty or single words in the textual content. We also perform lower-casing, stopwords removal, and removal of less important strings such as *'available'*, and *'requirements'*, which are present in most JDs. StackOverflow dataset consists of 6M words with 298,729 unique words. We split the dataset into training, validation and testing datasets with an 80:10:10 proportion. Similar splits have been utilized by competitive methods [25].

Graph construction. We utilize these pre-processed datasets to construct the job-skill graph using the methodology described in Module I (See Section 4.5.1).

[2]https://stackoverflow.com/

4.6.2 Implementation and Competing Methods

This section will discuss the baselines, training and hyperparameter details.
Baselines: We show the effectiveness of different aspects of JobXMLC and evaluate our model performance against competitive transformer-based baselines. These constitute CNN [152], LSTM [153], BiLSTM [154], BiGRU [155], BERT-XMLC [25], RoBERT-XMLC [41], GalaXC [141], JobXMLC (GraphSAGE) [93], and JobXMLC (GraphSaint) [94]. We discuss transformer-based approaches, BERT-XMLC [25] encodes the words of the job descriptions using a pre-trained BERT model. The encoding of the [CLS] token is then used to represent the job description. The job representation is passed to a bottleneck layer (i.e., an added linear layer before the output layer). The last layer treats every skill as a binary classification problem, so for each skill it calculates the probability that the skill is associated with JD.
State-of-the-art models such as CNN [152], LSTM [153], Bi-GRU [155], and Bi-LSTM [154] are self-explanatory. We utilize two neural network layers for all RNN-based models.
GalaXC [141] describes a novel framework for Extreme classification using graph neural networks (GNNs). The authors provided the implementations for all the above mentioned algorithms. We set the hyperparameters of the baseline algorithms as suggested by the authors, wherever applicable. GraphSAGE [93] and GraphSaint [94] encode the node information and are useful for graphs with rich node attribute information for Extreme multi-label classification. These methods have been thoroughly discussed in the Chapter 2, Section 2.3.

Training and Hyperparameter details: This section reports the hyperparameters used for

Hyperparameter	**Carrersfuture.sg**	**StackOverflow**
No. of epochs	20	30
num_HN_epochs	20	20
learning rate (lr)	0.0003	0.0003
attention_lr	0.0003	0.0003
dlr_factor	0.5	0.5
batch_size	256	256
fanouts	5,5,5	5,5,5
num_HN_shortlist	500	3
embedding_type	fastText	fastText
num_shortlist	1500	275
prediction_introduce_edges	3	3

Table 4.2: Hyper-parameters for mycareersfuture.sg and StackOverflow dataset for JobXMLC. fastText refers to 300-dimensional embeddings obtained by fine-tuning fastText model on job descriptions.

experiments conducted in work. We conducted our experiments using the list of hyperparameters reported in Table 4.2. We utilize binary cross-entropy loss and Adam optimizer. We use the drop-out layer after every ReLU layer.

- No. of epochs: refers to number of epochs for JobXMLC.
- Num_HN_epochs: number of hard negative epochs for JobXMLC.
- Learning rate (lr): is the learning rate for JobXMLC.
- Attention_lr: is the learning rate used by skill attention.
- dlr_factor: defines factor by which learning rate is decayed.
- Batch_size: refers to batch size used during training of JobXMLC.
- num_HN_shortlist: refers to number of hard negative labels to be selected by sampling from other data points from the same batch.
- num_shortlist: refers to number of skills sampled by shortlister.
- prediction_introduce_edges: refers to total edges that should be introduced to the graph at prediction time.
- fanouts: refers to the number of neighbors to sample for layer k.

4.7 Evaluation Metrics

We utilize Precision @k (P@k), Recall@k (R@k), and Retrieval based metrics (Normalized Discounted Cumulative Gain (N@k), and Mean Reciprocal Rank (MRR)). NDCG and MRR complements recall-oriented metrics by providing additional perspectives on the quality of ranked retrieval results. We also measure the implicit and explicit status of skills using Explicit Inference Measure (EIM), Relative Implicit Inference Measure (RIIM), Relative Explicit Inference measure (REIM) as evaluation metrics to capture model performance for the skill prediction task. These evaluation metrics have been adopted from literature [25].

- **Precision@k (P@k)** includes the proportion of relevant skills (skills in ground truth) in the top-k skill prediction list to focus on the quality of the top-k predictions. It measures how many of the top-k predicted skills are actually relevant skills (skills in ground truth) for the job posting (Equation 4.7).

$$\text{P@}k = \frac{\text{\# relevant skills in top } k \text{ predictions}}{k} \tag{4.7}$$

where k is the number of top skill predictions considered.

For instance, If out of the top 5 predicted skills, 3 are actually relevant, then Precision@5 is, $\frac{3}{5} = 0.6$.

- **Recall@k (R@k)** includes the proportion of relevant skills (skills in ground truth) found in the top-k skill prediction list. Recall@k focuses on the completeness of the predictions. It measures how many of the relevant skills (skills in ground truth) are found in the top-k predictions (Equation 4.8). The output of Recall@k ranges between 0-100.

$$\text{R@}k = \frac{\text{\# relevant skills in top } k \text{ predictions}}{\text{\# relevant skills}} \tag{4.8}$$

where numerator is the number of relevant skills found in the top-k predictions and denominator is the total number of relevant skills.

For instance, If there are 10 relevant skills for a job description and the top 5 predictions include 3 of them, then Recall@5 is $\frac{3}{10} = 0.3$.

- **Mean Reciprocal Rank (MRR)** indicates the (reciprocal) position of the first true positive in the predicted set of skills (Equation 4.9). The first true positive skill is the first relevant skill in the ranked list of predictions for a job description. Higher MRR values indicate that relevant skills are found earlier in the ranked lists. MRR is sensitive to the first occurrence of any relevant skill. If the first relevant skill is found at position 1, then MRR is maximized. MRR yields a score between 01.

$$\text{MRR} = \frac{1}{|N|} \sum_{i=1}^{|N|} \frac{1}{\text{rank}_i} \tag{4.9}$$

where $|N|$ is the total number of JDs and rank_i is the position of the first relevant skill for the i-th job. Table 4.3 shows an illustrative example to calculate Reciprocal Position for every JD (JD1, JD2, JD3) and their predicted skills.

$$\text{MRR} = \frac{1}{3}(0.5 + 1.0 + 1.0) = \frac{1}{3} \times 2.5 = 0.83$$

- **Normalized Discounted Cumulative Gain (N@k)** discounts the true positives that occur later in the prediction rankings. Discounted Cumulative Gain (DCG@k) calculates the sum of relevance scores of the top k predictions discounted by their position in the list (Equation 4.10). Ideal DCG (IDCG@k) measures the maximum possible DCG that could be achieved if all relevant skills in the ground truth appeared within the top k po-

Particular JDs→	**JD1**	**JD2**	**JD3**
Ground Truth (GT)	{Python, Machine Learning, Data Analysis}	{Project Management, Agile, Scrum}	{HTML, CSS, JavaScript}
Predicted Skills	{Java, Machine Learning, SQL, Python}	{Agile, Scrum, Project Management, Communication}	{JavaScript, Java, HTML, CSS}
First GT Skill and Position	Machine Learning (Position 2)	Agile (Position 1)	JavaScript (Position 1)
Reciprocal Position	$\frac{1}{2} = 0.5$	$\frac{1}{1} = 1.0$	$\frac{1}{1} = 1.0$

Table 4.3: A sample illustration of Reciprocal position for predicted skills using job descriptions.

sitions of the predictions (Equation 4.11). NDCG is computed as the ratio of DCG to IDCG (Equation 4.12) and its score ranges between 0-100, similar to recall.

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)} \tag{4.10}$$

where rel_i is the relevance score at position i in the prediction list. Since all relevant skills in the ground truth are treated equally, rel_i is 1 if the skill at position i in the prediction is relevant and 0 otherwise.

$$\text{IDCG@}k = \sum_{i=1}^{k} \frac{1}{\log_2(i+1)} \tag{4.11}$$

$$\text{N@k} = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{4.12}$$

For instance, If we have to calculate $N@2$ where k=2 where Ground Truth (GT)= {*SkillA, SkillB, SkillC*} and Skill predictions list= {*SkillB, SkillC, SkillA, SkillE, SkillD*}
We will first calculate DCG@2:

$$\text{DCG@2} = \frac{1}{\log_2(1+1)} + \frac{1}{\log_2(2+1)} = 1 + 0.631 \approx 1.631$$

IDCG@2:

$$\text{IDCG@2} = \frac{1}{\log_2(1+1)} + \frac{1}{\log_2(2+1)} = 1 + 0.631 \approx 1.631$$

N@2:

$$\text{NDCG@2} = \frac{\text{DCG@2}}{\text{IDCG@2}} = \frac{1.631}{1.631} = 1$$

The final score N@2 signifies that the predictions exactly matches the ideal case based on the ground truth, considering equal importance for all relevant skills.

- **Explicit Inference Measure (EIM)** is the micro, instance-based measure of explicit skills predicted by the model, compared against the explicit skills present in ground truth (See Figure 4.1) mentioned for a JD, for instance, if the model declares 5 substrings of the JD as skills, and there are 7 explicit skills present in required skill labels of JDs, EIM = 5/7 = 0.71 (71%).

- **Relative Implicit Inference Measure (RIIM):** macro, the recall-based measure of implicit skills predicted by the model, relative to the entire set of implicit skills. For example, if the model recovers 2 of 4 skills that are implied (See Figure 4.1) but not explicitly substrings in the JD, RIIM = 2/4 = 0.5 (50%).

- **Relative Explicit Inference Measure (REIM)**: macro, recall-based measure of explicit skills predicted by the model compared to the entire set of explicit skills. For instance, if the model declares 5 substrings of the JD as skills, and there are total 15 explicit skills present in JD, REIM =5/15 = 0.33 (33%).

For evaluations, we do not consider the negatives (skills absent in '*required skills*') as true negatives because they might have been overlooked during the construction of the ground truth for jobs. Similar observations were made in the literature [25], indicating that the required skills (ground truth) in job descriptions are often incomplete.

4.8 Results and Analysis

Table 4.4 and Table 4.5 reports **Recall@***k* and **Precision@***k* for all state-of-the-art approaches and JobXMLC on both datasets. Compared to other baselines, JobXMLC is at least 15% better than Bi-LSTM [154] in R@5 for StackOverflow dataset. Further, fastText initialization in JobXMLC for mycareersfuture.sg dataset is atleast 9% better than BERT-XMLC+CAB [25] in R@5, indicating that the global relationships improve the model better in addition to local connections through joint learning. Compared to BERT-XMLC [25] and RoBERTa [41], both utilize transformer-based embeddings and skill correlation-based features for training, JobXMLC is 13% and 20% better across R@5 and P@5 metrics for mycareersfuture.sg dataset. JobXMLC outperforms Graph-based methods such as GalaXC [141] by 11-12% for R@5 metrics on both datasets. For example, on the mycareersfuture.sg dataset, the Recall for the top 5 and top 30 skill labels for JobXMLC are 18.29 and 63.18, respectively. Based on an average of 19.98 skills per job, about 3.65 skills were derived within the top 5 and 12.63 within the top 30.

We also observe that for precision-based measures such as P@30, the numerator can be the

Dataset $\rightarrow$	**mycareersfuture.sg dataset**					
Model $\downarrow$	**R@5**	**R@10**	**R@30**	**P@5**	**P@10**	**P@30**
CNN [152]	14.17	23.58	45.34	56.67	47.17	30.23
LSTM† [153]	11.67	18.44	35.02	46.67	36.89	23.34
Bi-LSTM† [154]	13.02	21.37	41.54	52.07	42.75	27.70
Bi-GRU† [155]	13.98	23.43	44.41	55.94	46.87	29.61
BERT +XMLC [25]	15.27	25.96	51.18	61.06	51.92	39.32
RoBERTa+XMLC [25]	16.15	26.52	51.99	60.08	53.85	39.87
BERT +XMLC+CAB [25]	16.72	29.45	58.98	66.87	58.90	41.21
GalaXC [141]	16.31	28.34	54.16	65.25	56.70	36.11
JobXMLC (GraphSaint) [94]	16.23	27.79	53.32	64.93	55.59	35.55
JobXMLC (GraphSAGE) [93]	16.84	29.18	56.89	67.36	58.36	37.93
JobXMLC	**18.29**	**32.33**	**63.18**	**73.20**	**64.66**	**42.22**

Table 4.4: Results of JobXMLC along with state-of-the-art approaches on mycareersfuture.sg dataset. For RNN-based models (†), we have limited all model architectures to two layers.

Dataset	**StackOverflow**					
Model	**R@5**	**R@10**	**R@30**	**P@5**	**P@10**	**P@30**
CNN [152]	25.16	39.39	64.80	15.24	11.72	6.36
LSTM† [153]	26.63	40.47	67.89	16.07	11.95	6.65
Bi-LSTM† [154]	41.46	55.27	76.38	23.83	16.12	7.56
Bi-GRU† [155]	46.15	59.01	78.61	26.68	17.23	7.79
BERT +XMLC [25]	35.50	50.95	76.06	20.75	14.99	7.58
RoBERTa +XMLC [41]	36.20	52.23	77.05	21.98	15.09	7.88
BERT-XMLC+CAB [25]	37.20	51.24	**78.98**	22.18	15.02	8.03
GalaXC [141]	43.27	51.47	67.50	24.23	14.53	6.50
JobXMLC (GraphSaint) [94]	39.16	51.73	73.99	22.28	14.88	7.22
JobXMLC (GraphSAGE) [93]	38.76	52.26	74.19	21.98	14.99	7.23
JobXMLC	**47.85**	**59.26**	74.53	**26.92**	**16.94**	**7.23**

Table 4.5: Results of JobXMLC along with state-of-the-art approaches on StackOverflow dataset. For RNN-based models (†), we have limited all model architectures to two layers.

average number of relevant skills per job. For instance, with about 19.98 relevant skills per job on average, the maximum P@30 would be 66%. Our model achieves a precision of 42.22% at P@30. It shows that the model is able to correctly identify an average of 13 out of 19.98 skills. This is relatively better than GalaXC [141], which identifies only 11 skills.

Dataset $\rightarrow$	**mycareersfuture.sg**					
Model	**N@5**	**N@10**	**N@30**	**N@50**	**N@100**	**MRR**
CNN	28.21	40.23	60.60	66.37	71.96	0.77
LSTM	29.27	40.66	59.43	69.61	71.53	0.70
Bi-LSTM	30.32	48.07	44.55	50.30	57.04	0.76
Bi-GRU	30.83	50.52	46.45	52.37	59.15	0.76
BERT-XMLC	28.05	38.81	57.62	64.68	71.28	0.83
BERT-XMLC+CAB	29.13	40.74	60.60	67.51	73.74	0.85
GalaXC	32.86	44.51	63.73	70.11	74.77	0.82
JobXMLC	**37.91**	**49.63**	**67.83**	**73.81**	**78.94**	**0.90**

Table 4.6: Normalized Discounted Cumulative Gain (NDCG) is represented by N and Mean Reciprocal Rank (MRR) comparison of JobXMLC along with State-of-the-art approaches on mycareersfuture.sg dataset.

Dataset $\rightarrow$	**StackOverflow**					
Model	**N@5**	**N@10**	**N@30**	**N@50**	**N@100**	**MRR**
CNN	31.35	42.89	65.13	69.44	76.56	0.82
LSTM	31.75	42.98	61.34	74.51	76.64	0.72
Bi-LSTM	34.86	51.65	45.39	56.94	63.88	0.81
Bi-GRU	32.75	55.47	49.32	52.78	60.19	0.82
BERT-XMLC	30.56	44.93	63.76	65.82	73.89	0.85
BERT-XMLC+CAB	34.84	41.96	66.57	73.88	78.43	0.88
GalaXC	32.67	45.31	67.59	75.72	79.47	0.86
JobXMLC	**38.65**	**48.85**	**70.92**	**81.69**	**85.24**	**0.93**

Table 4.7: Normalized Discounted Cumulative Gain (NDCG) is represented by N and Mean Reciprocal Rank (MRR) comparison of JobXMLC along with State-of-the-art approaches on StackOverflow dataset.

Table 4.6 and Table 4.7 reports NDCG and MRR values on mycareersfuture.sg and StackOverflow dataset. An MRR of 0.90 and 0.93 for mycareersfuture.sg and StackOverflow datasets show that, on average, the first relevant skill appears in the top one or two positions of the predicted skill list. High MRR value signifies that JobXMLC is effective at placing atleast one relevant skill near the top of its skill predictions for job descriptions. We observe a significant improvement in model performance as JobXMLC consistently achieves the best scores, with a relative improvement of up to 18.30% and 15.36% in terms of N@5 over the best baseline [141] on both datasets. We compare JobXMLC with existing graph-based methods such as GalaXC [141], which are more well-suited to handle short text inputs for product queries. In contrast, JobXMLC leverages word-level components, including syntactic roles (POS tagging) such as nouns and verbs present in each job and word importance (TF-IDF), which explains the long document from the perspective of text relevancy and structure. We believe that JobXMLC is generalizable across many other applications.

Inference time: Table 4.8 reports the training time (in hours) and prediction time (in milliseconds) for JobXMLC. We compare our model with leading deep extreme classifiers, as they represent the existing baselines for missing skill prediction tasks. Compared to leading deep extreme classifiers, BERT-XMLC, RoBERTa+XMLC, and BERT-XMLC+CAB, JobXMLC is upto 18X faster for mycareersfuture.sg dataset to train on a single GPU. JobXMLC is also 634X faster at prediction than deep extreme classifiers. This demonstrates JobXMLC ability to efficiently scale to many real-world datasets with thousands of skill labels.

Datasets $\rightarrow$	mycareersfuture.sg		StackOverflow	
Models $\downarrow$	**TT**	**PT**	**TT**	**PT**
BERT+XMLC	5.50	1200	1.63	350
RoBERTa+ XMLC	4.72	1200	1.24	350
BERT+XMLC+CAB	9.20	1200	4.86	350
JobXMLC	**0.51**	**1.89**	**0.31**	**1.71**

Table 4.8: Comparison of JobXMLC with stronger baselines. JobXMLC is faster to train than leading Deep Extreme Classifiers like BERT at training and prediction time. Here TT= Train Time (in hours), PT= Prediction Time (in ms).

Analysis on Implicit and Explicit skills: The required skills are depicted in job descriptions, both explicitly (mentioned directly in the job description) and implicitly (not stated in the job description but inferred from context). When all skills are explicitly listed in the job description, identifying the required skills is straightforward because they are explicitly stated as substrings, and a basic string-matching algorithm would suffice. However, it has been observed in the mycareersfuture.sg dataset that the degree of implicitness in required job skills is higher, which makes this task more challenging. We found that on average, 86.13% of skills in the mycareersfuture.sg dataset are implicit. Similar observations have been made in the literature [25]. Hence, we are interested in the relevance and implicitness of the retrieved implicit skills. Table 4.9 shows the Explicit and Implicit Metrics (EIM, RIIM, REIM) for the missing skill prediction task. JobXMLC outperforms BERT+XMLC+CAB on EIM, RIIM, REIM metrics (See details in Section 4.7) by 4.4%, 2.7%, and 28.41% respectively.

Metrics	EIM	RIIM	REIM
BERT+XMLC +CAB	115.89	64.60	25.73
JobXMLC	**121.09**	**66.36**	**33.04**

Table 4.9: Comparison of EIM, RIIM, and REIM metrics on JobXMLC on mycareersfuture.sg dataset. JobXMLC outperforms BERT+XMLC+CAB model on all EIM, RIIM, and REIM metrics.

4.9 Ablation Study

Warm and Cold Start Scenarios: Table 4.10 reports the results in warm-start and cold-start settings separately. JobXMLC achieve P@5, R@5, and MRR of 72.86, 18.26, and 0.89, respectively in cold-start scenarios. In warm start settings, skill labels that were partially revealed were omitted from both predictions and the ground truth during evaluation (Section 4.5.3). In warm-start scenarios, JobXMLC achieve P@5, R@5, and MRR of 75.76, 22.09, and 0.91, respectively using initial fastText embeddings. These results show that partially revealed skill labels have relatively higher metrics (precision, recall, MRR) as compared to cold-start settings.

Model	P@5	R@5	MRR
JobXMLC (cold-start)	72.86	18.26	0.89
JobXMLC (warm-start)	75.76	22.09	0.91

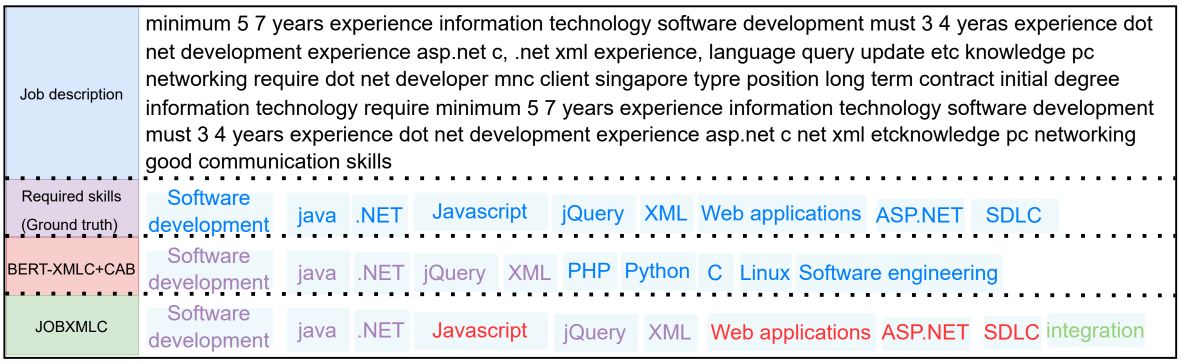
Table 4.10: Effectiveness of JobXMLC in warm-start and cold-start scenarios on mycareersfuture.sg dataset.

Initial embeddings: We observe that fastText embeddings were found to work better than BERT [148], DistilBERT [156], and Paraphrase-mini-LM-L6 [157] embeddings (Details in Table 4.11). The potential reason for the drop in performance could be that our raw dataset is relatively preprocessed, simple and misses predicate-argument structure dependencies. Therefore, we hypothesize that non-contextual embeddings such as fastText (having 98.69% of words from our dataset present in vocabulary) outperformed transformer-based models such as BERT in this scenario as it understands word-level information. Similar observations are made by [158] for classic embeddings with competitive (or even slightly better) performance than contextual embeddings. Table 4.11 reports the EIM, RIIM, and REIM measures for different Transformer-based [159] initializations.

Metrics	**EIM (Explicit inference measure)**	**RIIM (Relative implicit inference measure)**	**REIM (Relative explicit inference measure)**
BERT+XMLC+CAB	115.89	64.60	25.73
JobXMLC (with Mini-LM)	86.45	27.28	17.76
JobXMLC (with BERT)	84.07	25.77	15.59
JobXMLC (with RoBERTa)	86.45	24.12	15.02

Table 4.11: EIM, RIIM, REIM measures for JobXMLC and state-of-the-art approaches using Mini-LM, BERT and RoBERTa embedding initialization. Results from JobXMLC show that contextual models such as Mini-LM, BERT, and RoBERTa underperform the baseline with JobXMLC.

Qualitative Analysis: We compared JobXMLC with BERT+XMLC+CAB predictions and analyzed the skills predicted correctly and incorrectly as shown in Figure 4.3. BERT-XMLC with CAB predicts *'Java'*, *'C'*, *'Linux'*, *'Python'* as skills while more relevant skills such as *ASP.NET 'JavaScript'* and *'Web Applications'* are missed. In contrast, JobXMLC predicts more relevant skills such as *'Java'*, *'Software Development'*, *'XML'*, *'JavaScript'*, *'jQuery'*, etc. This shows that JobXMLC captures the structural relationships between jobs and skills effectively.



Job description	minimum 5 7 years experience information technology software development must 3 4 yeras experience dot net development experience asp.net c, .net xml experience, language query update etc knowledge pc networking require dot net developer mnc client singapore typre position long term contract initial degree information technology require minimum 5 7 years experience information technology software development must 3 4 years experience dot net development experience asp.net c net xml etcknowledge pc networking good communication skills
Required skills (Ground truth)	Software development, java, .NET, Javascript, jQuery, XML, Web applications, ASP.NET, SDLC
BERT-XMLC+CAB	Software development, java, .NET, jQuery, XML, PHP, Python, C, Linux, Software engineering
JOBXMLC	Software development, java, .NET, Javascript, jQuery, XML, Web applications, ASP.NET, SDLC, integration

Figure 4.3: Shows the skills predicted by BERT–XMLC+CAB and JobXMLC where input is job description. Purple shows correct skill predictions by JobXMLC as compared with required skills (ground truth). Green shows the extra skills predicted by JobXMLC. Red skills are missed by BERT+XMLC+CAB model as compared with ground truth.

4.10 Summary

In this work, we propose a JobXMLC framework, which uses a graph neural network to incorporate neighborhood information with the help of a collaborative graph over jobs and skills. JobXMLC leverages skill attention mechanism for more effective extreme classifiers and attends to multi-resolution representations of jobs and skills. JobXMLC also operate in warm and cold-start scenarios effectively. JobXMLC is 18X faster on training and 634X faster on predicting than deep extreme classifiers and can be scaled efficiently to real-world datasets with thousands of labels. We believe that JobXMLC can be deployed on large-scale recruitment platforms for predicting missing skills using job descriptions.

Chapter 5

Domain-specific Knowledge Extraction

"Knowledge is Power."

- Francis Bacon

While normalization (Chapter 3) and finding missing entities (Chapter 4) ensure consistency and completeness, they may not cover all the hidden nuances and unique context of domain-specific content. To improve the quality of enterprise applications, this valuable (domain-specific) information needs to be extracted from unstructured documents. In this[1] chapter, we present the framework to extract useful information from unstructured job postings data and convert it into a set of structured triples (subject, predicate, object). We discuss the end-to-end pipeline of building up the domain-specific knowledge graph from documents and provide its utility in identifying misleading facts (Chapter 6) and understanding Employment Scams (Chapter 9).

5.1 Introduction

Domain knowledge plays a crucial role in various enterprise applications, including entity recommendations [160], question answering [161], fraud detection [162], and knowledge discovery [163]. Almost all domain-specific knowledge on online professional platforms is in semi- or unstructured form (job postings, CVs, recruiter profiles), with the remainder in structured form. The primary objective of domain-specific information extraction is to extract important components such as entities (company, institutes, designations, certificates, skills, qualifications), relations (working hours, experience, type of job, etc.) and their

[1] Work presented in this chapter has been accepted in ACM Conference on Hypertext and Social Media. **Goyal, N.**, Sachdeva N., Choudhary V., Kar R., Kumaraguru P., and Rajput N. Con2KG-A Large-scale Domain-Specific Knowledge Graph. In Proceedings of the 30th ACM Conference on Hypertext and Social Media, pp. 287-288. 2019.

various attributes from unstructured content in job postings. For instance, Java developer is a designation, and Java is a skill that Java developers must have. Such valuable domain-specific information improves search efficiency and offers a competitive advantage to online recruitment platforms while enabling personalized recommendations to end users. Once we explore this massive resource of unstructured content and pinpoint the facts and entities, we need a way to store these domain-specific facts. Knowledge Graphs (KGs) are popular tools that provide a structured and de-facto solution to manage this complex domain-specific information. KGs contribute tremendously to many information retrieval applications such as search engines [164], AI assistants [77], and chatbots [165]. These can be categorized into Enterprise Knowledge Graphs such as Google [29], and Microsoft [166], and eBay [167] as well as Open KGs such as DBpedia [36], YAGO [34], and Freebase [38]. It can be visualized as a multi-relational structured graph, which stores triples by describing entities as nodes and different relations as edges between them. Online professional platforms such as LinkedIn and Indeed explore the utility of Knowledge Graphs in various crucial tasks in the recruitment business, such as personalized job suggestions [168], job search, candidate recommendation [169] and others that can leverage the connected data. Such Knowledge graphs provide a shared knowledge substrate within an organization, allowing various applications to use similar vocabulary and reuse definitions that others create. Furthermore, they usually provide a compact formal representation that developers can use to infer new facts and build up the knowledge [170].

Existing research focuses on building large-scale domain-specific knowledge graphs in industrial [81] and academic [82] settings. Despite the utility of KG in several domains, most of the existing Knowledge Bases (KBs) such as DBpedia [36], Freebase [38], YAGO [34], and NELL [78] provide limited information about domain-specific facts, important entities such as evolving skills, designations, and hidden properties of the job such as type of recruiter, shift timings, interview dates etc. Natthawut et al. [79] proposed an end-to-end framework for KG construction. However, the framework is limited to general concepts from day-to-day life and lacks domain-specific knowledge. Also, little is known about techniques to address the heterogeneity of information sources, such as evolving terminologies (skills, institutes, designations, job roles), varied requirements of different industries (shift timings, job type), etc., specific to the recruitment domain.

Given these challenges and limitations, we propose and demonstrate a framework for constructing an end-to-end large-scale recruitment domain-specific knowledge graph. It describes 4 million facts from 250 thousand job postings from a popular online recruitment platform. Figure 5.1 shows an example of recruitment domain-specific KG. The fact triples extracted include properties such as *has_work_location, needs_experience, has_skill, and has_job_role*, *offers_salary*, and *has_work_location* mapped from various heterogeneous information

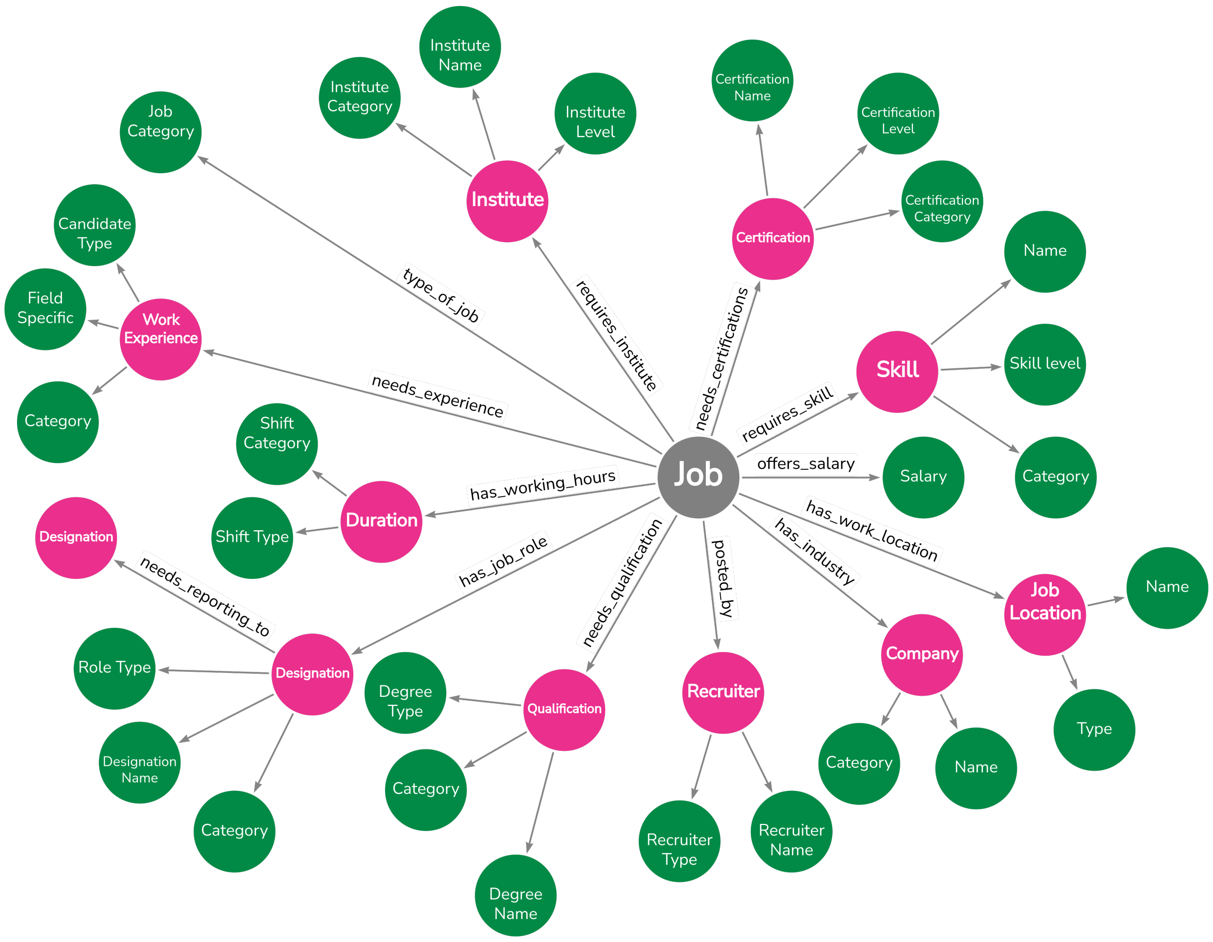


Figure 5.1: An overview of Online Recruitment Domain-specific Knowledge Graph. Nodes are represented as entities (job, company, certifications, skills, work experience, duration, designation, qualification, institute). Nodes represent entity attributes such as 'name', 'category', and 'type'; edges represent relationships such as 'requires_institute' and 'requires_skill' between two nodes.

sources. We develop a framework for the auto content population of underlying KB from both legacy jobs and growing active job pools. These KBs are supported via data-driven approaches for ontology mining. These ontologies help to identify edges between arbitrary sets of nodes, thus not limiting the graph to edges between any two nodes only. Therefore, it provides a hierarchical structure to the fundamental concepts (skills) to identify hidden relations and other possible semantic information. We also map the entity context and polarity while populating the KG in the graphical database. The proposed novel system can help researchers dynamically discover, visualize, and query the interesting relationships between any arbitrary combination of entities in the recruitment domain.

5.2 Contributions

The contributions of this work are:

- Con2KG exploits abundant information such as skills, companies, work locations, type of job, type of company, shift timings, important dates, designation, candidate experience, type of qualification (degrees, diploma), and salary into a structure that helps recruiters and job seekers to organize knowledge about the recruitment process.
- We provide a multi-tier framework for constructing a large-scale Knowledge Graph using structured and unstructured data gathered from various heterogeneous sources, including job postings, candidate profiles, and CVs.
- Con2KG offers a data-driven ontology mining of domain-specific concepts and represents nuanced meanings of an entity having different contexts and polarities.

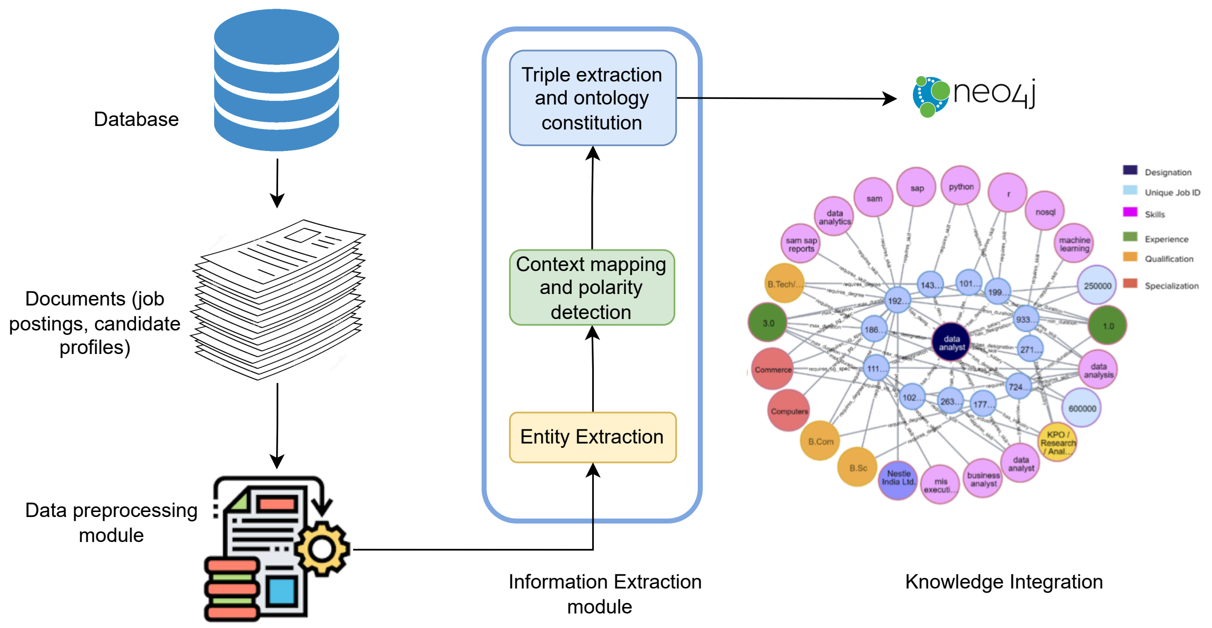

Database
Documents (job postings, candidate profiles)
Data preprocessing module
Triple extraction and ontology constitution
Context mapping and polarity detection
Entity Extraction
Information Extraction module
neo4j
Designation
Unique Job ID
Skills
Experience
Qualification
Specialization
Knowledge Integration

Figure 5.2: Proposed architecture of Con2KG system to build large-scale domain-specific knowledge graph.

5.3 Con2KG: Construction of Domain-specific Large-scale Knowledge Graph

This section provides a concise overview of the dataset and elaborates the proposed architecture in detail. Figure 5.2 gives an end-to-end pipeline for the proposed architecture of Con2KG, which consists of three main modules: Data pre-processing module, Information Extraction (Entity extraction, Context mapping and polarity detection, Triple extraction and Ontology constitution), and Knowledge Integration.

5.3.1 Dataset Description

In our dataset, we have gathered 1 million job postings from both legacy and active collections spanning over a period of 120 months. To ensure representative sampling, we randomly sampled 250 thousand jobs from these combined datasets. A job posting typically includes both structured and unstructured information. The unstructured part can have variations in content and format, with some containing candidate profiles while others may not.

5.3.2 Data pre-processing

In this module, we focused on pre-processing and cleaning the unstructured and noisy documents mainly job descriptions from our recruitment domain datasets. First, we pre-process HTML and Non-ASCII characters, followed by a sentence detection module to identify sentences. This module analyzes the unstructured documents (job descriptions) and separates them into individual sentences, effectively segmenting the content for further analysis. One of the typical challenges in the recruitment domain, as compared to others, is that the sentences present in job descriptions are ill-formed, complex, and ambiguous due to the nature of user-generated content. For instance, *“strong background in programming"* is a sentence fragment rather than a well-formed linguistically structured sentence. We address this limitation by employing part-of-speech approaches to identify such phrases using the Stanford Core NLP framework [171] and then revive missing phrases. Dependency parsers often rely on a sentence’s grammatical structure by identifying the syntactic relationships between words and representing them as a tree structure such as subject-verb, modifiers, and other linguistic dependencies. For example, we added phrases such as *“candidate needs"* to the fragment (*“strong background in programming"*) while preserving its original meaning. This helps to create a more favourable input environment for Core NLP dependency parsers. In turn, it enhances the accuracy, efficiency, and overall performance of the parsing process. We use a dependency parsing tree to capture syntactic and semantic relationships [171]. Finally, we exploit rule-based heuristics and leverage a proprietary vocabulary list to deal with abbreviations present in unstructured text.

5.3.3 Information Extraction

This module identifies domain-specific information from unstructured job descriptions. This process involves several sub-tasks, including Entity Extraction (Skills, Certifications, Companies, Institutes, Designations, Qualifications), context mapping, polarity detection and triple extraction while constituting an ontology.

Entity Extraction

In the context of the recruitment domain, a supervised approach requires a lot of labelled training data and is time-consuming. We are interested in the set of entities such as skills, companies, institutes, designations, work locations, important times, important dates, degrees, salary, type of company (MNC, PSU, Startup), type of job (work from home, internship, part-time), type of qualification (degree, diploma), recruiter type (company, consultant), type of candidate preferred (freshers, female only), graduation year preferred for the job. Therefore, we employ an unsupervised hybrid combination of Named Entity Recognition (NER), libraries, dependency parsers, and pattern-based heuristics for entity extraction. These components together help us extract important details from job descriptions. We explain our individual components in detail:

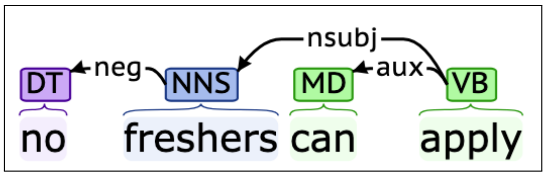
- Named Entity Recognition (NER) identifies and classifies entities such as organization names, locations, time expressions, day and date of interview, and monetary values (Salary) in unstructured text.
- Part-of-Speech (POS) Tagging assigns word categories (like nouns (NN), verbs (VB), adjectives (ADJ), adverbs (ADV)) in a sentence based on their roles.
- Dependency parsing help us to understand word relationships in a sentence using grammar rules. Using Dependency parsing, we formed the semantic structure of the sentence.
- Pattern-based heuristics are applied on dependency parsing results and extracted specific entities based on that rule.

Since we are interested in a specific set of entities, we looked at subtrees involving particular POS tags and dependencies. The Experience Extraction module identifies the experience associated with skills mentioned in the job descriptions. Given set of job descriptions $\mathcal{J}$ which contains a set of sentences $\mathcal{S}$, the aim is to extract the Experience (E) and its associated skill (S_k). For Example: *'Candidate needs experience of atleast 4 years in Java and Python*'. Here 'Java' and 'Python' are referred to as skills and '4 years' is the experience count associated with these skills. An experience can be mentioned in several ways in a job description. For instance, *{'2 yrs to 5 years of experience', 'one to 3 years of knowledge', '2-5 years of expertise', '2+ years of exp. in Java'}*. We looked for the dependencies which is a numerical modifier (nummod) with the dependent tagged with CD (cardinal number) and the governor, a Noun (from POS-tagging). The underlying assumption is that the sentence contains one of the keywords in the list a denoting time period (yrs, months, years). In this case, the dependent refers to the child, and the governor refers to the parent in the tree. The dependent denotes one of the few numerical values or ranges {'5+','5-6','5'} whereas the governor denotes the unit of time such as years or months. Next, we extract skill tokens from the sentence using an existing skill dictionary. We match the skill tokens from the sentence using the Keyword Processor in the FlashText Framework [2]. We also use hidden patterns for skills identification, one of them being a consecutive sequence of comma-separated Nouns. We extracted these nouns to build a skill list. Considering the existing limitations of dictionary-based systems, we enhance and evaluate our skills extraction system using the proprietary NER used by the company. Hence, we built a Skill Extraction system, which consists of three parts: a) Named Entity Recognition (NER), b) POS-tagger, and c) a dictionary list. This skill extraction system helps us find common skills in job descriptions in the unstructured form. Similarly, other entities are extracted using the combination of NER, POS tagging, dependency parsing, and pattern-based heuristics.

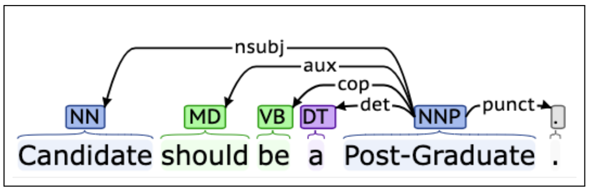
Context mapping and polarity detection

Once the entities are extracted from the entity extraction module, we needed a way to capture entity context and polarities. We identify entities with negative polarity connected with "negation" dependency. We tag entities with positive and negative polarities using dependency parsing. For instance, *"Candidate should be a Post-Graduate. No freshers can apply"*. We apply the entity extraction algorithm and remove 'Freshers'(negative polarity) from the entities extracted (Post-Graduate, Freshers). Furthermore, we enhance our module by incorporating contextual information such as preferred candidates (Experienced, Fresher), and ('Senior Backend Engineer','Experience-5+ years'). Figure 5.3 shows examples of dependency parsing

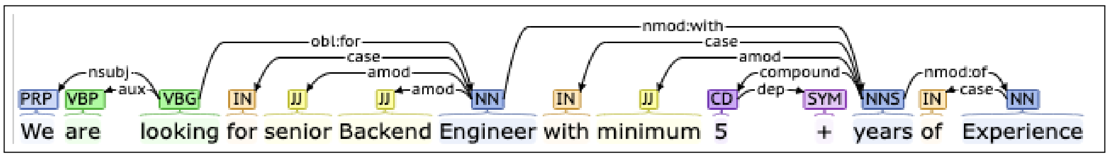
[2]https://github.com/vi3k6i5/flashtext



(a) Negative polarity in job requirement.



(b) positive polarity in job requirement.



(c) Context mapping: designation and experience in job requirement.

Figure 5.3: Context mapping and polarity detection of a fragment from a job description using Stanford CoreNLP dependency parser.

on a sentence fragment from the recruitment domain dataset.

Triple Extraction and Ontology Constitution

By first performing entity extraction, we identified entities with closed relationships based on specific rules. The structure of closed (target) relationships between entities is based on contextual cues and patterns in the job descriptions. However, there are some relationships which are less structured and do not have any predefined labels. We examine triple extraction to uncover meaningful relationships and structured information within unstructured text.

Triple Extraction: Triple Extraction uncovers the subject-predicate-object (SPO) form, where the subject and object are entities, and the predicate shows their relationship. Such relationships are open relationships which require more advanced natural language processing techniques. Open Information Extraction (Open IE) is a task to extract all types of relations found in unstructured job postings and is not limited to a small set of target relations known in advance. We analysed different OpenIE5 techniques [3, 172, 173]. For our dataset, Stanford OpenIE [173] performed well with single sentences but struggled with compound and numerical sentences. ClausIE [172] achieved higher precision and recall, but only for clauses and clause types. OpenIE5 [3] outperformed other techniques in both the quality and quantity of the triplets. OpenIE5 use pattern templates and semantic role labeling to extract triples (subject, predicate, object) in an arbitrary sentence. In this, we identify the relations and their associated arguments in a sentence without using either prior domain knowledge. For example, the sentence is *'Candidate should have experience of 4 years'*. Once we perform triple extraction, it gives *('Candidate', 'should have experience', '4 years')*.

Ontology Constitution: We employ a hierarchical structure to these extracted key

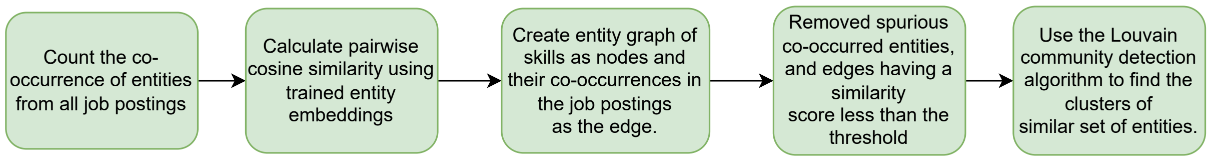

Count the co-occurrence of entities from all job postings
Calculate pairwise cosine similarity using trained entity embeddings
Create entity graph of skills as nodes and their co-occurrences in the job postings as the edge.
Removed spurious co-occurred entities, and edges having a similarity score less than the threshold
Use the Louvain community detection algorithm to find the clusters of similar set of entities.

Figure 5.4: Pipeline for extraction of key concepts.

concepts and hidden nuances. We manually curated some concepts related to the type of job and qualifications from the structured fields into the Ontology. Figure 5.4 elaborates a pipeline to find entity clusters such as ‘home-based jobs’, ‘freelancer jobs’, ‘online jobs’ for different types of jobs. We cluster entities to enhance our vocabulary for keyword extraction in entity and relation extraction modules. We apply the state-of-the-art clustering approaches [174] to these entities for capturing semantic information. More specifically, we cluster the semantically similar entities concerning recruitment domain data.

5.3.4 Knowledge Integration Module

We store all of the extracted triples in an efficient graphical storage. Neo4j [1] allows for flexible and dynamic schema design where we can add and remove properties to nodes and relationships without a fixed schema. Neo4j uses the Cypher Query Framework for querying, which is expressive for traversing and filtering through millions of nodes and relationships in a knowledge graph. Figure 5.5 shows an entity (Data Analyst) and its corresponding relationships fragmented from the large-scale knowledge graph.

5.4 Results and analysis

We identified popular entities such as 12,057 skills (Java, C, C#, python, etc.), 60 qualifications (B.Tech, Ph.D., M.A., etc.), 87,905 institutes (IIM’s, IIT’s, NIT’s, etc.), 1,100 certifications (Hadoop, J2EE, Cisco, etc.), 2,23,955 companies (TCS, Wipro, Infosys etc.), and 10,000 designations (Data Analyst, Software Developer, etc.). We also extract important attributes such as type of job (home-based, full-time, part-time), recruiter type (company, consultancy), and shift timings (night, day, alternate) for recruitment domain entities. Table 5.1 shows the entity statistics of the recruitment domain. We have $5,220$ unique relations linking $3,65,061$ entities extracted from unstructured and semi-structured heterogeneous data.

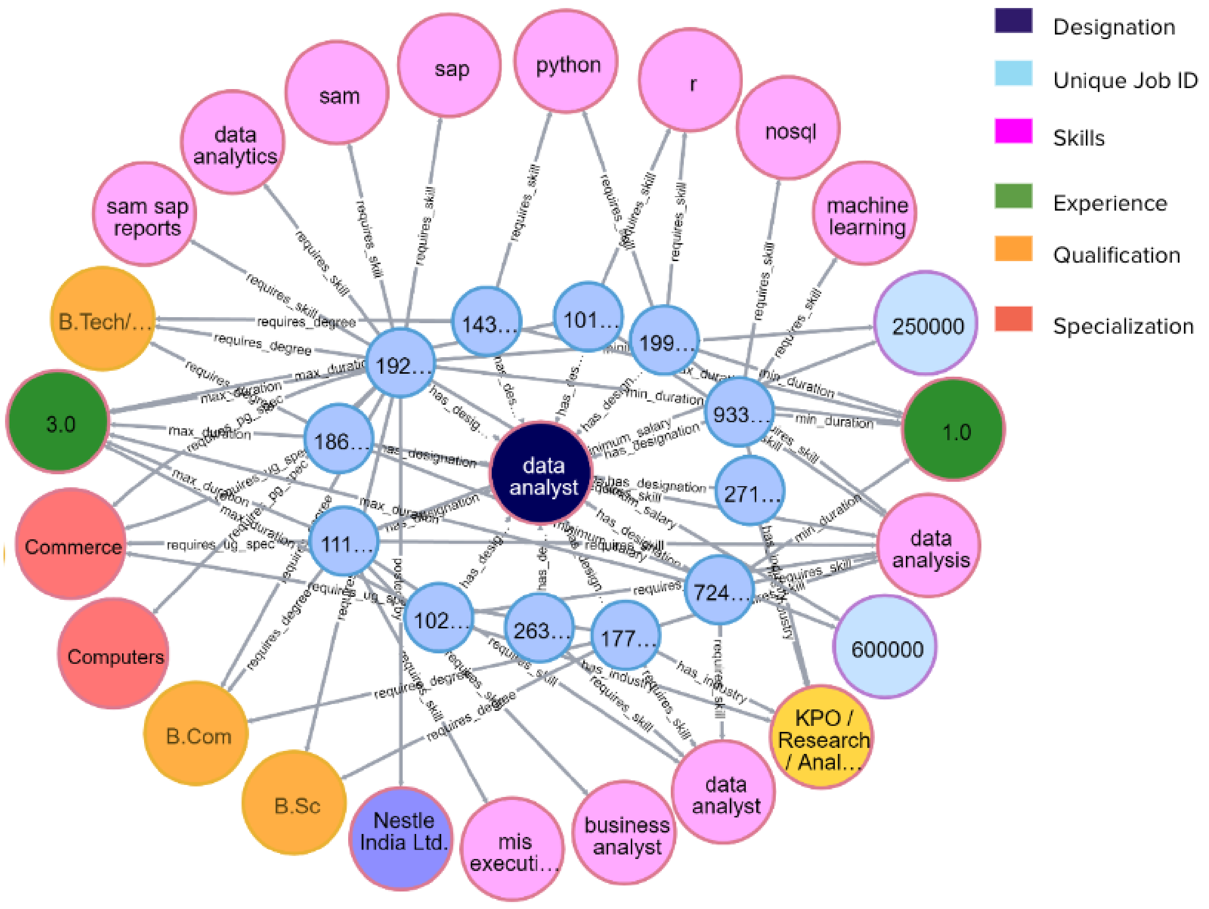


Figure 5.5: A snapshot of Data analyst (entity) from Large-scale Recruitment Domain Knowledge Graph visualized using Neo4j [1].

Entities	**Count**	**Entities**	**Count**
Skills	12,057	Institutes	87,905
Certifications	1,100	Designations	10,000
Companies	2,23,955	Qualifications	60
Total Entities	3,65,061	Total Relations	40,11,030

Table 5.1: Data statistics of entities (skills, institutes, certifications, designations, qualifications) of proprietary dataset from a popular recruitment platform.

5.5 Evaluation

Though many end-to-end information extraction pipelines have been developed over the last decade, yet there is a lack of a well-defined, large-scale, annotated domain-specific gold standard dataset for an objective. Consequently, these systems are predominantly evaluated manually on small-scale corpora that consist of only a few hundred sentences [79]. To this end, we randomly selected 310 jobs from our legacy dataset containing 4719 sentences to evaluate the quality and quantity of the triples extracted. A similar evaluation technique has been used in the literature [79]. We assess these triples and found 82% precision, 68.23% recall, and F-measure of 74.46% (Table 5.2).

Test Jobs	**Sentences**	**Precision**	**Recall**	**F1-measure**
310	4,719	82	68.23	74.46

Table 5.2: Evaluation results (Precision, Recall, and F1-measure) on 310 test jobs from a popular recruitment platform.

5.6 Error analysis

We analyzed the quality of triples from OpenIE system [3] in Table 5.3. We observe that triple extraction causes 0.05% errors due to incomplete triples and 0.20% due to no triple extraction for most of the sentences. We have shown these examples of erroneous triples in Table 5.3. Based on the preliminary analysis, errors in triple extraction occurs due to complexity in unstructured text and relations which are not identified clearly. Apart from these challenges, we still achieve approximately 74% F1-measure both in terms of quantity and quality. Con2KG can exploit the entity and its relationships to predict and personalize the job suggestions in the

Error Category	Sentences from job	Extracted triples
Incomplete	Proficiency in using query languages such as SQL, Hive, Pig.	0.38 (Hive; [is]; Pig)
Incomplete	Job will Identify, clean, and combine data to solve relevant business problems	0.56 (job; will Identify;)
No Triple Generation	Proficiency in using query languages such as SQL,Python,Java.	-
No Triple Generation	Job requires to monitor and track project progress.	-

Table 5.3: Examples where incomplete triples or no triples are generated from sentences present in job by OpenIE5 [3]. '-' denotes that there was no triple extracted from by OpenIE system.

recruitment domain using Entity Cards [175]. We can also query Con2KG and the complex data in real time to provide smart knowledge and detect false facts.

Chapter 6

Identify misleading information on online professional platforms

"Beware of false knowledge; it is more dangerous than ignorance."

- Bernard Shaw

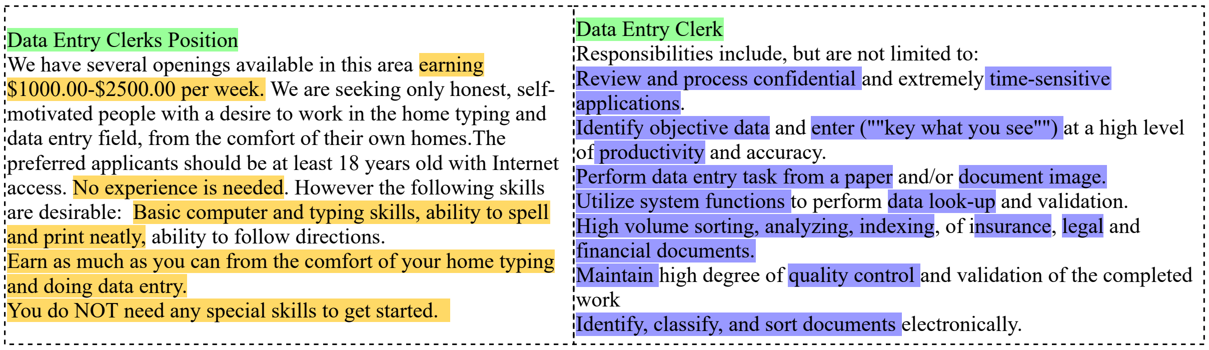
In Chapter 3 and Chapter 4, we investigated and designed solutions for two content quality issues: inconsistent variations and missing information in job postings. Another quality issue occurs when candidates find misleading jobs during the online job search process especially on legitimate job platforms where companies (which are hiring) ask for money to conduct interviews for candidates. Job seekers came across find jobs containing misleading (untenable) facts about domain-specific entities, such as mismatch in skills, offered compensation, and flexible working hours.

This[1] chapter focuses on identifying such misleading facts, posing it as a fraudulent jobs detection task in the recruitment domain. To accomplish this, we designed an end-to-end framework for extracting and managing complex domain-specific knowledge present in unstructured documents (such as job descriptions), which allowed us to build large-scale knowledge graphs in the recruitment domain in the previous chapter (Chapter 5). While domain-specific knowledge graphs are helpful for many enterprise applications such as chatbots, search engines, and AI assistants, little is known about their utility in identifying misleading information.

[1]Work presented in this chapter has been accepted in KSEM 2021. **Goyal, N.**, Sachdeva, N., and Kumaraguru, P. Spy The Lie: Fraudulent Jobs Detection in Recruitment Domain using Knowledge Graphs. In 14th International Conference on Knowledge Science, Engineering and Management (KSEM 2021).

6.1 Introduction

Online professional platforms such as Glassdoor, Indeed.com, and LinkedIn attract millions of job seekers per month. These platforms have transformed the way recruitment takes place, making it easier for candidates to find suitable positions and for companies to reach a broader talent pool. Unfortunately, in this ecosystem, candidates often find misleading jobs offering competitive salaries, flexible working hours, unrealistic skill requirements and career growth opportunities. While the motivations behind misleading jobs vary, it is important for job seekers and these platforms to remain vigilant as fraudsters often use legitimate job platforms to create accounts and advertise job postings maliciously.[2] Misleading job postings can lead to fraud when they involve intent, false representation, financial gain, harmful consequences, legal violations, and deceptive tactics. Fraudulent jobs[3] are dishonest, money-seeking, intentionally and verifiably false, that mislead job seekers. Federal Trade Commission (FTC) registered 101,917 fraud complaints[4] from job seekers over the period of 2014 to 2019. The proliferation of fraudulent jobs not only hamper candidate's experience [176] but also act as a repressing factor[5] in an enterprise's reputation. Therefore, it is desirable to detect and take off these fraudulent jobs. Recruitment platforms issue guidelines[6] to combat these fraudulent jobs through man-


Data Entry Clerks Position
We have several openings available in this area earning $1000.00-$2500.00 per week. We are seeking only honest, self-motivated people with a desire to work in the home typing and data entry field, from the comfort of their own homes.The preferred applicants should be at least 18 years old with Internet access. No experience is needed. However the following skills are desirable: Basic computer and typing skills, ability to spell and print neatly, ability to follow directions.
Earn as much as you can from the comfort of your home typing and doing data entry.
You do NOT need any special skills to get started.

Data Entry Clerk
Responsibilities include, but are not limited to:
Review and process confidential and extremely time-sensitive applications.
Identify objective data and enter (""key what you see"") at a high level of productivity and accuracy.
Perform data entry task from a paper and/or document image.
Utilize system functions to perform data look-up and validation.
High volume sorting, analyzing, indexing, of insurance, legal and financial documents.
Maintain high degree of quality control and validation of the completed work
Identify, classify, and sort documents electronically.

Figure 6.1: Examples of job postings: a) fraudulent job on the left and b) legitimate on the right. These job postings are taken from publicly available datasets.

ual review, user reporting, verification processes, content guidelines, and user education. They also encourage user reviews, offer customer support, and take legal action when necessary to maintain platform integrity and protect job seekers. However, such mechanisms need user in-

[2]https://www.linkedin.com/pulse/navigating-web-scammers-linkedin-my-experience-job-offer-quinones-oceje
[3]https://www.consumer.ftc.gov/articles/0243-job-scams
[4]https://www.aarp.org/money/scams-fraud/info-2020/ftc-job-scams.html
[5]https://hrdailyadvisor.blr.com/2015/01/19/what-is-recruitment-fraud-is-your\protect\@normalcr\relax-company-at-risk/
[6]https://www.naukri.com/imposter/report-fake-job-recruiter

volvement and heavily rely on reporting that may lead to oversights, particularly for eager job seekers.
Previous approaches mainly focus on supervised machine learning (e.g., ANN, bagging ensemble methods, and random forests) based on handcrafted feature engineering to detect fraudulent jobs [177]. These approaches mainly used binary [4], linguistic, string-based [68], writing styles [178], and textual features of job postings. Many of these approaches rely on rule-based methods and split public test datasets to enhance their performance but these methods face challenges in scaling effectively for real-world datasets. However, these methods ignore the important factual information [179] among domain-specific entities (salary, experience, etc.) present in job postings, such as inexperienced/unqualified candidates being offered high salaries. Figure 1 shows that the [left] job is fraudulent, which mentions implausible facts such as {*'offering very high weekly salary- $1000 $2500'*, *'No experience required'*, *'Earn as much as you can'*} for *Data Entry Clerks position*. In contrast, the [right] job is legitimate and covers genuine facts related to role and responsibilities such as {*'Review and process confidential time-sensitive applications'*, *'Identify, classify, and sort documents'*} for the same position (i.e., Data Entry Clerk). Another example of a job that contains implausible facts:

> 'HIRING for Reliance Jio/ Paytm/ Snapdeal/ Flipkart/ Amazon. Candidate must have 0 year of experience in the Customer Voice Process (Call Center). Candidate having prior experience in the Customer Service Industry will be an advantage. Candidates with backlogs/arrears in their degree can also apply.'

This example contains implausible facts such as *'Candidate must-have 0 year of experience in the Customer Voice Process (Call Center)'* and *'Candidates with baclogs/arrears in their degree'*). Encouraging candidates with backlogs or arrears in their degree to apply is misleading for reputable companies (Reliance Jio/Paytm). These companies usually have higher academic standards for their candidates and require some level of experience for customer service roles, even at entry-level positions. Checking the plausibility of these facts can be challenging even for human experts, as it requires domain-specific knowledge rather than relying solely on content. Such domain-specific knowledge is primarily stored in form of facts[7] or triples (subject entity, predicate, object entity) or (subject entity, attribute, literal value) in knowledge graphs. For instance, {(*Data Scientist*, *need skill*, *Python*)} or {(*Data Entry Clerk*, *requires_experience*, 0-1 *year*), (*Data Entry Clerk*, *weekly_salary*, $55-$65)}. To check plausibility of these facts, we would need a domain-specific knowledge graph that contains these facts. Therefore, we construct two knowledge graphs using the information present in both legitimate and fraudulent job postings which is an effective way when there is no external publicly available domain-specific

[7]Triples and facts are used interchangeably.

knowledge graph. One significant limitation of knowledge graphs is the incompleteness problem due to emerging new relations, where entities may be present in the knowledge graph but not their corresponding relations. For instance, the extracted triple for the position (*Data Entry Clerk*, *requires_experience*, 0 *year*) may include the entities {Data Entry Clerk, Experience (value)} in the graph but not the relationship 'require_experience'. In this work, we make use of knowledge graph embedding techniques like TransE [85], TransH [86], and HolE [123], etc. to compute semantic similarities and check the plausibility of these facts from the knowledge graph. These are well-known and widely preferred techniques for fact-checking [99]. Additionally, we do not rely on knowledge-based features alone but also utilize the unstructured job description content and additional information (job length, type of job, etc.) related to job postings to obtain contextual and metadata features. Towards this end, we present a multi-tier, novel end-to-end framework called FRaudulent Jobs Detection (FRJD) Engine, which considers a) fact validation module using KGs, b) contextual module using deep neural networks, and c) meta-data module to capture the semantics of job postings. We conducted our experiments using a fact validation dataset containing 4 million facts extracted from job postings. The extensive evaluation shows that FRJD yields a 0.96 F1-score on the curated dataset of 157,880 job postings. Finally, we provide insights into the performance of different fact-checking algorithms on recruitment domain datasets.

6.2 Related Work

This section describes the related literature on fraud detection in domain-specific scenarios and in general.

- **Domain-specific Scenarios.** Recent research [68, 177] focuses on content-based approaches that use handcrafted features such as empirical rulesets (binary, categorical, string-based) and Bag-of-words to identify fraudulent jobs in the recruitment domain. A research study [180] also reported that job seekers are unable to distinguish between genuine and fake postings. Works [4, 181] conducted the research using behavioural activity or binary features as context. Kim et al. [182] propose hierarchical clustering deep neural network to detect fraud in the work processes of job placement automatically.

- **Fact checking.** Most fact-checking methods rely on experts, such as journalists or scientists, to assess the content and the crowd's wisdom [113]. Fact-checking for jobs posted on social networks also relies on crowdsourcing and human experts to detect false claims/facts.[8] The propagation of these postings across multiple platforms such as WhatsApp

[8]https://www.vishvasnews.com/english/scam/fact-check-fraudulent-job-offers-going-viral-on-social-media/

aids in verification through crowdsourcing. However, this dynamic interaction and propagation are less common on online recruitment platforms, where job postings are often perceived as more credible. These expert-based methods are expensive as they require hiring specialists, and they are limited in number and unable to handle all the content being produced. Automatic fact-checking algorithms [100, 183–185] leverage unstructured and structured knowledge sources such as Wikipedia, DBpedia [36], and Wikidata [101] which contain a vast collection of factual knowledge. Next, we discuss various fact-checking algorithms in knowledge graphs.

  – *Path-based algorithms:* Existing path-based approaches [98, 186, 187] focus on utilizing the paths of relationships in a knowledge graph to infer new relationships. These methods are used for fact-checking by evaluating the probability of a fact being true based on the discovered paths. Such techniques are interpretable and can explicitly reason over the graph structure. However, they often struggle with scalability and handling noisy or incomplete data.
  – *Embedding-based algorithms:* Research studies [96, 99, 188] demonstrate how various knowledge graph embedding models (TransE [85], ProjE [188]) are applied to tasks that include fact-checking or validating the plausibility of triples/facts within knowledge graphs. Another set of approaches [86, 87, 189, 190] focus on either triple classification or link prediction in external KGs (DBpedia [36], Freebase [38]).

- **Content-based approaches.** Research explores the textual content using TF-IDF [191], stylometric [192], and RNN (recurrent neural networks) [193, 194]. Some approaches [195] exploits the graph-based techniques for fake content detection while others [196, 197] use contextual embedding models such as ELMO and BERT [156] to learn language-based features.

	Content	**Knowledge**	**Context**
Alghadmi et al. [177]	✔		
Vidros et al. [68]	✔		
Mahbub et al. [4]			✔
Nindyati et al. [181]			✔
Our work (FRJD)	✔	✔	✔

Table 6.1: Different kinds of features used in related literature for fraudulent jobs detection task.

Despite the popularity of content and knowledge graph-based approaches, these are still under-explored in domain-specific scenarios such as the recruitment domain. Therefore, we need a

solution tailored to authenticate job postings on these online recruitment platforms, especially when there is limited information available in external knowledge bases [36, 38].
Our research is uniquely inclined towards using a hybrid approach consisting of the knowledge graph, contextual, and meta-data features simultaneously requiring no job seeker responses and providing different insights on fact-checking algorithms. We compare the most relevant studies with our work in Table 6.1.

6.3 Contributions

The contributions of this work are:

- We propose a multi-tier novel unified framework called FRJD, which employs a fact-checking module using knowledge graph representations, a contextual module using deep neural networks and considers unique meta-data properties of job postings to accomplish fraudulent jobs detection tasks.
- We study the fact validation dataset that consists of 4 million facts in the form of entities and relationships and utilize it for the triple classification task.
- Extensive experiments on real-world recruitment domain datasets demonstrate the promising performance of FRJD compared to state-of-the-art models.

6.4 Concepts and Terminologies

This section introduces some terminologies used in this work and then formulate the studied problem.

- **Job postings:** Job postings refer to the associated job content posted on online recruitment platforms. Job postings can be represented as $\mathcal{J}$={ $\mathcal{J}_1$, $\mathcal{J}_2$,........,$\mathcal{J}_n$} where n is the total number of job postings.
- **Labels:** Every job posting in $\mathcal{J}$ has its corresponding labels in $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ such that $y_i \in \{0, 1\}$.
- **Meta features:** These are the set of features such as *number of skills mentioned in a job*, *job length*, *educational degree (graduation, post-graduation, etc.)*, *job location*, *telecommuting (e.g., Work from home position)*, and *employment type (e.g., full-time, part-time)*

extracted from a job posting $\mathcal{J}_i$. The meta-features such as number of skills mentioned in the job and job length are extracted from the job description (unstructured text present in the job posting) and other features are extracted from structured fields. These features are concatenated together and are represented as a vector m^i (See Section 6.6 for details).

- **Contextual features:** Contextual features are generated from raw job description text extracted from $\mathcal{J}_i$. and are represented as c^i. To derive contextualized representations, job descriptions are fed through Bidirectional Encoder Representations from Transformers (BERT) [159]. Additional details are provided in Section 6.6.

- **Triples:** For every $\mathcal{J}_i$, we extracted a set of triples $\mathcal{T}^i$ where $\mathcal{T}^i$ =$\{t_1^i, t_2^i, t_3^i, \ldots\ldots, t_k^i\}$ and $k > 0$; using OpenIE [3]. A triple $t_j^i \in \mathcal{T}^i$ is of the form (subject (s), predicate (p), object(o)) where $(s, o) \in \mathcal{E}$, i.e., the set of entities and $p \in \mathcal{P}$, i.e., the set of relationships (See details in Section 6.6).

6.5 Problem Formulation

Given a job posting $\mathcal{J}_i$ and its corresponding set of extracted facts $\mathcal{T}^i$, contextual vector c^i, and meta vector representation m^i. The goal of Fraudulent Job Detection Engine (FRJD) is to check if a job posting is legitimate or fraudulent based on its context, meta-features and fact-checking module. The Task of Fact checking is to check if a target triple t in $\mathcal{T}^i$ is likely to be true based on the given knowledge graphs ($\mathcal{KG}_{true}$, $\mathcal{KG}_{false}$). From fact checking module, we obtain a scoring function for each triple $t^i \in \mathcal{T}^i$ extracted from $\mathcal{J}_i$. Finally, we learn the function φ where φ: $\mathcal{F}$ ($\mathcal{KG}_{false}$ $(T)^i$, $\mathcal{KG}_{true}$ $(T)^i$, c^i, m^i). φ integrates the bias scores from knowledge graphs trained on fraudulent and legitimate job postings along with contextual and meta-features.

6.6 Fraudulent Jobs Detection Engine (FRJD)

This section describes our novel multi-tier framework- Fraudulent Job Detection Engine (FRJD), using knowledge graphs and deep neural networks. Figure 6.2 depicts the overall architecture for detecting fraudulent job postings. This framework consists of three components a) *Fact-checking module*, b) *Contextual embedding generation*, c) *Meta-features generation.*

Fact-checking module. This module is designed to detect whether a job is fraudulent or legitimate based on the triples extracted from the job postings. For instance, the triples ex-

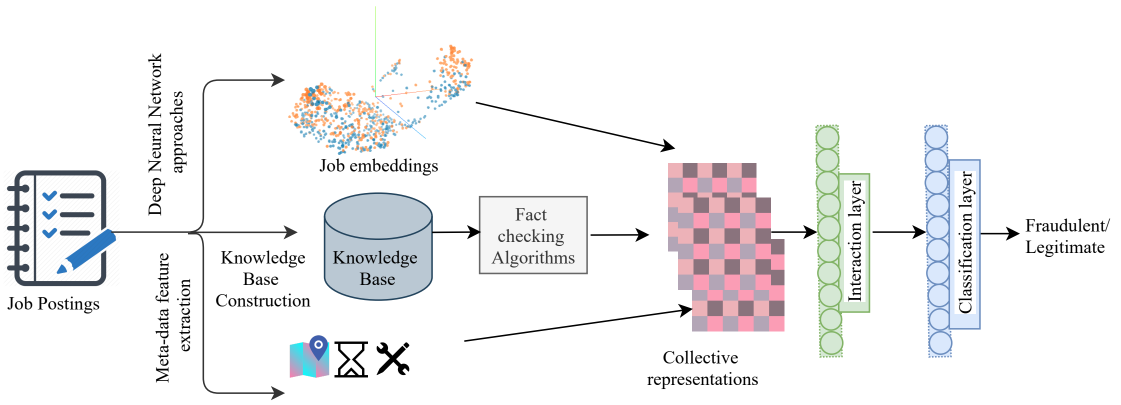


Figure 6.2: An overview of our proposed framework- FRaudulent Jobs Detection Engine (FRJD).

tracted from a legitimate job posting are (*Data Scientist job position*, *requires_skill*, *Python*); (*Data Scientist job position*, *located_in*, *San Francisco*); (*Data Scientist job position*, *offers_salary*, \$120,000); (*Data Scientist job position*, *requires_experience*, *5 years*). Similarly, the triples extracted from a fraudulent job posting are (*Data Scientist job position*, *located_in*, *remote*); (*Data Scientist job position*, *offers_salary*, *\$500,000*); (*Data Scientist job position*, *requires_experience*, *1 year*). Here, the triples offering an unusually high salary for very little experience are implausible. Such triples are crucial because they represent relationships between entities in a job posting making it easier to spot implausible patterns. Therefore, we use these triples to construct two domain-specific knowledge graphs: $\mathcal{KG}_{false}$, which contains triples extracted from fraudulent job postings, and $\mathcal{KG}_{true}$, which contains triples extracted from legitimate job postings (Figure 6.1 for examples). In order to extract triples, we pre-process the job postings and follow the methodology discussed in Chapter 5, Section 5.2 to construct these knowledge graphs. Thereafter, we obtain the embeddings to represent an entity (Skill, Company, Designation, Experience, Location, etc.) or relation (requires_skill, experience_needed, located_in, salary, etc.) as a low-dimensional vector using different knowledge graph embedding (KGE) methods also known as fact-checking algorithms [85, 189, 190, 198, 199]. Studies have demonstrated the effectiveness of knowledge graph embedding techniques in serving as fact-checking algorithms [96, 99, 188]. These embeddings capture semantic and relational information about entities to infer missing relationships or assess the plausibility of triples in the embedding space. To measure the plausibility of these triples, we train models using the TransH (See Section 6.8 for details) based on the triples present in $\mathcal{KG}_{true}$ and $\mathcal{KG}_{false}$.
Given the model and a triple (s, p, o), the embedding of s and o are first projected onto relation-specific hyperplane (the normal vector) w_p. The projections are denoted as $\mathrm{s}_\perp$ and $\mathrm{o}_\perp$, respec-

tively (Equation 6.1).

$$b^i_{triple} = \left\|s_\perp + d_p - o_\perp\right\|_2^2 \text{ where } s_\perp = s - w_p^\top s w_p, \quad o_\perp = o - w_p^\top o w_p \tag{6.1}$$

Here, $\|.\|$ denotes the L2 norm and d_p denotes the relation-specific translation vector.
The bias score, b^i_{triple} between the shifted subject entity $s_\perp$ and the object entity $o_\perp$ on the hyperplane is minimized for true triples obtained from legitimate job postings. The computed bias score b^i_{triple} measures the plausibility of a triple (s, p, o) that it is likely to be true indicating that the entities $(s,\ o)$ fit well together under the relation p. Lower bias score indicate higher plausibility. This score helps differentiate between plausible and implausible triples, which in turn helps classify job postings as legitimate or fraudulent.
Therefore, we compute two bias scores, b^i_{true} and b^i_{false}, i.e., b^i_{triple} obtained from models trained on $\mathcal{KG}_{true}$ and $\mathcal{KG}_{false}$, respectively. We employ a margin-loss function within our fact-checking module, separately training each model according to the methodology outlined in the TransH [86]. This process is repeated $\forall\ \mathcal{T}^i$ of $\mathcal{J}_i$ to obtain a vector $(b^i_{true})_{1:|T^i|}$ and $(b^i_{false})_{1:|T^i|}$ where $|T^i| > 0$. Once these scores are calculated, the final scoring function, $\mathcal{KG}^A_{true}\ (T)^i$ and $\mathcal{KG}^A_{false}\ (T)^i$ for a job posting $\mathcal{J}_i$ are described in Equation 6.2 and Equation 6.3.

$$\mathcal{KG}^A_{true}(\mathcal{T}^i) = \frac{\sum_{\gamma=1}^{|\mathcal{T}^i|}(b^i_{true})_\gamma}{|\mathcal{T}^i|} \tag{6.2}$$

$$\mathcal{KG}^A_{false}(\mathcal{T}^i) = \frac{\sum_{\gamma=1}^{|\mathcal{T}^i|}(b^i_{false})_\gamma}{|\mathcal{T}^i|} \tag{6.3}$$

Here, $|\mathcal{T}^i|$ is the total number of triples for the job posting $\mathcal{J}_i$. $(b^i_{true})_\gamma$ and $(b^i_{false})_\gamma$ represents the bias score of the γ-th triple in $\mathcal{T}^i$ according to the model $\mathcal{KG}^A_{true}$ and $\mathcal{KG}^A_{false}$. Here, A is the TransH algorithm and γ is an index variable used in the summation. γ iterates over all the triples in the set $\mathcal{T}^i$.

Finally, we fuse both $\mathcal{KG}^A_{true}(T)^i$ and $\mathcal{KG}^A_{false}(T)^i$ to obtain a representation vector f^i (Equation 6.4).

$$f^i = \mathcal{KG}^A_{true}(T)^i \bigoplus \mathcal{KG}\_false^A(T)^i \tag{6.4}$$

When we classify based on obtained scores (using only fact-checking module), $KG^A_{true}(T)^i \leq KG^A_{false}(T)^i$ for triple set $\mathcal{T}^i$ which means that the job is likely to be legimitate based on the scoring function.

Contextual embedding generation. We employ a pre-trained deep neural network, i.e., BERT [156] to generate contextual features for all job postings. Generally, fraudulent job descriptions have distinct characteristics such as language, style, and vagueness. BERT effectively captures these characteristics and context well to create meaningful semantic representations. Research suggests that for real-time applications, a lightweight model should be preferred [200–202]. Hence, we use the distilled version which requires fewer parameters, less space and time complexity while retaining the 97% performance of BERT. Finally, we obtain contextual vector representation c^i for each job posting $\mathcal{J}_i$ (Equation 6.5).

$$c^i = BERT(\mathcal{J}_i) \tag{6.5}$$

Meta-features generation. We consider the meta-information such as the number of skills mentioned, job length, educational degree (graduation, post-graduation, etc.), job location, telecommuting (work from home position or not) and employment type (full-time, part-time) from our public dataset. The importance of these features have been demonstrated in the literature [68]. For example, Fraudulent jobs tend to be shorter than legitimate jobs (See Table 6.2 for details). This characteristic can be used as a distinguishing feature between fraudulent and legitimate job postings. Similarly, average skills per fraudulent job are less than average skills per legitimate job. We use min-max normalization of these features to ensure that all features are on a similar scale. This helps reduce the influence of features with larger ranges and ensures that each feature contributes equally to the model's training process. After extracting these features, we obtain a fused representation m^i such that m^i= [m^i_1, m^i_2, m^i_3, m^i_4,.....,m^i_k] where k is the number of meta-features extracted from job posting $\mathcal{J}_i$. The given list of meta-features can be further extended based on the availability of information in recruitment domain datasets.

Finally, we concatenate the factual, contextual, and meta representations to form $\mathcal{F}$ and pass them through the fully connected neural network layers.

$$\mathcal{F} = \{f^i \oplus c^i \oplus m^i\} \tag{6.6}$$

To avoid overfitting, we use the Rectified Linear Units [203] as the non-linearity for faster training and drop-out layers. We apply the sigmoid ($\sigma(.)$) layer and binary cross-entropy loss to classify the job postings into legitimate and fraudulent. We use *ADAM* as an optimizer [204] to handle sparse gradients.

6.7 Experiments

This section elaborates on datasets, and evaluation metrics and compares FRJD against several baselines for classifying fraudulent job postings.

6.7.1 Datasets

We utilize two real-world recruitment datasets: proprietary, jobs dataset collected from a popular recruitment platform and the Employment Scam Aegean Dataset, a publicly available dataset of job postings. In recruitment domain literature, there is a scarcity of publicly available datasets of fraudulent and legitimate job postings.

- **Proprietary Dataset.** We use the real-world job posting dataset from one of the largest online recruitment platforms in India. We curated a balanced dataset by sampling 70K Legitimate and 70K Fraudulent job postings from the legacy database annotated by domain experts. Table 6.2 reports the statistics on our proprietary dataset. Due to the proprietary nature of the dataset, some analyses and insights related to the dataset has been redacted in this work.

Statistic	**Count**
# of Fraudulent Jobs	70K
# of Legitimate Jobs	70K
Avg. words per Fraudulent job	70
Avg. words per Legitimate job	231
Avg. skills per Legitimate job	12
Avg. skills per Fraudulent job	9
# of entities	37.5K
# of relations	4.5K
# of triples	4M

Table 6.2: Statistics of Fraudulent and Legitimate jobs on the proprietary dataset.

- **Public Dataset.** The Employment Scam Aegean Dataset (EMSCAD)[9] contains 17,014 legitimate and 866 fraudulent jobs. For all experiments, we apply class balancing techniques by penalizing the class (legitimate) having more samples [68].

- **Fact checking Dataset.** We create a fact-checking dataset of positive and negative triples using the standard methodology used in the literature of knowledge graph representations [86, 99]. As a result, our fact validation dataset consists of 4 million facts extracted from job postings using the methodology (OpenIE5) discussed in Chapter 5. Given a set of triples (s,p,o) where entity pairs are (s, o) and p is the predicate between them. We use these knowledge representations to map each entity to a v-dimensional vector and relation to a w-dimensional vector in the embedding space where v and w are hyper-parameters. We use OpenKE [205] toolkit implementation to obtain knowledge graph

[9] http://emscad.samos.aegean.gr/

representations. We implement popular fact-checking algorithms such as TransE [85], TransR [206], TransD [189], TransH [86], DistMult [207], ComplEx [199], HolE [190], and RotatE [87] from knowledge graph representation literature.

6.7.2 Baselines

- **J48 Decision Tree classifier [4]:** Authors include the following binary features (*hasCompanyName*, *hasCompanyWebsite*, *hasMaturedCompanyWebsite*, *hasLinkedInPage*, *previouslySeenAsFraudulent*) in their contextual feature space. The features in the entire feature space were divided into three major categories: textual features, structural features, contextual features and then WEKA's [208] J48 classifier which is an implementation of the C4.5 algorithm was used for classification. The hyperparameters and other details are as suggested by authors [4].
- **JRip rule-based classifier [4]**: WEKA's JRip, a Java implementation of the RIPPER algorithm was used which creates a set of classification rules by iteratively generating and optimizing rules that cover the positive examples in the dataset. The classifier was applied on the same feature space as obtained previously. JRip's rules are straight forward and are easily interpretable as compared to J48 whose criteria is based on minimizing the entropy. Some sample rules used by the authors are (*hasMoneyInTitle*, *hasNonOrgEmailLinks*, *educationLevelLow*) etc to make a decision. The hyperparameters and other details are as suggested by authors [4].
- **Naive Bayes Classifier [4]**: The Naive Bayes classifier is a probabilistic model that applies Bayes' theorem with the assumption of conditional independence between features. It calculates the probability of each class given the feature values and selects the class with the highest probability. In this study, the Naive Bayes classifier was used to predict whether a job posting is fraudulent or legitimate by analyzing the textual and contextual features of the postings. The classifier was trained and tested using a 10-fold cross-validation on the EMSCAD dataset.
- **Random Forest Classifier [68].** The approach consists of a ruleset-based binary classifier which consists of three categories: linguistic, contextual, and metadata. We report the results of the model trained on the empirical ruleset against the complete imbalanced dataset of 17,880 job postings in the public dataset (reported in published work). On other datasets, we use the same rulesets to report the results.
- **Logistic Regression.** Logistic regression is a statistical model that is popular for classification as well as regression tasks. We model the textual features using a count-vectorizer

and perform classification using Logistic Regression.

- **Support Vector Machines [209].** SVM is a supervised machine-learning algorithm widely used for binary classification tasks. We use a spacy tokenizer to clean the text present in job description and utilize the count-vectorizer to create textual features to train SVM.

6.7.3 Evaluation metrics

- **Mean Reciprocal Rank (MRR):** MRR is the average of the reciprocal ranks of all the triples. The value ranges from 0 to 1. The higher the value, the better the algorithm. Equation 6.7 shows the formula for MRR calculation.

$$\mathcal{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank(s,p,o)_i} \tag{6.7}$$

  where $|\mathcal{Q}|$ is the set of all ranked predictions of the knowledge graph embeddings model, and rank (s,p,o) is the position of the triple in the list.
  The gold standard for the true triples has been prepared from legitimate job postings, annotated by domain experts.
  MRR is categorized into raw and filtered MRR. To evaluate the test set, we corrupt the subject and object with all the entities in the KG. We generate the corruptions only from a subset of entities in KG. The graph comprises different entity types such as skills, designations, companies, etc. We use a particular category to generate the corruptions for the object side. In raw MRR, we are not filtering the true positives, i.e., some of the corruptions may be ground truth triples observed during training. Training triples usually get a high score as they are ‘observed’ by the model. In filtered MRR, we filter out such triples. Therefore, a test triple would get a lower rank if such triples appear in corruption.

- **Hits@k:** Hits@k is the percentage of computed ranks that are greater than (in terms of ranking) or equal to a rank of k. The value ranges from 0 to 1. The higher the value, the better the algorithm. Equation 6.8 shows the calculation of Hits@k where k={1,3,10}.

$$\text{Hits@}k = \sum_{i=1}^{|Q|} 1 \qquad \text{if rank}_{(s,p,o)_i} \leq k \tag{6.8}$$

	MRR		**Hits @ k**		
Model	**Raw**	**Filter**	**k=1**	**k=3**	**k=10**
TransH	**0.52**	**0.69**	**0.63**	**0.73**	**0.82**
TransD	0.50	0.67	0.62	0.69	0.80
TransR	0.20	0.60	0.55	0.64	0.73
TransE	0.51	0.60	0.56	0.62	0.68
HolE	0.22	0.48	0.34	0.49	0.71
ComplEx	0.29	0.34	0.25	0.35	0.52
DisMult	0.30	0.40	0.30	0.40	0.50
RotatE	0.28	0.41	0.39	0.40	0.43

Table 6.3: Results of link prediction task using knowledge graph embedding algorithms on the proprietary dataset.

6.8 Results and Analysis

We provide insights on various Knowledge graph embedding methods/fact-checking algorithms for our datasets. These methods differ in their scoring functions, transformations of entity and relation embeddings, and triple-specific loss functions. Table 6.3 reports the results on link prediction using OpenKE [205]. We report the raw and filtered Mean Reciprocal Rank (MRR) and Hits@ (1, 3, 10) for all the models. Hits metrics are filtered (removal of the triples from the test list that appeared during evaluation in the dataset). TransH and TransD achieve significant performances on these metrics, i.e., filtered MRR (0.69 and 0.67) and on Hits@10 (0.82 and 0.80). We observe that TransH [86] outperforms all other methods for our dataset where it effectively handles complex relations by projecting entities onto relation-specific hyperplanes and better utilizes the one-to-many and many-to-many properties of the relation. For instance, a job position requiring multiple skills. Similarly, algorithms such as RotatE are unable to perform well due to the large number of many-to-many relations in our knowledge graph. These embeddings capture the semantic and relational properties of entities, evaluating the plausibility of triples to aid in fact-checking within the knowledge graph. These relationships generalize well and are suited to other downstream tasks like triple classification (M2 component). Table 6.4 presents the performance of FRJD as compared to state-of-the-art baselines. We are able to demonstrate that incorporating meta and factual features leads to better results for both datasets. The individual FRJD modules (M1, M2, and M3) show varying degrees of performance, with FRJD (M2) achieving the best individual performance F1 (0.92) compared to M1 and M3. When combined with meta, they achieve the highest performance. Table 6.5 shows that individually FRJD trained on M3 features exhibit lower precision due to However, their combination with other features yielded significantly higher overall precision (0.98).

Analysis on textual information: In Figure 6.3, we observe that the keywords such as (*'earn'*, *'no experience'*, *'extra'*, *'cash'*, *'earnings'*, and *'money'*) have a higher occurrence in fraudulent jobs as compared to legitimate ones which focus on keywords such as (*'opportunity'*, *'online'*).

Approaches	**Metrics**					
	Fraudulent			**Legitimate**		
	P	R	F1	P	R	F1
Logistic Regression [68]	0.90	0.90	0.90	0.86	0.60	0.70
Support Vector Machine [209]	0.57	0.83	0.67	0.98	0.97	0.97
Random Forest [68]	0.28	0.75	0.41	0.98	0.90	0.94
JRip rule-based classifier† [4]	0.93	0.91	0.92	0.96	0.97	0.96
Naive Bayes† [4]	0.61	0.91	0.73	0.96	0.80	0.87
J48 decision tree classification† [4]	0.91	0.86	0.88	0.95	0.97	0.96
FRJD (M1)	0.40	0.20	0.27	0.60	0.80	0.69
FRJD (M2)	0.94	0.90	0.92	0.90	0.73	0.80
FRJD (M3)	0.91	0.61	0.73	0.81	0.22	0.34
FRJD (M1+M2+M3)	**0.98**	0.98	0.98	**0.98**	0.98	**0.98**

Table 6.4: Performance of different models on the public dataset (EMSCAD) where M1, M2, and M3 are contextual, factual, and meta-features. Results of † are taken from [4].

Approaches	**Metrics**					
	Fraudulent			**Legitimate**		
	P	R	F1	P	R	F1
LR	0.71	0.64	0.67	0.94	0.31	0.47
SVM [209]	0.95	0.53	0.68	0.83	0.15	0.25
RF	0.84	0.75	0.79	0.96	0.48	0.64
FRJD (M1)	0.88	0.52	0.65	0.92	0.25	0.39
FRJD (M2)	0.84	0.82	0.83	0.65	0.49	0.55
FRJD (M3)	0.49	0.79	0.61	0.21	0.32	0.25
FRJD (M1+M2+M3)	0.98	0.99	**0.98**	0.97	0.96	**0.96**

Table 6.5: Performance of different models on a proprietary dataset where M1, M2, M3 are contextual, factual, and meta features.

6.9 Ablation Study

We use contextual features (M1), factual (M2), and metadata features (M3) separately as sub-models 'M1, 'M2', and 'M3'. In Table 6.4, we observe the substantial increase in precision from 0.40 with M1 (contextual) alone to 0.98 with M1+M2+M3 highlighting the significant

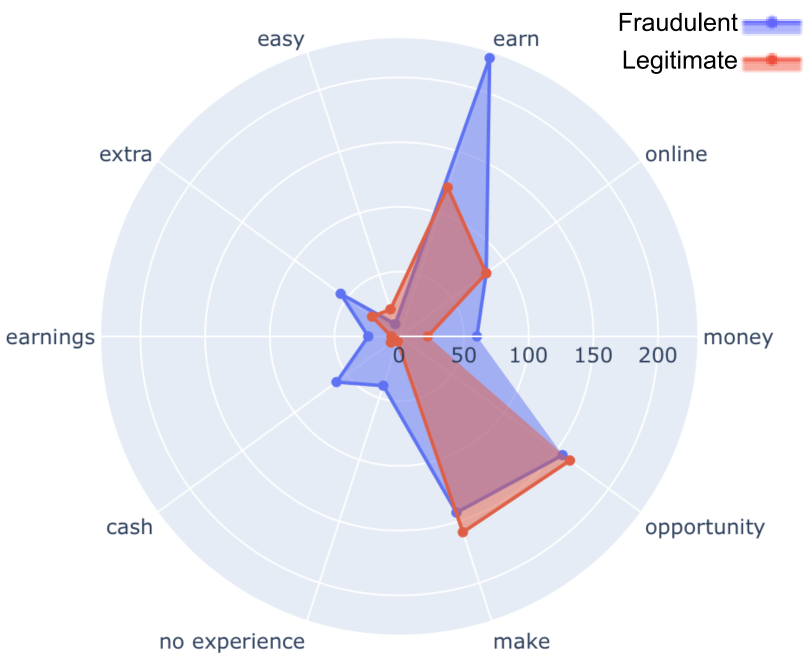


Figure 6.3: The radar plot provides a visual representation of the keywords present in both legitimate and fraudulent job postings. The plot shows that keywords such as ('earn', 'no experience', 'extra', 'cash', 'earnings', 'money') have more frequent occurrences in fraudulent jobs than legitimate ones. This also suggests that legitimate jobs focus more on generic keywords ('opportunity',' online').

contribution of factual (M2) and meta (M3) features on public dataset. The F1-score increased from 0.69 with M1 alone to 0.98 with all features for legitimate job postings in the public dataset. The ablation study reveals that component M1 captures context with a precision of 0.88 for fraudulent job postings (Table 6.5). Furthermore, the component M2 gives a Precision of 0.84 for fraudulent job postings but yields a Precision of 0.65 for legitimate job postings. The possible reason could be the similar facts in both the knowledge graphs. With F1-score of 0.61, the experiment on M3 component shows that the meta-features, such as the number of skills, qualifications, and job length, are rudimentary for proprietary dataset. We also verified the significant reasons for marking these job postings as fraudulent, including seeking money, using legitimate employer names, advertising paid training-based courses, sharing multiple accounts for promotion, etc. We identified some facts where the model fails to distinguish between true and false facts. These facts are demanding visa fees, common to both legitimate and fraudulent job postings for some job titles.

6.10 Summary

We proposed a multi-tier novel end-to-end framework called FRaudulent Jobs Detection (FRJD), which jointly considers a) fact validation module using knowledge graphs, b) contextual module using deep neural networks c) meta-data inclusion to capture the semantics of job postings. We conducted our study on a fact validation dataset containing 4 million facts extracted from job postings. We compared and performed an extensive evaluation of $157,880$ job postings. Finally, we provided various insights on fact-checking algorithms for our dataset. We believe that our framework is generalizable to other datasets in the recruitment domain.

Chapter 7

Identify low-quality content on online professional platforms

“Quality is never an accident; it is always the result of intelligent effort."

- John Ruskin

In the previous chapters, we explored the different issues related to content quality (inconsistent variations, missing entities and misleading information) using domain-specific learning and knowledge. In this[1] chapter, we aim to address another facet of low-quality content such as being off-topic, lacking clarity, and relying on opinions on online professional platforms.

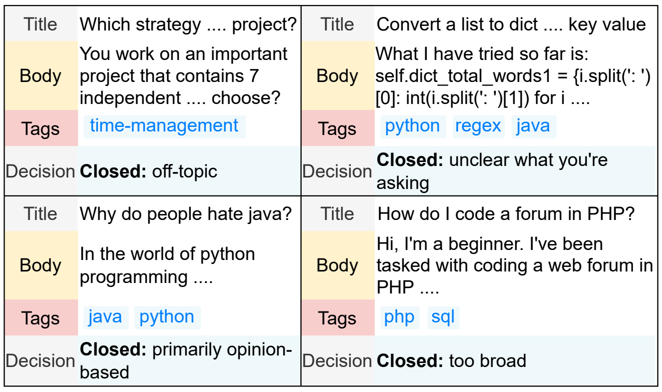
7.1 Introduction

Knowledge sharing on online professional platforms is essential for fostering a productive and collaborative environment. Stack Overflow (SO), a popular Community Question Answering (CQA) service as well as knowledge sharing platform with 22.1 million questions and 16.6 million users as of 2022 [210]. SO is an open-access website used by both novices and experts that relies heavily on a system of reputation points, and this reputation is built by not only providing answers but also by asking questions effectively. In Stack Overflow, as in many other knowledge-sharing communities, the quality of your questions is closely linked to your overall reputation and influence within the community. Conversely, asking low-quality (off-topic, irrelevant, unclear) or repetitive questions can lead to downvotes and negatively impact the rep-

[1]Work presented in this chapter has been accepted to IJCNN 2022. **Goyal, N.***, Arora, U.*, Goel, A., Sachdeva, N., Kumaraguru, P. Ask It Right! Identifying Low-Quality questions on Community Question Answering Services. In Proceedings of International Joint Conference on Neural Networks (IJCNN-2022), July 19 - July 23, Padua,Italy.

utation of the platform. Therefore, it is imperative to maintain the quality of questions posted over the platform [56]. To maintain the quality of questions, SO issues guidelines and employs a reward-based voting mechanism to incentivize users to ask good quality questions [211]. Questions not following the guidelines are ‘closed’ via a community-based voting system. ‘Closed’ questions can not receive answers, but the user can improve them for reopening. The primary reasons for ‘closing’ questions from a quality perspective are *‘off-topic’*, *‘unclear what you’re asking’*, *‘primarily opinion-based’*, and *‘too broad’*. There are approximately 5900+ new questions posted daily over the SO platform.[2] As a result of high traffic, it is inefficient and difficult to review every question manually. These questions also increase the workload of moderators and experienced users with distinguished privileges who spend time ‘closing’ questions. Therefore, it is imperative to identify and close low-quality questions automatically to minimize the workload on moderators.

Figure 7.1 shows ‘closed’ questions from SO. Existing works [56, 211–213] focus on pre-



Title	Which strategy .... project?	Title	Convert a list to dict .... key value
Body	You work on an important project that contains 7 independent .... choose?	Body	What I have tried so far is: self.dict_total_words1 = {i.split(': ')[0]: int(i.split(': ')[1]) for i ....
Tags	time-management	Tags	python regex java
Decision	**Closed:** off-topic	Decision	**Closed:** unclear what you're asking
Title	Why do people hate java?	Title	How do I code a forum in PHP?
Body	In the world of python programming	Body	Hi, I'm a beginner. I've been tasked with coding a web forum in PHP ....
Tags	java python	Tags	php sql
Decision	**Closed:** primarily opinion-based	Decision	**Closed:** too broad

Figure 7.1: A sample of ‘closed’ questions from Stack Overflow. These are ‘closed’ due to different reasons such as ‘*off-topic*’, ‘*unclear what you’re asking*’, ‘*too broad*’ and ‘*primarily opinion-based*’.

dicting the quality of questions on Community Question Answering websites. They either rely on textual styles like *title length*, *body length*, handcrafted features (*number of URLs*), and supervised learning (e.g., Random forest, SGBT) or Convolutional Neural Network [107], Hierarchical Attention Network [214], and Gated Recurrent Units [215, 216].

However, these approaches fall short in: a) effectively capturing the semantic and structural information of the content, b) detecting low-quality questions at the time of creation (*cold-start problem*) when features such as votes, question scores, answers, and comments are unavailable, c) estimating the survival probability of a question till it gets ‘closed.’

We utilize the semantic information in the question’s content using a transformer-based model, i.e., BERT [148] to improve the modelling of questions. We observe that including relevant tags

[2]https://stackexchange.com/sites?view=list#traffic

is one of the guidelines for asking a good-quality question. Therefore, we utilize tag-specific information, which is crucial in determining the question's visibility and answering the questions. Recent advances in graph-based deep learning [90] have led to the rise of graph neural networks (GNNs) that can model structural relationships well. We introduce a novel way to exploit the tag-related information to capture implicit relationships between a question and tag using Graph Convolutional Networks (GCNs) [44].

Towards this end, we propose a multi-tier hybrid framework, Low-Quality Question Detection (LQuaD), that incorporates the content-related information associated with each question using BERT. Using GCNs over a graph of 2.9M nodes and 8.4M edges, LQuaD can collaboratively identify patterns between questions and tags in a transductive manner. LQuaD addresses the cold-start problem using only pre-submission information of questions posted over the platform. Furthermore, we conduct survival analysis [45] to provide insights on closed questions by comparing mean times and median survival time(s) for the question's closure. We use Kaplan-Meier Estimator [217] for survival analysis to compare the temporal event of low-quality questions being 'closed'. The analysis compares the timeframe (mean time) for question closing among different categories and tags.

7.2 Related Work

This section provides an overview of related literature on the detection of low-quality questions, graph-based approaches, and survival analysis.

Low-quality Questions Detection: Previous works [54, 211] studied the handcrafted features of posts, community, and users for detection of closed and deleted questions over Stack Overflowplatform. Baltadzhieva et al. [56] studied and analyzed the linguistic terms contained in the post's content to predict the question's quality. However, the work defines the question's quality in abstract terms, making it prone to subjectivity and may result in somewhat arbitrary assessments. Ahmed et al. [102] studied user behaviour in guiding the feature engineering process to determine the question's quality. Jan et al. [108] focused on identifying unclear questions as they have low quality. Research Work [107] contributes with a multi-class classification of 'closed' questions along with 'closing' reasons for questions. Deepak et al. [106] estimate the relative difficulty of questions. Most of these methods [56, 211] predict question quality by utilizing handcrafted features like community-based features, part-of-speech tagging, user profile features, textual features (*body length*, *no. of URLs, count of words*) and writing styles. These features capture limited information related to the context and semantics of the question's content, which is a crucial step to exploit the maximum efficiency of the natural language processing [218]. To further improve the identification of low-quality questions, LQuaD combines contextual and semantic properties of questions into our proposed framework.

Graph Neural Networks: Recent advancements in GNNs [90, 140, 219] (discussed in Chapter 4) allow using node neighborhoods to learn discriminative features collaboratively. Literature [219] utilizes a joint document-words graph for text classification via node classification task. However, GNNs are unexplored in capturing the information using a heterogeneous question-tag graph for the identification of low-quality questions. GNNs are well-equipped to capture information flow between the questions and their associated tags for node classification tasks.

Survival Analysis: Survival analysis provides statistical methods to study the time-to-event data [220]. It assists in determining the longevity of responses on social media platforms [221] and in assessing the wait time of the callers [222]. The technique incorporates valuable information about unresolved questions in the prior studies [223]. A previous work [224] uses survival analysis to estimate the time till the first response and time till the accepted answer to questions on Stack Overflow. We extend the study and carry out the temporal analysis using the Kaplan-Meier Estimator [217] to provide an approximate time of question getting 'closed'. This work is uniquely inclined towards using graph convolutional networks and transformers in the cold-start scenario for low-quality question detection. We also provided various insights and compared the timeframe to 'close' a low-quality question across different tags and 'closing' reasons.

7.3 Contributions

The major contributions of this work are:

- We propose a novel framework, LQuaD, which establishes the utility of a question-tag graph and transformers to detect low-quality questions that are likely to get 'closed' at the time of posting. Our framework acts as an early-assessment tool to assist users in composing a question, which would remain open and receive responses.

- We also examine the impact of non-content related characteristics of the question using survival analysis to compare the time duration of closure of the question among different categories of reasons and tags.

- We evaluate LQuaD on the dataset of 'closed' and non-'closed' questions from Stack Overflow platform and make the resources publicly available for reproducibility.[3]

[3] https://precog.iiit.ac.in/resources.html

7.4 Preliminary & Problem Definition

Objective: Consider the set of questions $\mathcal{Q} = \{q_1, q_2, \cdots\cdots, q_i\}$, a question $q_i \in \mathcal{Q}$ is a tuple $(c_i, \mathcal{T}_i, y_i)$ where c_i, $\mathcal{T}_i$, y_i represents the content, tag set, and label of the i^{th} question, respectively.
The content is represented by the concatenation of *title* and *body* of the question.
The tag set is represented as, $\mathcal{T}_i = \{t^i_1, t^i_2, t^i_3, ..., t^i_k\}$ $\forall\, 1 \leq k \leq 6$ and t^i_k represents the k^{th} tag of the i^{th} question.
Also, $y_i \in \{0, 1\}$, where $y_i = 0$ corresponds to the 'closed' class i.e., the question is unfit for the SO platform, and $y_i = 1$ corresponds to the non-'closed' class.
Graph construction: We construct an undirected question-tag graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ and $\mathbb{E}$ are the set of nodes and edges respectively. Here $\mathbb{V}$ consists of questions belonging to $\mathcal{Q}$ and tagset $\mathbb{T} = \{\mathcal{T}_1 \cup \mathcal{T}_2 \cup ... \cup \mathcal{T}_{|\mathcal{Q}|}\}$. We construct an edge $e \in \mathbb{E}$ between q_i and t^i_k where $\mathbb{E} \subset \mathcal{Q} \times \mathbb{T}$. Additionally, for each question q_i, we represent its content c_i and each tag t^i_k with d-dimensional vectors. Let $\mathcal{C} \in \mathbb{R}^{n \times d}$ be the feature matrix with each row containing vector representation of question's content c_i or tag t^i_k where $n = |\mathbb{V}|$.

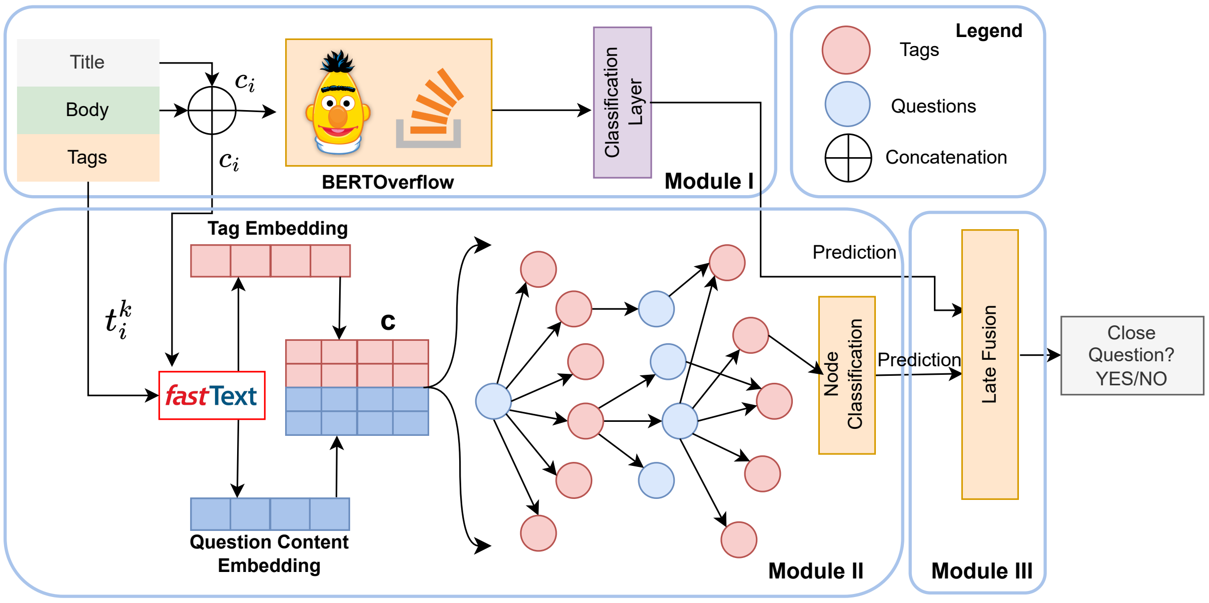


Figure 7.2: **LQuaD** consists of three components: Module I fine-tunes the BERTOverflow model, which inputs the title and body for the question's classification task, Module II consists of a graph convolutional network initialized with the question's content and tag embedding for the node classification task, Module III consists of a late fusion strategy that combines predictions from both modules.

7.5 Methodology

In this section, we discuss the problem definition and our proposed framework.

7.5.1 Low Quality Question Detection (LQuaD)

Module I: In this module, our goal is to capture semantic features from content c_i of the question. First, we pre-process the question's content to remove code snippets, HTML tags, non-ASCII characters, and hypertext references. Then, we fine-tune BERTOverflow [225] model, trained on 152 million sentences from the Stack Overflow's ten-year archive, for our classification task. The BERT model consists of 12 layers, with each layer consisting of an attention mechanism [218] to capture the hidden styles, context, and complexities of low-quality questions effectively. We obtain m-dimensional vector g_i after applying the dropout layer to the pooled output from BERT. Furthermore, the final output is returned by classification layer as shown in Equation 7.1.

$$p_i^{bert} = \sigma\left(W_g g_i + b_g\right) \tag{7.1}$$

where σ is softmax; p_i^{bert} are the final predictions; weight matrix W_g and bias b_g are trainable parameters.

Module II: This module captures the structural relationships between the questions and their corresponding tags. Specifically, we construct a heterogeneous graph consisting of question and tag nodes. We initialize the feature matrix $\mathcal{C}$ by fine-tuning the fastText skip-gram model [131] on the SO data in an unsupervised manner as shown in Figure 7.2. Using the fastText model, we obtain d-dimensional vector for every question and each tag, respectively. Each node $v \in \mathbb{V}$ is associated with a row of matrix $\mathcal{C}$ that gets updated during training. Let $\mathbb{A}$ be the adjacency matrix which represents the connection between the nodes of $\mathbb{G}$. We add self-connections with the equation $\widetilde{\mathbb{A}} = \mathbb{A} + \beta I$ where β is a trainable trade-off parameter. The degree matrix is given as $\mathbb{D}_{ii} = \Sigma_{j=0}^{n} \widetilde{\mathbb{A}}_{ij}$. The normalized adjacency matrix is given by $\hat{\mathbb{A}} = \mathbb{D}^{-\frac{1}{2}} \widetilde{A} \mathbb{D}^{-\frac{1}{2}}$. Equation 7.2 shows the GCN layer that updates the node representation using a weighted sum of neighboring node features.

$$h_i^{(l+1)} = ReLU\left(\left(\frac{1}{d_i} h_i^{(l)} + \sum_{j \in \mathcal{N}_i, j \neq i} \frac{1}{\sqrt{d_i d_j}} h_j^{(l)}\right) W^{(l)}\right) \tag{7.2}$$

where $\mathcal{N}_i$ be the set of neighboring nodes of a i^{th} node; $d_i = |\mathcal{N}_i|$, $h_i^{(l)}$ be the representation of the i^{th} node after layer l, and $W^{(l)}$ be the weight matrix of layer l. Equation 7.3 represents the output feature matrix after l layers of GCN.

$$H^{(l+1)} = ReLU\left(\hat{\mathbb{A}}H^{(l)}W^{(l)}\right) \tag{7.3}$$

where $H^0 = \mathcal{C}$ i.e. the input feature matrix when $l = 0$ depicts initialization of network. We utilize ReLU as non-linearity and use dropout layers between successive convolutions. The first message-passing layer facilitates the information flow between questions and their respective tags. Here, the questions capture the aggregate information from their neighborhood nodes (tags). The second layer captures information between questions connected via common tags. Module II employs two layers to execute transductive learning via GCN and employs a classification layer for predictions as shown in Equation 7.4.

$$p_i^{gcn} = \sigma(W_h h_i^{(l+1)} + b_h) \tag{7.4}$$

where σ is softmax; p_i^{gcn} are the final predictions; weight matrix W_h and bias b_h are trainable parameters.

Module III: In this module, we combine the predictions (p_i^{bert} and p_i^{gcn}) from Module I and Module II using late fusion techniques [69] in two different settings, i.e., maximum and weighted mean. Equation 7.5 and Equation 7.6 fuses the predictions from both modules for maximum and weighted mean.

$$\begin{aligned} p_i^{pred} = argmax(&max(p_i^{bertc}, p_i^{gcnc}), \\ &max(p_i^{bertnc}, p_i^{gcnnc})) \end{aligned} \tag{7.5}$$

$$\begin{aligned} p_i^{pred} = argmax(&\lambda * p_i^{bertc} + (1-\lambda) * p_i^{gcnc}, \\ &\lambda * p_i^{bertnc} + (1-\lambda) * p_i^{gcnnc}) \end{aligned} \tag{7.6}$$

where λ is a trainable parameter. From Module I, p_i^{bertc} and p_i^{bertnc} are the obtained probabilities for 'closed' and non-'closed' class respectively. Similarly, from Module II, p_i^{gcnc} and p_i^{gcnnc} are the obtained probabilities for 'closed' and non-'closed' respectively.

	Module I			**Module II**	
Dataset	**No. of train samples**	**No. of val. (test) samples**	**Total**	**Graph**	**Total**
‘Closed’	155,554	51,852	259,258	Nodes (questions)	2,851,838
Non-‘Closed’	1,555,548	518,516	2,592,580	Nodes (unique tags)	48,374
Total questions	1,711,102	570,368	2,851,838	Edges	8,442,584

Table 7.1: Dataset statistics from the Stack Overflow platform.

7.6 Datasets

We describe our dataset creation methodology for LQuaD (Module I and II) and temporal event analysis.

LQuaD (Module I) dataset: We create a dataset of ‘closed’ and non-‘closed’ questions using a data dump of 17.7 million questions from SO platform. First, we filter 847,721 ‘closed’ questions from the data dump of SO posts from August 2008 until June 2019. Table 7.1 reports our dataset statistics from SO platform. When SO went live, questions were closed due to seven different reasons. However, SO changed the reasons to close the questions after June 2013. The new reasons to close the question are *‘duplicate’*, *‘off-topic’*, *‘unclear what you’re asking’*, *‘too broad’*, and *‘primarily opinion-based’*. We filter out *‘duplicate’* questions while making the ‘closed’ dataset because these are not necessarily low-quality questions. SO closes duplicate questions to eliminate redundancy over the platform. We consider the pre-submission information of questions present when the question gets ‘closed.’ It ensures the retention of the original content before the owner makes any edits to improve its quality. We consider only those questions that are ‘closed’ only once since their creation and not reopened after that. The number of questions in the ‘closed’ class is significantly less than that of the non-‘closed’ class. Therefore, we randomly sample non-‘closed’ questions from the data dump such that its size is ten times the size of the ‘closed’ dataset, leading to an imbalanced dataset. We utilize weighted loss [226] to handle the class imbalance problem for our dataset.

LQuaD (Module II) dataset: For Module II, we extract the tag-related information corresponding to the questions dataset created in the previous module. A question can have a maximum of five tags as per SO guidelines.[4] However, it turns out that a question has a maximum of six tags in the data dump.[5] Therefore, we find 344,491 questions with only 1-tag,

[4]https://stackoverflow.com/help/tagging

[5]https://stackoverflow.com/questions/14071930/creating-a-playlist-out-of-songs-from-selected-artists

740,264 questions with 2-tags, 825,405 with 3-tags, 567,209 questions with 4-tags, 374,295 questions with 5-tags, and 173 questions with 6-tags, respectively. Finally, we construct a bipartite graph $\mathbb{G}$ by connecting questions to their corresponding tags. Table 7.1 reports the number of nodes, edges, and tags for Module II.

Temporal event analysis dataset: We use the 'closed' questions dataset (Module I) and define our EOI for survival analysis as the 'time until a question gets closed.' We collect the temporal data of the question's creation and the 'closing' date for these questions. We filter out the questions from the closed dataset with unknown timestamps and 'closing' reasons. To compute the time elapsed for each question, we find the difference between the timestamp of a question's 'closed' date and its creation date. We extract the reasons for closing these questions and found 118,048 *off-topic*, 49,391 *unclear what you're asking*, 52,023 *too broad*, and 22,132 *primarily opinion-based* questions. We also categorize[6] the corresponding tags into 17,774 *database*, 1,438 *cloud*, 20,912 *web frameworks*, 132,875 *programming languages*, and 6,684 *other frameworks*.

[6] https://insights.stackoverflow.com/survey/2021

	Reasons				**Tag Categories**				
Category	**Off-topic**	**Unclear what you're asking**	**Too broad**	**Primarily opinion-based**	**Database**	**Cloud**	**Web frameworks**	**Programming languages**	**Other frameworks**
Mean Time Till EOI (days)	330.76	65.91	230.87	**531.81**	145.54	228.13	185.03	177.89	**482.89**
Median Survival Time (hrs)	12.89	7.51	8.57	**15.07**	9.04	**27.33**	8.17	7.27	16.51

Table 7.2: We report the mean time (days) till EOI and median survival time (hrs) corresponding to reasons and tag categories. The highest values in respective rows are shown in bold and the lowest values are underlined.

7.7 Temporal Event Analysis

We perform survival analysis, a statistical tool to examine the duration until a particular Event Of Interest (EOI) occurs as a function of time [221, 222, 227]. For our study, we define EOI as the time elapsed from the question's creation to when it gets 'closed'. Unlike other statistical methods, survival analysis can handle events that may occur after the study's observation period. This method is well-suited for handling situations where the EOI can occur at different times, thereby including cases where questions remain open (not closed) during the observation or data collection phase. The dataset details for survival analysis (Temporal event analysis) in Section 7.6. We explore these temporal aspects of the questions before they get 'closed' using non-parametric Kaplan-Meier estimator [217] as shown in Equation 7.7.
We establish the following definitions:

- **Censoring** occurs when some observations do not experience the EOI (i.e., question closure) by the end of the study period.
- **Survival Probability** $S(t)$, which gives the probability that a question survives (i.e., remains open) beyond time t. We reported the mean and median survival times (Table 7.2).
- **Kaplan Meier Estimator**, $\widehat{S}(t)$ is used to estimate the survival function, $S(t)$. using observed data especially when the data include censoring (where not all observations have experienced the event by the end of the study).

$$\widehat{S}(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}) \tag{7.7}$$

where t_i is the time at which the question is 'closed', d_i denotes no. of posts 'closed' at time t_i, and n_i denotes no. of posts which haven't been 'closed' up to time t_i but will get 'closed' at some point after t_i.
Therefore, $\widehat{S}(t)$ is calculated by multiplying survival probabilities for each event time $t_i \leq t$. Each term $(1 - \frac{d_i}{n_i})$ adjusts for the fraction of questions closed at t_i relative to those still open at t_i.
Table 7.2 reports mean time to close the question, and median statistics of the survival function for 'closing reasons' and tag category. We observe that *'primarily opinion-based'* has the highest mean time till EOI and median $\widehat{S}(t)$ while *'unclear what you're asking'* has the least mean till EOI and median $\widehat{S}(t)$. This suggests that *primarily opinion-based* questions remain open for longer than *'unclear what you're asking'* questions. We also observe that *cloud* and *other frameworks* have higher mean time till EOI and median $\widehat{S}(t)$ than other categories. This

suggests that questions with tags in these categories are not 'closed' for relatively longer periods. Our dataset is represented more accurately by median than mean using the Kaplan-Meier estimator $\widehat{S}(t)$ due to its non-parametric nature. Figure 7.3 (a) shows *'primarily opinion-based'*

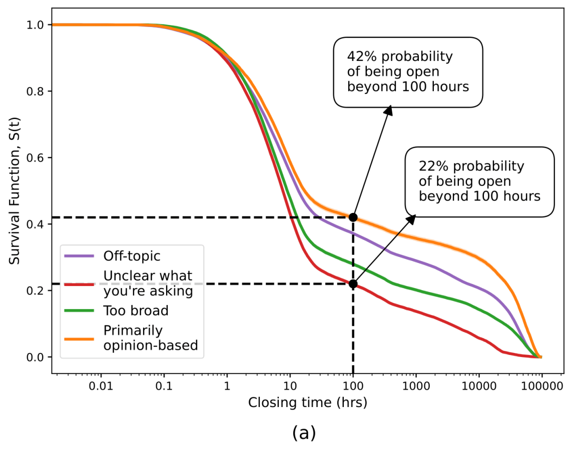


Figure 7.3: Kaplan-Meier estimator $\widehat{S}(t)$ of the survival function for the time period of 'closed' questions for different categories of reasons and tags. Plot (a) specifies the reasons due to which the question gets 'closed' whereas Plot (b) specifies the tag category.

questions have a higher probability of survival than *'unclear what you are asking'* questions for most of the time period after a question's creation. These trends show that the SO community promotes and encourages users to discuss and share opinions on the highly subjective, '*primarily opinion-based*' questions by not 'closing' them during their early stages. On the other hand, the community doesn't support unclear questions by 'closing' them relatively early which inhibits users from spending time answering these ambiguous questions.
Figure 7.3 (b) shows that the questions with tags related to *cloud* and *other frameworks* consistently have higher survivability (i.e., less chance of getting 'closed' in the future) than a question with tags related to *database*, *web frameworks*, and *programming languages.*

7.8 Experiments

This section provides baselines and model configurations along with hyperparameters.

7.8.1 Baselines

We compare LQuaD with the following state-of-the-art baseline models. Baselines (1-2) are state-of-the-art approaches for 'closed' question detection task. Furthermore, we created our baselines using standard techniques from literature (3-7).

- **Denzil et al.** [54] carry out the characterization study and detection of ‘closed’ questions using Stochasitc Gradient Boosting Trees (SGBT) classifier to predict whether the question will get ‘closed’ or not.
- **Toth et al.** [216] introduce a gated-recurrent-unit-based (GRU) classifier that uses textual features of the post to accomplish binary classification task.
- **CountVec + LR** [228] We utilize count vectorizer as a feature extractor module for post’s textual data and pass it through the logistic regression model.
- **CountVec + XGBoost** [229] utilize count vectorizer to extract features from textual data and pass it through the XGBoost classifier model.
- **FastText + LR** [131] extract the pre-trained embeddings using fastText and pass it through the Logistic regression model.
- **Distilled-BERT + LR** [156] utilize the pre-trained embeddings using DistilBERT and pass it through the Logistic Regression classifier model.
- **GCN (fastText)**[230] initialize the node embeddings using pre-trained embeddings from fastText [131] model and pass it through the Module II of LQuaD.

7.8.2 Model configurations

Table 7.3 reports the optimal values after tuning hyperparameters of Module I and II of LQuaD. In Module I, we use the coarse grid-search algorithm to search the hyperparameter space. The BERTOverflow model converges before one epoch during the fine-tuning task. In Module II, we initialize nodes with d-dimensional vector representation using fine-tuned fastText model where d=300 and run it for 500 epochs. We find that the trainable trade-off parameter, β, has optimal value of two. We use the train-validation-test (60:20:20) split and perform 10-fold calculations for classification run of Module I and II. For Module III, the optimal value of λ for weighted mean technique is obtained by tuning its value in the uniform distribution $\mathcal{U}\{0,1\}$. Further, we apply 5-Fold cross-validation during late fusion to obtain $\lambda = 0.55$. We use cross-entropy loss and the Adam optimizer [231] to train LQuaD.

Module I		Module II	
Hyperparameter	**Value**	**Hyperparameter**	**Value**
Batch size	8	No. of layers	2
Learning rate	1e−5	Learning rate	1e−2
Weight Decay	0.01	No. of output channels after each GCN layer	64
Dropout	0.1	Dropout	0.5

Table 7.3: Hyperparameter settings for Module I and Module II of LQuaD.

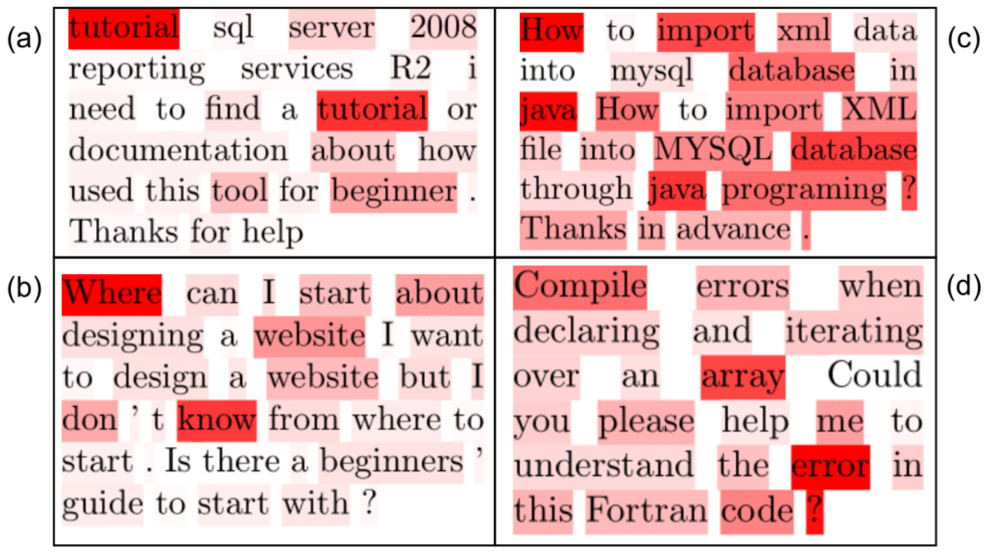
Model	Precision	Recall	F1
Denzil et al. [54]	70.25	70.25	70.24
Toth et al. [216]	73.78	73.33	73.66
Count Vectorizer + LR [228]	89.94	76.90	81.38
Count Vectorizer + XGBoost [229]	90.17	77.48	81.82
FastText + LR [131]	90.52	79.71	83.44
DistilBERT + LR [156]	92.19	85.26	87.59
GCN (fastText) [230]	90.69	83.98	86.42
LQuaD (LF [mean])	**94.85**	**95.20**	**94.86**

Table 7.4: Average Performance metrics (weighted) on the Stack Overflow dataset.

7.9 Results and Analysis

We compare LQuaD with different state-of-the-art baselines (see Table 7.4). Denzil et al. [54] also predicted closed questions based on various predictive features using post-submission information with F1-score of 70.24%. Toth et al. [216] similar to our work, predicted question closing based on pre-submission information using a gated recurrent unit and obtained an F1-score of 73.66%. LQuaD (LF[max]) and LQuaD (LF[mean]) outperforms by 19% and 21% as compared to the best baseline [216] for the binary classification task. Table 7.4 shows the average over the results obtained from 10-fold calculations. We perform the student's t-test [232] to compare the results obtained from different modules. The t-test results of Module I and II follow the distribution and differ significantly from each other at the applied significance level $p < 0.01$.

Analysis: We analyze the attention layers of fine-tuned model in Module I. Figure 7.4 (a) shows that the model gives higher attention to words like *'tutorial'* and *'beginner'*. The model is able to capture the newbie behavior over the platform, and the question is correctly predicted as 'closed'. Similarly, Figure 7.4 (d) shows that the model give higher attention to words like {'error', 'array', 'code', 'compile'}. Therefore, the model can capture semantic and contextual information about the *'debugging a coding error'* which represents the relevant question posted


(a) tutorial sql server 2008 reporting services R2 i need to find a tutorial or documentation about how used this tool for beginner . Thanks for help

(b) Where can I start about designing a website I want to design a website but I don ' t know from where to start . Is there a beginners ' guide to start with ?

(c) How to import xml data into mysql database in java How to import XML file into MYSQL database through java programing ? Thanks in advance .

(d) Compile errors when declaring and iterating over an array Could you please help me to understand the error in this Fortran code ?

Figure 7.4: Some examples of predictions using Module I. For each token in the input, we show the visualization of self-attention averaged over all 12- attention heads of fine-tuned BERT Overflow. Higher attention weights corresponds to darker color (red). Plot (a) is an example of 'closed' predicted correctly, Plot (b) is an example of non-'closed' predicted as 'closed', Plot (c) is an example of 'closed' predicted as non-'closed' and Plot (d) is an example of non-'closed' predicted correctly.

over the platform. LQuaD is able to capture tag information such as '*agile*' and *'open-source'* as majority of questions containing these tags are 'closed'. The questions containing these tags are correctly predicted as 'closed'. The SO website also clarifies that questions using *'open-source'* and *'agile'* tags would be considered *off-topic*.[7] Therefore, LQuaD is able to model the structural relationships between the questions and tags effectively. Figure 7.5 depicts the node representations before and after training of Module II using PCA [233]. We observe that Module II learns tags patterns associated with questions in a transductive manner.

7.10 Ablation Study

Table 7.5 shows the ablation study of Module I and II separately. Module I achieves Weighted performance metrics, i.e., Precision, Recall, and F1-score of 94.81, 95.08, and 94.53, respectively. Similarly, Module II, i.e., GCN model initialized with fine-tuned fastText embeddings achieves Precision, Recall and F1-score of 91.86, 84.38, and 86.92 respectively. We also observe lower-precision and recall values by initializing the question-tag graph nodes with pre-trained language models such as BERT [156]. Additionally, fastText embeddings are

[7] https://stackoverflow.com/questions/tagged

light-weight embeddings which makes Module II less computationally expensive as compared BERT-based embeddings. These results show that GCNs achieved comparable precision and relatively lower recall than transformer-based models.

Model	**Precision**	**Recall**	**F1**
Module I	94.81 (0.03)	95.08 (0.04)	94.53 (0.01)
Module II	91.86 (0.07)	84.38 (0.07)	86.92 (0.08)
LQuaD (LF [max])	93.62 (0.02)	92.85 (0.04)	93.16 (0.07)
LQuaD (LF [mean])	**94.85 (0.03)**	**95.20 (0.05)**	**94.86 (0.02)**

Table 7.5: Effectiveness of LQuaD (and variances) as compared to Module I and Module II.

7.11 Error Analysis

We demonstrate some of the errors encountered by LQuaD. For example, words like *'java programming'* and *'MYSQL database'* are weighted more by the model as 'java' has usually more instances in non-'closed' dataset. However, due to these words, the model incorrectly predicted the 'closed' as non-'closed'. Additionally, *'Where can ....with'* shows that the

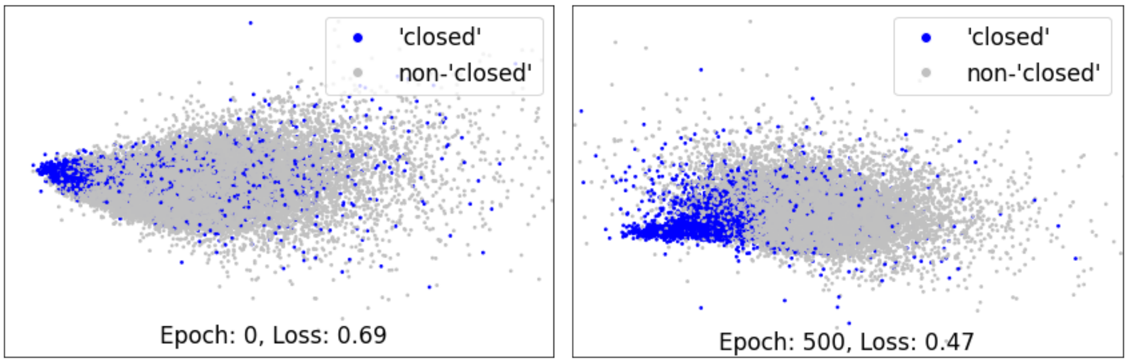


Figure 7.5: PCA visualizations of node representations for 'closed' and 'non'-closed questions before (left) and after (right) training.

actual class is non-'closed' whereas the model predicted it 'closed' which is an example of false positive. Our transformer-based model is able to look at some words like 'array' and 'error' to make the decision for non-'closed' class. *'Compile ....code'* shows that the actual class is non-'closed' and LQuaD predicted it correctly as well.

7.12 Summary

We propose LQuaD that incorporates semantic information of questions associated with each post using transformers and learns question and tag graphs in a transductive manner using GCNs. Our graph consists of 2.9M nodes and 8.4M edges. LQuaD detects low-quality questions that are likely to get ‘closed’ at the time of posting. Our framework acts as an early-assessment tool to assist users in composing a question, which would remain open and receive responses. Further, we use survival analysis that reduces the number of questions closed by informing users to take appropriate action. We compare that the timeframe between the stages from the question’s creation till it gets ‘closed’ varies significantly for tags and different ‘closing’ reasons for these questions. LQuaD outperforms the state-of-the-art methods by a 21% in F1-score on the dataset of 2.8 million questions.

Chapter 8

Conclusion

"A conclusion is the place where you get tired of thinking."
-Martin Henry Fischer

This thesis explored novel frameworks for handling and improving content quality issues on online professional platforms using domain-specific learning and knowledge. We curated datasets from online recruitment and knowledge-sharing platforms as discussed in Chapter 1. We introduced novel techniques that improve downstream applications such as skill prediction, job recommendations, community question-answering systems, job search, entity cards and evaluated these techniques for enhancing professional content quality using domain-specific learning and knowledge. Our proposed studies, analysis, and solutions have been published at top-tier conferences in natural language processing and knowledge discovery. Section 8.1 presents a summary of our contributions.

8.1 Summary

The main contributions of this thesis are:

8.1.1 Standardization of online professional content

We built an a novel multi-tier framework Kernel-based Canonicalization Network (KCNet) that outperforms all the known statistical and deep learning methods. We focused on canonicalizing real-world entities from the recruitment domain such as companies, designations, institutes, and skills. KCNet induces a non-linear mapping between the contextual vector representations while capturing fine-granular and high-dimensional relationships among vectors. KCNet is able to handle online professional entities from heterogeneous sources and leverage side information from external sources. KCNet efficiently models semantic and meta side information such as industry type, url of websites, etc. from external knowledge towards exploring kernel features for canonicalizing entities in the recruitment domain. Furthermore, we applied Hierarchical Agglomerative Clustering (HAC) using the pairwise similarity matrix $\mathcal{M}_{sim}$ to create unique clusters of entities. Experiments revealed that the Kernel-based neural network approach achieves significantly higher performance on both proprietary and open datasets. We demonstrate that our proposed methods are also generalizable to domain-specific entities in similar scenarios.

8.1.2 Deal with missing entities in online professional content

We proposed a novel JobXMLC framework, which uses a graph neural network to incorporate neighborhood information with the help of a collaborative graph over jobs and skills. JobXMLC leverages skill attention mechanism for more effective extreme classifiers and attends to multi-resolution representations of jobs and skills. JobXMLC outperforms leading deep extreme classifiers on precision and recall metrics by 6% and 3%. JobXMLC also operates in warm and cold-start scenarios effectively. JobXMLC is 18X faster on training and 634X faster on predicting than deep extreme classifiers and can be scaled efficiently to real-world datasets with thousands of labels. We believe that JobXMLC can be deployed on large-scale recruitment platforms for predicting missing skills using job descriptions.

8.1.3 Identify misleading information in online professional content

We proposed a multi-tier novel end-to-end framework called FRaudulent Jobs Detection (FRJD), which jointly considers a) fact validation module using knowledge graphs, b) contextual module using deep neural networks c) meta-data inclusion to capture the semantics of job postings. We conducted our study on a fact validation dataset containing 4 million facts extracted from job postings. We compared and performed an extensive evaluation of 157,880 job postings. Finally, we provided various insights on fact-checking algorithms for our dataset.

8.1.4 Identify low-quality (off-topic, unclear, opinion- based, too broad) content

We built LQuaD that incorporates semantic information of questions associated with each post using transformers and learns question and tag graphs in a transductive manner using GCNs.

Our graph consists of 2.9M nodes and 8.4M edges. LQuaD detects low-quality questions that are likely to get ‘closed’ at the time of posting. Our framework acts as an early-assessment tool to assist users in composing a question, which would remain open and receive responses. Further, we use survival analysis that reduces the number of questions closed by informing users to take appropriate action. We also provided various insights and compared the timeframe to ‘close’ a low-quality question across different tags and ’closing’ reasons. LQuaD outperforms the state-of-the-art methods by a 21% in F1-score on the dataset of 2.8 million questions.

Chapter 9

Future Work

“The best way to predict the future is to create it"

-Abraham Lincoln and Peter Drucker

In this chapter, we discuss the limitations and future directions in Section 9.1. For future work, we also added our initial Analysis on Understanding Employment Scams in Section 9.2.

9.1 Limitations and Future directions

- Our Kernel-based Canonicalization architecture relies on external sources of side information for emerging entities. In our future work, we plan to use more contextual information (e.g., information present on the employer’s website). We intend to develop more features to handle the quality of entities. Another branch of potential work is improving the employer KB’s quality and coverage. The dynamically evolving nature of entities warrants new solutions enabling entity canonicalization in the recruitment domain.
- We perform experiments on jobs sampled from a popular Singaporean government job portal and StackOverflow, which is limited to English. Our approach can handle missing skills, which are part of our skill vocabulary, but it cannot infer new emerging skills from job descriptions, i.e., out-of-vocabulary. We will consider domain knowledge and the popularity of job skills to generalize our approach for job-candidate mapping applications for future work. We wish to expand our work to other recruitment domain applications with resumes and candidate profiles.
- Our work on community question answering dataset is limited to low-quality (closed) questions. Generally, the community takes significant time to detect potentially deleted questions. Deleted questions are generally significantly lower in quality than ‘closed’ questions. We wish to explore this dimension of deleted questions. In future, we want

to explore advanced graph transformer-based architectures for modelling question-tag pairs. We also plan to study the ‘closed’ questions that re-open after edits.

- We wish to study the time complexity of the Fraudulent Jobs Detection (FRJD) framework and compare it with other approaches. We limit our study to translation-based fact-checking algorithms [85–87, 189]. In future, we plan to apply and test our approach for hierarchy-based, neural network-based, and path- based [234–236] fact-checking algorithms. We wish to compare different algorithms for learning heterogeneous documents, such as CVs and candidate profiles, to build an integrated framework and explore user features in future studies. We plan to build lightweight models for misleading content detection since deep learning approaches incur high computational costs when deployed as real-world systems. Another aspect is that with the rise of technology, fraudulent activities have seen a significant surge. This includes fraudulent job listings generated by artificial intelligence. In future, we need to develop robust algorithms and frameworks capable of detecting and mitigating AI-generated fraudulent job postings with greater accuracy and efficiency.

- Many job seekers share their experiences and concerns on various online complaint platforms, such as BBB[1]. Therefore, the viewpoints shared by job seekers in these complaints can help create strategies for vulnerable job seekers to detect and avoid scams.

We have initiated preliminary work on understanding employment scam complaints to help platforms continuously provide insights to their advisories based on the user complaint base. This will help them provide feedback to victims to ensure they stay updated with the dynamically evolving tactics used by scammers. To accomplish this, we have gathered our complaint dataset of employment scams from popular platforms and converted it into structured form to construct the Employment Scam knowledge graph using the Con2KG pipeline discussed in Chapter 5.

9.2 Initial Analysis on Understanding Employment Scams

Employment scams are among the top five scams registered over consumer complaint platforms[2]. These scams typically include jobs from fake recruiters, fake check scams[3] asking for

[1]https://bbbfoundation.images.worldnow.com/library/d8707e47-c886-48ec-b143-7b3db2806658.pdf

[2]https://www.bbbmarketplacetrust.org/story/39089075/bbb-scam-tracker

[3]https://bbbfoundation.images.worldnow.com/library/d8707e47-c886-48ec-b143-7b3db2806658.pdf

money or offering attractive non-existing jobs. Figure 9.1 shows a sample Employment scam complaint from a popular platform. According to the report on employment scams, 14 million people are exposed to employment scams with more than $2 billion lost per year. These reports

"I applied for an admin job on Indeed. I received an interview over the phone and did not think anything of it because of the Coronavirus. I was informed that I had the job and it was a work from home. I was issued a check in the amount of $4,800 off of a M&T business account. I finally got an information that the check was fraudulent."

Figure 9.1: A sample Employment scam complaint from BBB platform. Entities (Indeed, work from home, $4800, Coronavirus) present in text are shown in color.

focus on understanding the overall impact of these employment scams, providing prevention tips and structural interventions to help people avoid losing money. Researchers [237, 238] have previously analyzed, normalized, and detected online job scams.
However, there are a few limitations to these approaches, a) most of these qualitative studies are survey-based and are costly, b) these methods also ignore the heterogeneous information about money, location, employment type, organization, email, and phone number in the unstructured text that can help prevent consumers from losing money, c) understanding the victim's perspective (reported scams) and providing them objective information about scams is still unexplored in the domain. To address these issues, knowledge graphs [239] are considered suitable to capture heterogeneous information in entities and their relationships. A multi-relational knowledge graph stores the information in triples, consisting of entities as nodes and relations as different types of edges. Many research teams have organized the knowledge in their domain into a structured format, e.g., YAGO [240], NELL [241], and DBpedia [242].
Towards this end, we collect a dataset of six years consisting of 17K scam complaints from the BBB website.[4] Further, we investigate and extract the entities (money, location, date, email, phone number, employment type, and organization) from employment scam complaints. Then, we construct an Employment Scam Knowledge Graph consisting of 0.1M entities and 0.2M relationships to help workers protect themselves before falling for scams. Our preliminary work leverages graph representations to identify and avoid potential scams using this Employment scam knowledge graph. The Work [5] presented in this section has been accepted to CODS-COMAD 2023.
Dataset: The dataset source for this research is BBB Scam Tracker, an online platform where

[4] https://www.bbbmarketplacetrust.org/story/39089075/bbb-scam-tracker
[5] Goyal, N., Mamidi, R., Sachdeva,N., and Kumaraguru, P. Warning: It's a scam!! Towards understanding the Employment Scams using Knowledge Graphs. Accepted at ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD 2023) YRS track. Bombay, Jan 4 - 7, 2023

consumers and businesses report scams. BBB scam tracker website was accessible at the time of data collection. We collected around 17K employment scam complaints from USA and Canada. These complaints are from August 2015 to April 2021. We consider the employment scam complaints, which have a non-empty description field. The dataset consists of the scam's reported date, textual descriptions, the dollar value of any loss, zip code, and business name (imposter).

Methodology: The approach consists of three components: a) Understanding the complaints, b) Entity Extraction, and c) Knowledge integration. We extract the mode of approaching the victim from textual descriptions using a set of keywords such as Email, mailbox, inbox, spammed by inbox, text message, Facebook, phone call, call, called me up, social media, Instagram, Twitter, LinkedIn, personal message, WhatsApp, message, telegram, phone message, and contact number. We categorize them into six significant ways, i.e., *'email'*, *'call'*, *'online social networks'*, *'online professional networks'*, *'text message'* and others. These are the most common modes scammers use to approach victims online [243]. We use rule-based heuristics and Spacy NER [244] to identify these entities from text. There are 2.2K unique money values, 1.9K unique dates, 1.1K unique URLs, 4.6K unique organisations, 0.2K unique phone numbers, and 55 individual states from employment scam complaints content. We define seven types of relationships (*'reported_from'*, *'reported_date'*, *'job_type'*, *'contact'*,*'mentions_money'*, *'mentions_org'*, and *'email')* from these complaints. To integrate the knowledge, we extract these entities and relationships and store them efficiently in graphical storage, i.e., Neo4j Cypher Query Framework [1]. Figure 9.2 shows a sample snapshot of an Employment Scam Knowledge Graph. **Discussion:** Our findings in section 9.2 show that 61% of the complaints mentioned *'email'* and 12% are engaged through *'online professional networks'*. 10% scammed through *'online social networks'*, usually Facebook chats. At least 500 scam complaints mentions *'work from home'*, *'Covid'* or *'Corona'*. For scammers, email is the most common method of approaching the victim. We observe that scammers impersonated well-known retailers like *Amazon* and *Walmart* to seem legitimate, posting scam jobs on major online employment platforms. We find that the money or amount mentioned in the complaint text was mostly $32). Interestingly, there was a surge in scams during late 2018, where scammers impersonated popular business names using Amazon jobs. Finally, such information, when converted to KG, is helpful for victims to protect themselves from potential scams.

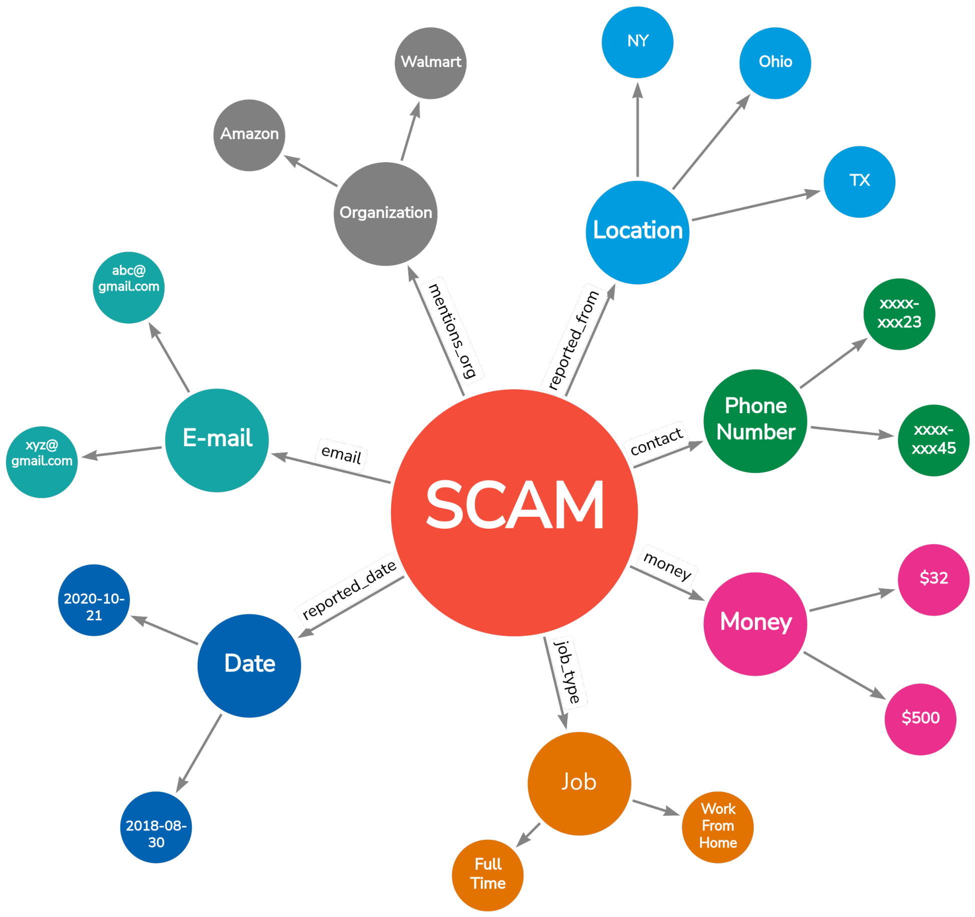


Figure 9.2: A sample snapshot of Employment Scam Knowledge Graph. All the entity types are shown in bold. We show edges among the entity types in the snapshot for better visualisation. For instance, a fact in the KG is of the form (Scam, mentions_org, Amazon).

References

[1] Jim Webber and Ian Robinson. *A programmatic introduction to neo4j*. Addison-Wesley Professional, 2018.

[2] Nausheen Fatma, Vijay Choudhary, Niharika Sachdeva, and Nitendra Rajput. Canonicalizing knowledge bases for recruitment domain. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 500–513. Springer, 2020.

[3] Mausam Mausam. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016.

[4] Syed Mahbub and Eric Pardede. Using contextual features for online recruitment fraud detection. 2018.

[5] Indeed. https://www.indeed.com.

[6] URL https://www.simplyhired.co.in/.

[7] Monster. https://www.monsterindia.com/.

[8] Glassdoor. URL https://glassdoor.com/.

[9] URL https://economicgraph.linkedin.com/.

[10] Careerbuilder. URL http://www.careerbuilder.com/.

[11] Naukri. https://www.naukri.com.

[12] Stackexchange. https://stackexchange.com/, .

[13] Stack overflow. http://stackoverflow.com, .

[14] Blind. https://www.teamblind.com/.

[15] Infoedge. https://www.infoedge.in/pdfs/News_Events_pdfs/Naukri-Jobspeak-May-2023.pdf.

[16] So questions. https://stackoverflow.com/questions.

[17] Online recruitment market. https://www.fortunebusinessinsights.com/online-recruitment-market-103730.

[18] How will the recruitment industry change over the next 10 years?, 2018. URL https://www.talentlyft.com/en/blog/article/243/how-will-the-recruitment-industry-change-over-the-next-10-years.

[19] Shaha T Al-Otaibi and Mourad Ykhlef. Job recommendation systems for enhancing e-recruitment process. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2012.

[20] Jan Tegze. Top 11 reasons why fake linkedin profiles exist, 2019. URL https://medium.com/@jantegze/top-11-reasons-why-fake-linkedin-profiles-exist-e77264b9d90f.

[21] Carlo Batini, Cinzia Cappiello, Chiara Francalanci, and Andrea Maurino. Methodologies for data quality assessment and improvement. *ACM computing surveys (CSUR)*, 41 (3):1–52, 2009.

[22] Tom Breur. Data quality is everyone's business—designing quality into your data warehouse—part 1. *Journal of Direct, Data and Digital Marketing Practice*, 11:20–29, 2009.

[23] Xavier Pleimling, Vedant Shah, and Ismini Lourentzou. [data] quality lies in the eyes of the beholder. In *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*, pages 118–124, 2022.

[24] Baoshi Yan, Lokesh Bajaj, and Anmol Bhasin. Entity resolution using social graphs for business applications. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 220–227. IEEE, 2011.

[25] Akshay Bhola, Kishaloy Halder, Animesh Prasad, and Min-Yen Kan. Retrieving skills from job descriptions: A language model based extreme multi-label classification framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5832–5842, 2020.

[26] Shawni Dutta and Samir Kumar Bandyopadhyay. Fake job recruitment detection using machine learning approach.

[27] Bandar Alghamdi and Fahad Alharby. An intelligent model for online recruitment fraud detection. *Journal of Information Security*, 10(3):155–176, 2019.

[28] Fake candidate. https://welovesalt.com/news/hiring-advice/fake-candidate-cvs-curse-real-recruiters/.

[29] Amit Singhal. Introducing the knowledge graph: things, not strings, May 2012. URL https://www.blog.google/products/search/introducing-knowledge-graph-things-not/. [Online; updated 16-May-2012].

[30] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 1375–1384, 2011.

[31] Muhammad Asaduzzaman, Ahmed Mashiyat, Chanchal Roy, and Kevin Schneider. Answering questions about unanswered questions of stack overflow. pages 97–100, 05 2013. ISBN 978-1-4799-0345-0. doi: 10.1109/MSR.2013.6624015.

[32] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

[33] Deepika Kumawat and Vinesh Jain. Pos tagging approaches: A comparison. *International Journal of Computer Applications*, 118(6), 2015.

[34] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 2013.

[35] Qiaoling Liu, Faizan Javed, Vachik S Dave, and Ankita Joshi. Supporting employer name normalization at both entity and cluster level. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1883–1892, 2017.

[36] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2015.

[37] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. *arXiv preprint arXiv:1908.08210*, 2019.

[38] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 2008.

[39] Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. Canonicalizing open knowledge bases. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management*, pages 1679–1688, 2014.

[40] Akshay Bhola, Kishaloy Halder, Animesh Prasad, and Min-Yen Kan. Retrieving Skills from Job Descriptions: A Language Model Based Extreme Multi-label Classification Framework. pages 5832–5842, 2021. doi: 10.18653/v1/2020.coling-main.513.

[41] NINANDE VERMEER, VERA PROVATOROVA, DAVID GRAUS, THILINA RAJAPAKSE, and SEPIDEH MESBAH. Using robbert and extreme multi-label classification to extract implicit and explicit skills from dutch job descriptions. *acm*, 2020.

[42] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[44] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1209.

[45] Laurel Lord, John Sell, Feyzi Bagirov, and Mark Newman. Survival analysis within stack overflow: Python and r. In *2018 4th International Conference on Big Data Innovations and Applications (Innovate-Data)*, pages 51–59. IEEE Computer Society, 2018.

[46] Aditi Gupta and Ponnurangam Kumaraguru. Credibility ranking of tweets during high impact events. In *Proceedings of the 1st workshop on privacy and security in online social media*, pages 2–8, 2012.

[47] Aditi Gupta, Ponnurangam Kumaraguru, Carlos Castillo, and Patrick Meier. Tweetcred: Real-time credibility assessment of content on twitter. In *Social Informatics: 6th*

International Conference, SocInfo 2014, Barcelona, Spain, November 11-13, 2014. Proceedings 6, pages 228–243. Springer, 2014.

[48] Rong Yan and John Hegeman. Using polling results as discrete metrics for content quality prediction model, February 28 2017. US Patent 9,582,812.

[49] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 183–194, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939272. doi: 10.1145/1341531.1341557. URL https://doi.org/10.1145/1341531.1341557.

[50] Sai T Moturu and Huan Liu. Quantifying the trustworthiness of social media content. *Distributed and Parallel Databases*, 29:239–260, 2011.

[51] Prateek Dewan and Ponnurangam Kumaraguru. Facebook inspector (fbi): Towards automatic real-time detection of malicious content on facebook. *Social Network Analysis and Mining*, 7(1):15, 2017.

[52] Andrew Guess, Jonathan Nagler, and Joshua Tucker. Less than you think: Prevalence and predictors of fake news dissemination on facebook. *Science advances*, 5(1): eaau4586, 2019.

[53] Aditi Gupta, Hemank Lamba, and Ponnurangam Kumaraguru. $1.00 per rt# bostonmarathon# prayforboston: Analyzing fake content on twitter. In *2013 APWG eCrime researchers summit*, pages 1–12. IEEE, 2013.

[54] Denzil Correa and Ashish Sureka. Fit or unfit: analysis and prediction of'closed questions' on stack overflow. In *Proceedings of the first ACM conference on Online social networks*, pages 201–212, 2013.

[55] Haifa Alharthi, Djedjiga Outioua, and Olga Baysal. Predicting questions' scores on stack overflow. In *2016 IEEE/ACM 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*, pages 1–7. IEEE, 2016.

[56] Antoaneta Baltadzhieva and Grzegorz Chrupała. Predicting the quality of questions on stackoverflow. In *Proceedings of the international conference recent advances in natural language processing*, pages 32–40, 2015.

[57] Pradeep Kumar Roy. Multilayer convolutional neural network to filter low quality content from quora. *Neural Processing Letters*, pages 1–17, 2020.

[58] Mohammad A Al-Ramahi and Izzat Alsmadi. Using data analytics to filter insincere posts from online social networks. a case study: Quora insincere questions. 2020.

[59] Suman Maity, Jot Sarup Singh Sahni, and Animesh Mukherjee. Analysis and prediction of question topic popularity in community q&a sites: a case study of quora. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 9, pages 238–247, 2015.

[60] Wern Han Lim, Mark James Carman, and Sze-Meng Jojo Wong. Estimating relative user expertise for content quality prediction on reddit. In *Proceedings of the 28th ACM conference on hypertext and social media*, pages 55–64, 2017.

[61] Yongcheng Zhan, Zhu Zhang, Janet M Okamoto, Daniel D Zeng, and Scott J Leischow. Underage juul use patterns: Content analysis of reddit messages. *Journal of medical Internet research*, 21(9):e13038, 2019.

[62] Shagun Jhaver, Iris Birman, Eric Gilbert, and Amy Bruckman. Human-machine collaboration for content regulation: The case of reddit automoderator. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(5):1–35, 2019.

[63] Shikhar Vashishth, Prince Jain, and Partha Talukdar. CESI: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1317–1327, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5639-8.

[64] Swapnil Gupta, Sreyash Kenkre, and Partha Talukdar. Care: Open knowledge graph embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 378–388, 2019.

[65] Liting Liu, Jie Liu, Wenzheng Zhang, Ziming Chi, Wenxuan Shi, and Yalou Huang. Hiring Now: A Skill-Aware Multi-Attention Model for Job Posting Generation. Technical report.

[66] Qiaoling Liu, Faizan Javed, and Matt Mcnair. Companydepot: Employer name normalization in the online recruitment industry. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 521–530, 2016.

[67] Qiaoling Liu, Josh Chao, Thomas Mahoney, Alan Chern, Chris Min, Faizan Javed, and Valentin Jijkoun. Lessons learned from developing and deploying a large-scale employer name normalization system for online recruitment. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 556–565, 2018.

[68] Sokratis Vidros, Constantinos Kolias, Georgios Kambourakis, and Leman Akoglu. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, 9(1):6, 2017.

[69] Greet Baldewijns, Glen Debard, Gert Mertes, Tom Croonenborghs, and Bart Vanrumste. Improving the accuracy of existing camera based fall detection algorithms through late fusion. In *2017 39th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 2667–2671. IEEE, 2017.

[70] Jack E Olson. *Data quality: the accuracy dimension*. Elsevier, 2003.

[71] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[72] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics, 2012.

[73] Surafel Lemma Abebe and Paolo Tonella. Towards the extraction of domain concepts from the identifiers. In *2011 18th Working Conference on Reverse Engineering*, pages 77–86. IEEE, 2011.

[74] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *International Semantic Web Conference*, pages 542–557. Springer, 2013.

[75] https://www.aajtak.in/education/news/story/iit-delhi-seeks-btech-degree-for-dog-handler-post-job-now-gave-clarification-tedu-1125805-2020-09-07.

[76] Radityo Eko Prasojo, Mouna Kacimi, and Werner Nutt. Stuffie: Semantic tagging of unlabeled facets using fine-grained information extraction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 467–476, 2018.

[77] Prantika Chakraborty, Sudakshina Dutta, and Debarshi Kumar Sanyal. Personal research knowledge graphs. In *Companion Proceedings of the Web Conference 2022*, WWW

’22, page 763–768, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391306. URL https://doi.org/10.1145/3487553.3524654.

[78] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*. Atlanta, 2010.

[79] Natthawut Kertkeidkachorn and Ryutaro Ichise. T2kg: An end-to-end system for creating knowledge graph from unstructured text. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[80] Isabelle Augenstein, Sebastian Padó, and Sebastian Rudolph. Lodifier: Generating linked data from unstructured text. In *Extended Semantic Web Conference*, pages 210–224. Springer, 2012.

[81] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*. ACM, 2015.

[82] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. Acekg: A large-scale knowledge graph for academic data mining. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018.

[83] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[84] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[85] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2013.

[86] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[87] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[88] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[89] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[90] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[91] Yuanfei Dai, Shiping Wang, Neal N Xiong, and Wenzhong Guo. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750, 2020.

[92] Jing Wang. A survey on graph neural networks. *EAI Endorsed Transactions on e-Learning*, 8(3):e6–e6, 2022.

[93] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[94] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

[95] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[96] Viet-Phi Huynh and Paolo Papotti. Buckle: Evaluating fact checking algorithms built on knowledge bases. *Proceedings of the VLDB Endowment*, 12(12):1798–1801, 2019.

[97] Baoxu Shi and Tim Weninger. Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 101–102, 2016.

[98] Prashant Shiralkar, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. Finding streams in knowledge graphs to support fact checking. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 859–864. IEEE, 2017.

[99] Jeff Z Pan, Siyana Pavlova, Chenxi Li, Ningxi Li, Yangmei Li, and Jinshuo Liu. Content based fake news detection using knowledge graphs. In *International Semantic Web Conference*, pages 669–683. Springer, 2018.

[100] Saransh Khandelwal and Dhananjay Kumar. Computational fact validation from knowledge graph using structured and unstructured information. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, pages 204–208. 2020.

[101] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[102] Tanveer Ahmed and Abhishek Srivastava. Understanding and evaluating the behavior of technical users. a study of developer interaction at stackoverflow. *Human-centric Computing and Information Sciences*, 7(1):8, 2017.

[103] Doris Hoogeveen, Andrew Bennett, Yitong Li, Karin M Verspoor, and Timothy Baldwin. Detecting misflagged duplicate questions in community question-answering archives. In *Twelfth international AAAI conference on web and social media*, 2018.

[104] Rodrigo FG Silva, Klérisson Paixão, and Marcelo de Almeida Maia. Duplicate question detection in stack overflow: A reproducibility study. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 572–581. IEEE, 2018.

[105] Di Liang, Fubao Zhang, Weidong Zhang, Qi Zhang, Jinlan Fu, Minlong Peng, Tao Gui, and Xuanjing Huang. Adaptive multi-attention network incorporating answer information for duplicate question detection. *SIGIR 2019 - Proceedings of the 42nd International ACM SIGIR Conference on R&D in IR*, pages 95–104, 2019.

[106] Deepak Thukral, Adesh Pandey, Rishabh Gupta, Vikram Goyal, and Tanmoy Chakraborty. Diffque: Estimating relative difficulty of questions in community question answering services. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(4):1–27, 2019.

[107] Pradeep Kumar Roy and Jyoti Prakash Singh. Predicting closed questions on community question answering sites using convolutional neural network. *Neural Computing and Applications*, pages 1–18, 2019.

[108] Jan Trienes and Krisztian Balog. Identifying unclear questions in community question answering websites. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41*, pages 276–289. Springer, 2019.

[109] Imane Khaouja, Ismail Kassou, and Mounir Ghogho. A survey on skill identification from online job ads. *IEEE Access*, 9:118134–118153, 2021.

[110] Tong Xu, Hengshu Zhu, Chen Zhu, Pan Li, and Hui Xiong. Measuring the popularity of job skills in recruitment market: A multi-criteria approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[111] Liting Liu, Jie Liu, Wenzheng Zhang, Ziming Chi, Wenxuan Shi, and Yalou Huang. Hiring now: A skill-aware multi-attention model for job posting generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3096–3104, 2020.

[112] Akshay Bhola, Kishaloy Halder, Animesh Prasad, and Min-Yen Kan. Retrieving skills from job descriptions: A language model based extreme multi-label classification framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5832–5842, 2020.

[113] Jooyeon Kim, Behzad Tabibian, Alice Oh, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. Leveraging the crowd to detect and reduce the spread of fake news and misinformation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 324–332, 2018.

[114] Xueling Lin and Lei Chen. Canonicalization of open knowledge bases with side information from the source text. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 950–961. IEEE, 2019.

[115] Anja Zojceska. Social recruiting: A complete guide on how to recruit on facebook, 2017. URL https://www.talentlyft.com/en/blog/article/82/social-recruiting-a-complete-guide-on-how-to-recruit-on-facebook.

[116] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, 2015.

[117] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[118] Linh Le and Ying Xie. Deep embedding kernel. *Neurocomputing*, 339:292–302, 2019.

[119] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, 15, 2002.

[120] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. 2008.

[121] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[122] European Commission. *ESCO handbook*. EU publications, 2019.

[123] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, volume 2, pages 3–2, 2016.

[124] Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8, 2011.

[125] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pages 639–655. Springer, 2012.

[126] Bor-Chen Kuo, Hsin-Hua Ho, Cheng-Hsuan Li, Chih-Cheng Hung, and Jin-Shiuh Taur. A kernel-based feature selection method for svm with rbf kernel for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(1):317–326, 2013.

[127] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://www.aclweb.org/anthology/D19-1410.

[128] Artur Starczewski and Adam Krzyżak. Performance evaluation of the silhouette index. In *International Conference on Artificial Intelligence and Soft Computing*, pages 49–58. Springer, 2015.

[129] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[130] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://www.aclweb.org/anthology/D19-1410.

[131] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

[132] Vijay Raghavan, Peter Bollmann, and Gwang S Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989.

[133] Sameep Mehta, Rakesh Pimplikar, Amit Singh, Lav R Varshney, and Karthik Visweswariah. Efficient multifaceted screening of job applicants. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 661–671, 2013.

[134] Silvia Fareri, Nicola Melluso, Filippo Chiarello, and Gualtiero Fantoni. Skillner: Mining and mapping soft skills from any text. *Expert Systems with Applications*, 184:115544, 2021.

[135] Mike Zhang, Kristian Nørgaard Jensen, Sif Dam Sonniks, and Barbara Plank. Skillspan: Hard and soft skill extraction from english job postings. *arXiv preprint arXiv:2204.12811*, 2022.

[136] Akshay Gugnani and Hemant Misra. Implicit skills extraction using document embedding and its use in job recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13286–13293, 2020.

[137] Ilkka Kivimäki, Alexander Panchenko, Adrien Dessy, Dries Verdegem, Pascal Francq, Hugues Bersini, and Marco Saerens. A graph-based approach to skill extraction from text. In *Proceedings of TextGraphs-8 graph-based methods for natural language processing*, pages 79–87, 2013.

[138] Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272, 2014.

[139] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. *Social network data analytics*, pages 243–275, 2011.

[140] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[141] Deepak Saini, Arnav Kumar Jain, Kushal Dave, Jian Jiao, Amit Singh, Ruofei Zhang, and Manik Varma. Galaxc: Graph neural networks with labelwise attention for extreme classification. In *Proceedings of the Web Conference 2021*, pages 3733–3744, 2021.

[142] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

[143] Zhenpeng Zhou and Xiaocheng Li. Graph convolution: A high-order and adaptive approach. *arXiv preprint arXiv:1706.09916*, 2017.

[144] Kameni Florentin Flambeau Jiechieu and Norbert Tsopze. Skills prediction based on multi-label resume classification using cnn with model predictions explanation. *Neural Computing and Applications*, 33:5069–5087, 2021.

[145] Sujay Khandagale, Han Xiao, and Rohit Babbar. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109:2099–2119, 2020.

[146] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers. *arXiv preprint arXiv:1905.02331*, 2019.

[147] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM*

SIGIR conference on research and development in information retrieval, pages 115–124, 2017.

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

[149] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[150] Yashoteja Prabhu, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 441–449, 2018.

[151] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 528–536, 2019.

[152] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL https://aclanthology.org/D14-1181.

[153] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[154] Chengjie Sun, Yang Liu, Chang'e Jia, Bingquan Liu, and Lei Lin. Recognizing text entailment via bidirectional lstm model with inner-attention. In *International Conference on Intelligent Computing*, pages 448–457. Springer, 2017.

[155] Kishaloy Halder, Lahari Poddar, and Min-Yen Kan. Cold start thread recommendation as extreme multi-label classification. In *Companion Proceedings of the The Web Conference 2018*, pages 1911–1918, 2018.

[156] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019.

[157] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[158] Simran Arora, Avner May, Jian Zhang, and Christopher Ré. Contextual embeddings: When are they worth it? *arXiv preprint arXiv:2005.09117*, 2020.

[159] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[160] Fukun Chen, Guisheng Yin, Yuxin Dong, Gesu Li, and Weiqi Zhang. Khgcn: Knowledge-enhanced recommendation with hierarchical graph capsule network. *Entropy*, 25(4):697, 2023.

[161] Shiyan Ou, Viktor Pekar, Constantin Orasan, Christian Spurk, and Matteo Negri. Development and alignment of a domain-specific ontology for question answering. In *LREC*. Citeseer, 2008.

[162] Shaik Masihullah, Meghana Negi, Jose Matthew, and Jairaj Sathyanarayana. Identifying fraud rings using domain aware weighted community detection. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 150–167. Springer, 2022.

[163] Suk-Chung Yoon, Lawrence J Henschen, EK Park, and Sam Makki. Using domain knowledge in knowledge discovery. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 243–250, 1999.

[164] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. Knowledge graphs: Opportunities and challenges. *Artificial Intelligence Review*, pages 1–32, 2023.

[165] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, (3), 2010.

[166] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.

[167] Yordanos Beyene, Michalis Faloutsos, Duen Horng Chau, and Christos Faloutsos. The ebay graph: How do online auction users interact? In *IEEE INFOCOM Workshops 2008*, pages 1–6. IEEE, 2008.

[168] Xi Chen, Yiqun Liu, Liang Zhang, and Krishnaram Kenthapadi. How linkedin economic graph bonds information and product: applications in linkedin salary. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 120–129, 2018.

[169] Giovanni Maria Biancofiore, Tommaso Di Noia, Eugenio Di Sciascio, Fedelucio Narducci, and Paolo Pastore. Guapp: Enhancing job recommendations with knowledge graphs. In *IIR*, 2021.

[170] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[171] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[172] Luciano Del Corro and Rainer Gemulla. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366, 2013.

[173] Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1034. URL https://aclanthology.org/P15-1034.

[174] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[175] Afroza Sultana, Quazi Mainul Hasan, Ashis Kumer Biswas, Soumyava Das, Habibur Rahman, Chris Ding, and Chengkai Li. Infobox suggestion for wikipedia entities. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012.

[176] Divy Thakkar, Neha Kumar, and Nithya Sambasivan. Towards an ai-powered future that works for vocational workers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[177] Bandar Alghamdi and Fahad Alharby. An intelligent model for online recruitment fraud detection. *Journal of Information Security*, 10(3):155–176, 2019.

[178] M Niharika Reddy, T Mamatha, and A Balaram. Analysis of e-recruitment systems and detecting e-recruitment fraud. In *International Conference on Communications and Cyber Physical Engineering 2018*, pages 411–417. Springer, 2018.

[179] Shaden Shaar, Nikolay Babulkov, Giovanni Da San Martino, and Preslav Nakov. That is a known lie: Detecting previously fact-checked claims. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3607–3618, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.332. URL https://aclanthology.org/2020.acl-main.332.

[180] Divy Thakkar, Neha Kumar, and Nithya Sambasivan. Towards an ai-powered future that works for vocational workers. In *proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[181] Okti Nindyati and I Gusti Bagus Baskara Nugraha. Detecting scam in online job vacancy using behavioral features extraction. In *2019 International Conference on ICT for Smart Society (ICISS)*, volume 7, pages 1–4. IEEE, 2019.

[182] Jeongrae Kim, Han-Joon Kim, and Hyoungrae Kim. Fraud detection for job placement using hierarchical clusters-based deep neural networks. *Applied Intelligence*, 49 (8):2842–2861, 2019.

[183] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193, 2015.

[184] James Thorne and Andreas Vlachos. Automated fact checking: Task formulations, methods and future directions. In Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1283.

[185] Viet-Phi Huynh and Paolo Papotti. Towards a benchmark for fact checking with knowledge bases. In *Companion Proceedings of the The Web Conference 2018*, pages 1595–1598, 2018.

[186] Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539, 2011.

[187] Baoxu Shi and Tim Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-based systems*, 104:123–133, 2016.

[188] Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[189] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.

[190] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, 2016.

[191] Liang-Chun Chen, Chien-Lung Hsu, Nai-Wei Lo, Kuo-Hui Yeh, and Ping-Hsien Lin. Fraud analysis and detection for real-time messaging communications on social networks. *IEICE TRANSACTIONS on Information and Systems*, 100(10):2267–2274, 2017.

[192] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*, 2017.

[193] Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm. From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 84–89, 2017.

[194] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. Detecting rumors from microblogs with recurrent neural networks. 2016.

[195] Yi-Ju Lu and Cheng-Te Li. Gcan: Graph-aware co-attention networks for explainable fake news detection on social media. *arXiv preprint arXiv:2004.11648*, 2020.

[196] Gerald Ki Wei Huang and Jun Choi Lee. Hyperpartisan news and articles detection using bert and elmo. In *2019 International Conference on Computer and Drone Applications (IConDA)*, pages 29–32. IEEE, 2019.

[197] Jan Christian Blaise Cruz, Julianne Agatha Tan, and Charibeth Cheng. Localization of fake news detection via multitask transfer learning. *arXiv preprint arXiv:1910.09295*, 2019.

[198] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[199] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2016.

[200] Jamshid Mozafari, Afsaneh Fatemi, and Parham Moradi. A method for answer selection using distilbert and important words. In *2020 6th International Conference on Web Research (ICWR)*, pages 72–76. IEEE, 2020.

[201] Keke Gai and Meikang Qiu. Reinforcement learning-based content-centric services in mobile sensing. *IEEE Network*, 32(4):34–39, 2018.

[202] Keke Gai, Meikang Qiu, Hui Zhao, and Xiaotong Sun. Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. *IEEE Transactions on Sustainable Computing*, 3(2):60–72, 2017.

[203] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[204] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. *arXiv preprint arXiv:1709.07417*, 2017.

[205] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.

[206] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[207] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[208] Geoffrey Holmes, Andrew Donkin, and Ian H Witten. Weka: A machine learning workbench. In *Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference*, pages 357–361. IEEE, 1994.

[209] Sancho Salcedo-Sanz, José Luis Rojo-Álvarez, Manel Martínez-Ramón, and Gustavo Camps-Valls. Support vector machines in engineering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3):234–267, 2014.

[210] Markus Nivala, Alena Seredko, Tanya Osborne, and Thomas Hillman. Stack overflow–informal learning and the global expansion of professional development and opportunities in programming? In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 402–408. IEEE, 2020.

[211] Denzil Correa and Ashish Sureka. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In *Proceedings of the 23rd international conference on World wide web*, pages 631–642, 2014.

[212] Sujith Ravi, Bo Pang, Vibhor Rastogi, and Ravi Kumar. Great question! question quality in community q&a. 2014.

[213] Mahmood Neshati. On early detection of high voted q&a on stack overflow. *Information Processing & Management*, 53(4):780–798, 2017.

[214] Mun Kit Ho, Sivanagaraja Tatinati, and Andy WH Khong. Distilling essence of a question: A hierarchical architecture for question quality in community question answering sites. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[215] László Tóth, Balázs Nagy, Dávid Janthó, László Vidács, and Tibor Gyimóthy. Towards an accurate prediction of the question quality on stack overflow using a deep-learning-based nlp approach. In *ICSOFT*, pages 631–639, 2019.

[216] László Tóth, Balázs Nagy, Tibor Gyimóthy, and László Vidács. Why will my question be closed? nlp-based pre-submission predictions of question closing reasons on stack overflow. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 45–48. IEEE, 2020.

[217] Edward L Kaplan and Paul Meier. Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, 53(282):457–481, 1958.

[218] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[219] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377, 2019.

[220] David W Hosmer Jr, Stanley Lemeshow, and Susanne May. *Applied survival analysis: regression modeling of time-to-event data*, volume 618. John Wiley & Sons, 2011.

[221] Niharika Sachdeva and Ponnurangam Kumaraguru. Call for service: Characterizing and modeling police response to serviceable requests on facebook. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 336–352, 2017.

[222] Siddhartha Asthana, Pushpendra Singh, and Parul Gupta. Survival analysis: objective assessment of wait time in hci. In *Proceedings of the 33rd annual acm conference on human factors in computing systems*, pages 367–376, 2015.

[223] Felipe Ortega, Gregorio Convertino, Massimo Zancanaro, and Tiziano Piccardi. Assessing the performance of question-and-answer communities using survival analysis. *CoRR*, abs/1407.5903, 2014.

[224] Laurel Lord, John Sell, Feyzi Bagirov, and Mark Newman. Survival analysis within stack overflow: Python and r. In *2018 4th International Conference on Big Data Innovations and Applications (Innovate-Data)*, pages 51–59. IEEE Computer Society, 2018.

[225] Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. Code and named entity recognition in stackoverflow. In *The Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[226] Seba Susan, Dhaarna Sethi, and Kriti Arora. Cw-cae: pulmonary nodule detection from imbalanced dataset using class-weighted convolutional autoencoder. In *International Conference on innovative computing and communications*, pages 825–833. Springer, 2021.

[227] Ali Miller. Comparing the seminonparametric survival function estimator to the kaplan-meier estimator and equivalent parametric methods for rightcensored data.

[228] Waqas Arshad, Muhammad Ali, Muhammad Mumtaz Ali, Arifa Javed, and Saddam Hussain. Multi-class text classification: Model comparison and selection. In *2021*

International Conference on Electrical, Communication, and Computer Engineering (ICECCE), pages 1–5, 2021. doi: 10.1109/ICECCE52056.2021.9514108.

[229] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1 (4):1–4, 2015.

[230] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[231] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. IEEE, 2018.

[232] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[233] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, 1901. doi: 10.1080/14786440109462720.

[234] Qianren Mao, Yiming Wang, Chenghong Yang, Linfeng Du, Hao Peng, Jia Wu, Jianxin Li, and Zheng Wang. Higil: Hierarchical graph inference learning for fact checking. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 329–337. IEEE, 2022.

[235] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193, 2015.

[236] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*, 2017.

[237] Alexandrea J Ravenelle, Erica Janko, and Ken Cai Kowalski. Good jobs, scam jobs: Detecting, normalizing, and internalizing online job scams during the covid-19 pandemic. *New Media & Society*, 24(7):1591–1610, 2022.

[238] Syed Mahbub, Eric Pardede, and ASM Kayes. Online recruitment fraud detection: A study on contextual features in australian job industries. *IEEE Access*, 10:82776–82787, 2022.

[239] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4):1–37, 2021.

[240] Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *International semantic web conference*, pages 177–185. Springer, 2016.

[241] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. Introduction: what is a knowledge graph? In *Knowledge Graphs*, pages 1–10. Springer, 2020.

[242] Kuldeep Singh, Ioanna Lytra, Arun Sethupat Radhakrishna, Saeedeh Shekarpour, Maria-Esther Vidal, and Jens Lehmann. No one is perfect: Analysing the performance of question answering components over the dbpedia knowledge graph. *Journal of Web Semantics*, 65:100594, 2020.

[243] Deanna Grant-Smith, Alicia Feldman, and Cassandra Cross. Key trends in employment scams in australia: What are the gaps in knowledge about recruitment fraud. *QUT Centre for Justice Briefing Papers*, 21, 2022.

[244] Xavier Schmitt, Sylvain Kubler, Jérémy Robert, Mike Papadakis, and Yves LeTraon. A replicable comparison study of ner software: Stanfordnlp, nltk, opennlp, spacy, gate. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 338–343. IEEE, 2019.