



Efficient Text Extraction Methodologies for Sentiment Analysis: Utilizing University of South Africa Students' Email Communications as a Case Study

Malusi Sibiyi

Computer Science Department, University of South Africa, Florida, South Africa.

Email: sibiym@unisa.ac.za

ARTICLE INFO

Received: 06 May 2024
Accepted: 13 Sep 2024

ABSTRACT

After the advent of transformers, highlighted in the paper 'All You Need is Attention' by Vaswani et al. [1], Large Language Models (LLMs) gained significant traction, notably for tasks like sentiment analysis due to their improved accuracy. Our study focuses on devising a systematic approach to extract textual data from student emails addressed to (University of South Africa) UNISA staff members via the Viva Engage platform. UNISA has customized Yammer, a Microsoft-owned platform typically used for enterprise collaboration and communication, into Viva Engage. Within this platform, students utilize it to express their concerns and opinions on various issues. Given that Viva Engage is intricately linked with UNISA staff emails, every staff member is promptly notified of incoming student messages. Our primary objective is to mine this data for sentiment analysis, aiming to discern the prevalent concerns among students. Specifically, our study delineates the methodologies employed for data extraction and preprocessing tailored for sentiment analysis utilizing LLMs. It is worth noting that this paper exclusively addresses the data extraction phase from Viva Engage emails. The subsequent sentiment analysis, utilizing a BERT LLM, is elaborated upon in a separate research endeavor.

Keywords: Large Language Models, Viva Engage, BERT, Email.

INTRODUCTION

In contemporary organizational settings, efficient communication and collaboration are crucial for productivity and cohesion. Platforms like Yammer serve as pivotal tools, facilitating internal communication within enterprises. Extracting data from Yammer notifications embedded within emails becomes imperative for analyzing organizational discourse and engagement patterns. This paper presents a methodological approach to retrieve and parse Yammer notifications from Outlook emails using Python written code. Common student comments on the Yammer platform, customized as Viva Engage at UNISA, mainly centers around NSFAS. NSFAS, the National Student Financial Aid Scheme, is

funded by the Department of Higher Education and Training, offering financial assistance to undergraduate students for their tertiary education expenses after high school. Several researchers have examined the advantages and disadvantages of NSFAS [2,3]. For instance, Jackson investigated the functioning of NSFAS [2], while Yende explored the challenges of its model [3]. This study focused on extracting the NSFAS-related emails sent via Viva Engage by students to communicate with lecturing staff and management at large. Existing literature provides evidence of Microsoft's Yammer platform being widely used by organizations for internal communications. Section 2 reviews the applications of Yammer across different industries.

LITERATURE REVIEW

In this study, Riemer et al. [4] aimed to contribute to the understanding of the role and impact of social technologies in enterprises, particularly in knowledge-intensive work environments. The phenomenon of Enterprise Social Networking (ESN) was investigated within Professional Service Firms (PSF). The case study focused on emerging communicative work practices on the ESN platform Yammer within Deloitte Australia. A genre analysis was conducted on actual communication data captured on the Yammer platform. The study revealed a set of emerging practices facilitated by the platform within the case company. Results were reflected upon in the context of the knowledge-intensive nature of professional service work. It was found that within the case company, Yammer served as an information-sharing channel, a space for crowdsourcing ideas, a platform for finding expertise and solving problems, and a medium for conversation conducive to context and relationship building. Borge and Goggins evaluated the quality of interactions on Yammer by fostering an online community of learners [5]. Pinto and other researchers, such as Pinto [6], conducted an exploratory study on the utilization of Yammer in Higher Education. Pinto's study specifically delved into the application of Yammer for communication and collaboration within project teams in an upper-level marketing course. The research revealed a significant positive correlation between Yammer usage and communication effectiveness. Despite initial hesitancy among student respondents, the findings highlighted substantial online activity among college students and underscored the potential benefits of integrating Yammer or similar social media technologies into educational settings to enhance collaboration. These insights emphasize the significance of digital communication tools in educational contexts and suggest implications for future instructional practices. Munusamy et al. utilized Yammer and Socrative online tools to foster interactive learning in pharmacy education [7]. Lytvynova and Naboka used Yammer cloud service to organize project-based learning methods [8]. Their study analyzed the core elements of project-based learning facilitated by cloud-based services, encompassing social engagement, streamlined team communication across project phases, an inclusive educational milieu, autonomous learning, interdisciplinary collaboration, and the cultivation of digital literacy among students. It also assessed the strengths and limitations of Yammer as a cloud service, conducting a comparative analysis with analogous platforms and illustrating its application in professional project contexts. Additionally, it scrutinized the stages of project methodology employing the small group approach, including initiation, planning, execution, presentation, and evaluation, offering comprehensive analysis and insights into each phase. Other research studies employed Yammer for software related research [9,10]. For instance, Radityatama et al [9]., their study investigated the augmented functionalities of Next Generation Firewall (NGFW) in contrast to traditional firewalls, emphasizing its capability to scrutinize packet contents, thereby enhancing precision. NGFW served three primary roles: augmenting Quality of Service (QoS) for businesses, acting as an application-based filtering firewall, and shielding networks from known security threats. A

comprehensive NGFW system comprised three fundamental components: Deep Packet Inspection (DPI), Intrusion Prevention System (IPS), and an additional firewall intelligence mechanism. The research entailed an examination of an open-source DPI implementation termed nDPI, which faced limitations in enterprise software support amid the proliferation of enterprise applications. Consequently, the study aimed to devise and implement improved enterprise-grade software support protocols on nDPI. Five prevalent enterprise applications were selected and integrated, followed by a comparison of experimental outcomes with commercial NGFW implementations in terms of nDPI's overall precision and performance. Findings revealed that the accuracy of nDPI with the newly implemented protocols exceeded 90%, accompanied by a slight (less than 3.5%) increase in CPU execution time and minimal (less than 1%) elevation in peak heap memory usage. Yammer was one of the enterprise software platforms among the five examined in this study, which aimed to devise and apply protocols within nDPI to discern enterprise application protocols. Specifically, the investigation targeted Adobe Creative Suite, Microsoft SharePoint, Salesforce.com, Yammer, and Zendesk, with the exclusion of their mobile iterations from the research scope.

METHODOLOGY

UNISA's Yammer platform, customized as Viva Engage, is linked to the lecturing staff's Outlook emails. Mining email data directly from Outlook became a challenge since extracting emails using Python requires an Internet Message Access Protocol (IMAP). IMAP is a protocol used by email clients to retrieve emails from a mail server. Due to this, the client should have a valid email address from which it should receive emails. However, the UNISA's Viva Engage platform does not have a valid email address but is connected to the staff emails via the URL address. To solve this issue, we took advantage of Outlook's rules. We developed a rule that detected incoming emails by subject and used the subject "Updates from the University of South Africa, University of South Africa, and more at the University of South Africa," which is a common subject of messages delivered by Viva Engage to the staff emails. In the experiment, we targeted Viva Engage messages with NSFAS as a keyword and added it to the conditions of the Outlook rules. Finally, in the Outlook rules, we set Gmail email address where the Viva Engage messages related to NSFAS were to be delivered. Figure 1 shows how the rule to forward messages to a Gmail account was designed.

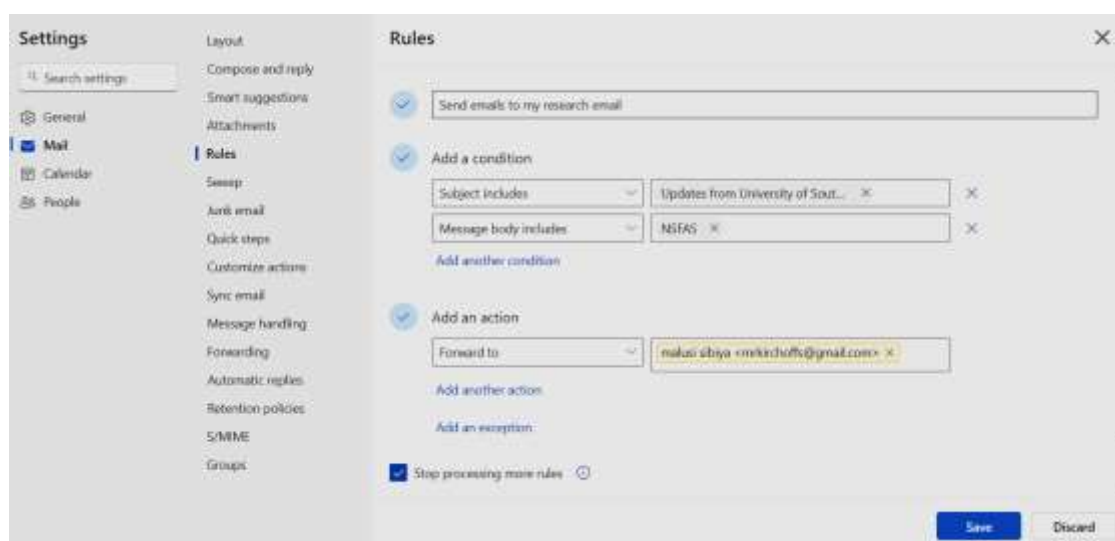


Figure 1. Design of a rule for forwarding messages from UNISA's Viva Engage (A customized Yammer platform) to a Gmail account.

Figure 2 depicts the complete process of sending emails from an Outlook email account to a Gmail email account for extraction of messages by Python written code.

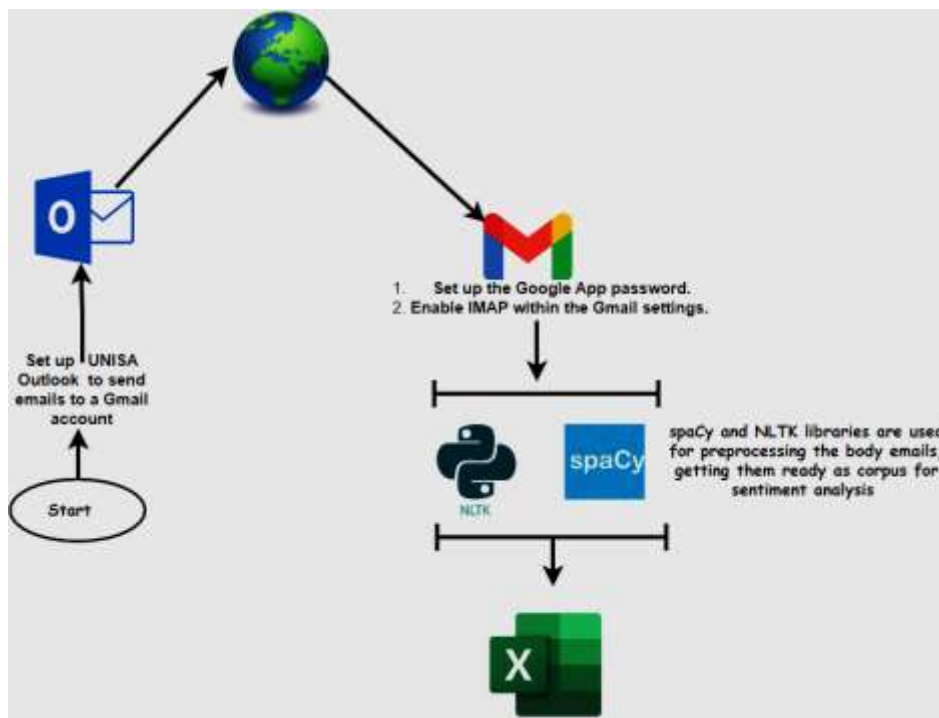


Figure 2. The complete process of sending emails from Outlook to Gmail then Python extraction of emails.

3.1 Experiment 1 – Setting up A Gmail Account for Email Extraction in Python

In the experiment conducted for extracting emails using Python, an App Password was generated in Google to enable access to Gmail accounts via the Internet Message Access Protocol (IMAP). The process of creating the App Password involved navigating to the Google Account settings and selecting the "Security" tab. From there, the option to generate an App Password was chosen. Following the prompts, the user selected the app and device for which the password was intended. In this case, Python was specified as the application. Google then generated a unique sixteen-character password specifically for Python to access the Gmail account securely. This App Password was utilized within the Python script to authenticate and retrieve emails from the Gmail server via IMAP, thus facilitating the experimental data extraction process.

3.2 Experiment 2 – Settings within the Gmail Account That Python Used for the Extraction of Emails

In experiment 2 we logged into the Gmail email account intended for message extraction and navigated to settings where we found the email settings option. Within this menu, we accessed the POP/IMAP settings, where we enabled IMAP to allow email clients to connect to the server using this protocol. After toggling the IMAP option, we saved the changes, ensuring that email retrieval via IMAP was enabled for the account. This process enabled us to access emails from the server using the IMAP protocol, facilitating synchronization across multiple devices.

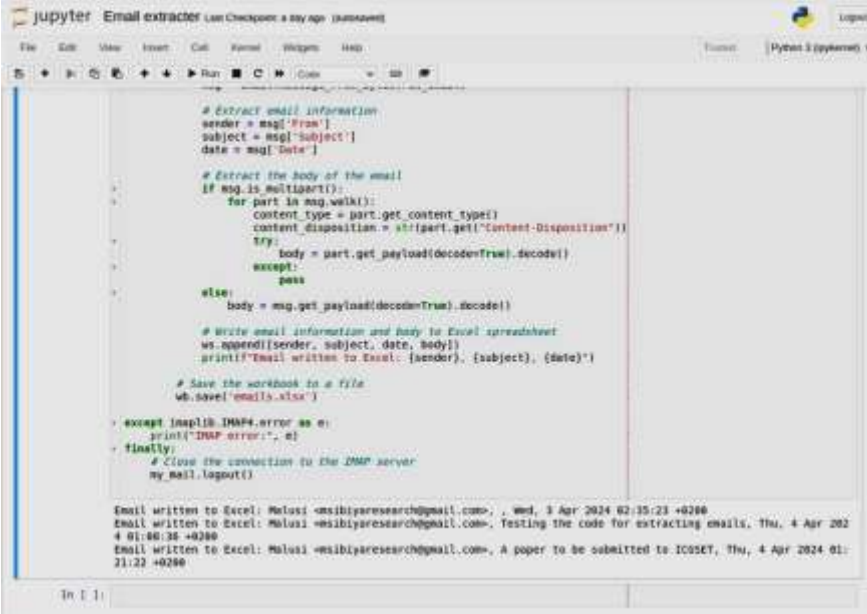
3.3 Experiment 3 – Actual Extraction of Email Messages from the Targeted Email

In Experiment 3 of the research, text extraction was performed using a combination of Python libraries and IMAP (Internet Message Access Protocol) for email retrieval. The

process began with establishing a secure connection to the Gmail IMAP server through SSL (Secure Sockets Layer) using the IMAP4_SSL class from the imaplib library. Once the connection was established, the code logged in to the Gmail account using provided credentials. The Inbox folder was then selected to fetch email messages. Email search was conducted based on specific criteria, including the sender's email address. For each email matching the search criteria, the code iterated through the email messages and extracted relevant information such as sender, subject, and date from the email headers. This information was retrieved using the From, Subject, and Date fields of the email message. Text extraction from the email body was achieved by parsing the raw email data using the email.message_from_bytes() function from the email library. If the email message was multipart (i.e., containing attachments or multiple parts), the code iterated through each part to extract the body content. This was necessary to handle cases where the email body was divided into multiple parts. Finally, the extracted email information including sender, subject, date, and body text was stored in an Excel spreadsheet. This was accomplished using the openpyxl library, which facilitated the creation of an Excel workbook and writing of email data into the spreadsheet. After completing all three experiments, an Excel document containing extracted emails was prepared as a corpus for sentiment analysis. Given that this research is ongoing, the analysis of the data for sentiment analysis will be pursued in a separate research project.

RESULTS AND DISCUSSION

Figure 3 depicts the output cell of the Jupyter code snippet in experiment 3 showcasing the extracted emails.



```

jupyter Email extractor [Last Checkpoint: a day ago] (username)
File Edit View Insert Cell Format Help
Python 3 (ipykernel)

# Extract email information
sender = msg['From']
subject = msg['Subject']
date = msg['Date']

# Extract the body of the email
if msg.is_multipart():
    for part in msg.walk():
        content_type = part.get_content_type()
        content_disposition = str(part.get('Content-Disposition'))
        try:
            body = part.get_payload(decode=True).decode()
        except:
            pass
    else:
        body = msg.get_payload(decode=True).decode()

# Write email information and body to Excel spreadsheet
wb.append([sender, subject, date, body])
print(f'Email written to Excel: {sender}, {subject}, {date}')

# Save the workbook to a file
wb.save('emails.xlsx')

except imaplib.IMAP4.error as e:
    print('IMAP error:', e)
finally:
    # Close the connection to the IMAP server
    my_mail.logout()

Email written to Excel: Malusi-msibiya@researchgmail.com, , Wed, 3 Apr 2024 02:35:23 +0200
Email written to Excel: Malusi-msibiya@researchgmail.com, Testing the code for extracting emails, Thu, 4 Apr 2024 01:00:35 +0200
Email written to Excel: Malusi-msibiya@researchgmail.com, A paper to be submitted to ICOSSEF, Thu, 4 Apr 2024 01:21:22 +0200

```

Figure 3. Experiment 3 Python code snippet showcasing email extraction within a Jupyter Notebook.

After conducting all the 3 experiments, the outcomes of the preprocessing stage conducted on the extracted email data, which was previously saved in an Excel spreadsheet. The preprocessing phase aimed to prepare the email corpus for sentiment analysis, which will be a pivotal component of the subsequent research project. Initially, the extracted email data from the Excel spreadsheet was loaded into Python using the openpyxl library. Subsequently, the

text data underwent preprocessing using either the Natural Language Toolkit (NLTK) or spaCy, prominent libraries for natural language processing (NLP). The preprocessing steps included tokenization (breaking down the text data into individual words or tokens), normalization (converting all text to lowercase to ensure consistency and reduce vocabulary size), removal of stopwords (eliminating common words that do not contribute significant meaning to the text), and lemmatization or stemming (reducing words to their base or root form to further reduce vocabulary size and improve text analysis accuracy). These preprocessing steps were essential for cleaning and transforming the raw email text data into a format suitable for sentiment analysis. By removing noise and irrelevant information, the email corpus was refined to focus on the essential content, thereby enhancing the accuracy of sentiment analysis results. In the subsequent research project, the preprocessed email corpus will serve as the foundation for sentiment analysis tasks, enabling us to gain deeper insights into the sentiment expressed within email communications. This analysis will contribute valuable findings to our ongoing research endeavors.

Acknowledgements

This study was undertaken as a component of research aimed at evaluating the author based on the Thuthuka NRF rating criteria.

REFERENCES

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [2] Jackson, R. (2002). The national student financial aid scheme of South Africa (NSFAS): How and why it works. *Wales Journal of Education*, 11(1).
- [3] Yende, S. J. (2021). Funding opportunities and challenges: A case of South African institutions of higher learning. *Journal of Public Administration*, 56(1), 70-79.
- [4] Riemer, K., Scifleet, P., & Reddig, R. (2012). Powercrowd: Enterprise social networking in professional service work: A case study of Yammer at Deloitte Australia.
- [5] Borge, M., & Goggins, S. (2014). Towards the facilitation of an online community of learners: Assessing the quality of interactions in Yammer. Boulder, CO: International Society of the Learning Sciences.
- [6] Pinto, M. B. (2014). The Use of Yammer in Higher Education: An Exploratory Study. *Journal of Educators Online*, 11(1), n1.
- [7] Munusamy, S., Osman, A., Riaz, S., Ali, S., & Mraiche, F. (2019). The use of Socrative and Yammer online tools to promote interactive learning in pharmacy education. *Currents in Pharmacy Teaching and Learning*, 11(1), 76-80.
- [8] Lytvynova, S. H., & Naboka, O. H. (2021, December). Using the Yammer cloud service to organize project-based learning methods. In *CEUR Workshop Proceedings* (No. 3085, pp. 245-258).
- [9] Radityatama, G. A., Lim, C., & Ipung, H. P. (2018, May). Toward full enterprise software support on nDPI. In *2018 6th International Conference on Information and Communication Technology (ICoICT)* (pp. 1-6). IEEE.
- [10] Oduor, M. (2013). Software architectures for social influence: analysis of Facebook, Twitter, Yammer and FourSquare (Master's thesis, M. Oduor).

Authors

Malusi Sibiya

Malusi Sibiya holds a Ph.D. in Science, Engineering, and Technology, specializing in Machine Learning and Natural Language Processing. He pursued his undergraduate studies in Electronics Engineering and earned a National Diploma from the Central University of Technology, Free State. In 2015, he obtained his BTECH in Electrical Engineering with a specialization in control engineering from the University of South Africa. His passion for programming, self-taught, led him to focus on computer science-related research during both his master's and Ph.D. studies. Dr. Malusi Sibiya has accumulated 19 years of experience in academia, with 14 years spent at the college level as a lecturer of programming and electronics subjects, and 5 years at the university level teaching electronics and computer science.