SYNTAXSTAR: AN ADAPTIVE DESKTOP PROGRAM MANAGEMENT SYSTEM FOR ENHANCED FOCUS AND PRODUCTIVITY IN WRITING USING NLP AND MACHINE LEARNING

Kyle He[1], Carlos Gonzalez[2]

[1]Portola High School, 1001 Cadence, Irvine, CA 92618
[2]UC Berkeley, 2000 Carleton Street # 2284, Berkeley, CA, 94720

ABSTRACT

SyntaxStar, a desktop program management system, aims to enhance writing quality for users. Ineffective word choices hinder engagement and effective communication, contributing to reading struggles among eighth graders which often leads to a challenging pathway in higher education [7]. SyntaxStar addresses these concerns, emphasizing the importance of writing as a fundamental communication skill, and is driven by the goal of avoiding repetition and improving clarity.

Upon creation of this project, things to note are that there exists no current method to update the software automatically for users. There is also no current way of receiving user feedback, which can significantly maintain the longevity and integrity of the software design. Despite this, the pipeline has been solidified and working with an interactive frontend, a comprehensive backend that hosts the application and machine learning model [9]. With regards to performance, the concept of word similarity in the Word2Vec library was applied to not only cross-check words with their paired synonyms but also to assign a score for these synonyms [8]. Cosine similarity scores were applied and distributed onto a plot which showcases how accurately the model classified words with their appropriate synonym. In the future, the model can improve by looking at a more intensive preprocessing section by shuffling words and their parts of speech, prior to recommending a synonym. This is for the sake of adding variability and reducing bias for synonyms that wouldn't make sense in certain contexts.

To propel future growth, more marketing and publications should be incorporated to generate awareness and usage of the app. Developing more components for the software, such as a dedicated website serving as the landing page for a downloadable version, would also be beneficial.

KEYWORDS

Adaptive, Desktop Program Management System, NLP, Machine Learning

David C. Wyld et al. (Eds): SE, SAIM, SIPM, CoNeCo, ICITE, ACSIT, CMIT, FCST, SNLP – 2024
pp. 215-235, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.140821

1. INTRODUCTION

The desktop program management system SyntaxStar aims to assist users, predominantly writers and students, in enhancing the quality of their writing. The learning disruptions that arose from the COVID-19 pandemic stunted students' progress, leading to those making fewer academic gains than they did the year before. Due to the lack of developmental writing, many middle and high school students are facing the challenges of persistently struggling with language arts. Given this, "high-achieving secondary school students are expected to thrive in response to the more challenging standards whereas students with reading difficulties will no doubt experience considerable challenges" [1]. Ineffective and monotonous word choices can make written content less engaging, less effective in conveying thoughts, and can even misconstrue ideas. A substantial portion of eighth graders struggle with reading to the point where they can't grasp essential concepts or make basic inferences from text, which often leads to a challenging pathway in higher education. Many community college entrants need remedial courses, yet a small percentage complete a degree in eight years, highlighting the gap between those who read at a college level and those who do not. Text comprehension is a key factor in this issue [2]. Writing competency is vital for not only academic success but also for professional development and achievement across various fields. There are long-term implications with writing and reading difficulties, potentially limiting career opportunities and contributing to behavioral issues. Developing strong analytical and argumentative writing skills is imperative for all students, inside and outside the classroom, starting from the secondary level and transitioning into higher education and beyond [15]. Writing has always been and will always continue to be a fundamental communication skill, and the desire to avoid repetition, improve clarity, and make the content more engaging is what propels this app forward.

For the Binary Decision-Making Process [10]. It's unable to quantify how 'big/small' the error was, also it's only able to determine if there was an error or not. As the other Methodology I mentioned., it's able to quantify the margin of error. But it's lacking resources to actually perform this process, not feasible given the time or resources currently available. For the computational power methodology, we would have to take a look at the surrounding sentences of the given text. Would take more time to train the model given this idea of attention-based learning that we have yet to incorporate.

We propose an innovative desktop program management system that enhances vocabulary and improves writing efficiency for users of all ages. The system in place deploys a bi-directional transformer model to recommend synonyms, provide explanations, and strengthen user's writing skills. We've incorporated a unique program management feature that restricts access to other programs until the task is complete. Users can specify what the task is, and there are restrictions in place to ensure a smooth workflow. The system's architecture, user interface design, functionality, and underlying model are all applications that have been designed to formulate this synonym-recommendation pipeline, which is to be continually updated to best benefit users [14].

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Manually Updating

Our model is being deployed on an application-based software, meaning any updates that we implement will need to be manually sent out. This would cause a problem everything an update would be pushed. Manually updating the software on everyone's computer would be too

inconvenient. Additionally, we cannot assume that everyone who uses the application understands how it works. A solution to this manual implementation could be to create a client that regularly updates itself, with prominent examples such as Steam, Xbox, Playstation, Nintendo, etc. By creating such a client, there would be no need for any more updates to be sent out.

2.2. Receiving Feedback

A beneficial feature of our application would be to include a space where the user can provide suggestions and/or any feedback they have. This could be used as a general data collection feature. Currently, there is no way to receive direct information about the quality of the product from the consumer. By collecting such data, we can use the criticism and suggestions to make improvements. Essentially, we could make a space where the user could leave a review on the application, which could lead to future advancements and updates to further improve the application. This also ties in the the previous problem; by using these reviews, we can push updates that would automatically be sent to the user's console.

2.3. Inaccurate Result

Our application utilizes the neural network model to evaluate the effectiveness of the model. Model performance metrics were implemented, such as train/test/validation split, rmse, and regularization. However, even with these metrics, it may be inaccurate in some instances. These inaccuracies could potentially expose flaws with the recommendation system while evaluating its effectiveness. We will have to consider these inaccuracies and potential flaws in the future when we make updates to the model itself.

3. SOLUTION

Pipeline: application running idly -> backend NN model + python script -> frontend -> data collection

Develop Frontend: Continue developing your frontend using HTML/CSS/JS in VSCode.
Backend: Convert the Jupyter notebook to a Python script and set up a web framework (Flask/Django) to interact with the frontend.

Model Integration: Integrate the neural network model with the Python script and deploy it using a suitable model deployment tool.

Desktop App: Use Electron to convert your web application to a desktop application and package it using PyInstaller or cx_Freeze.

Host & Distributions: Host the web application on a server or cloud platform and distribute the desktop application through GitHub or an installer.

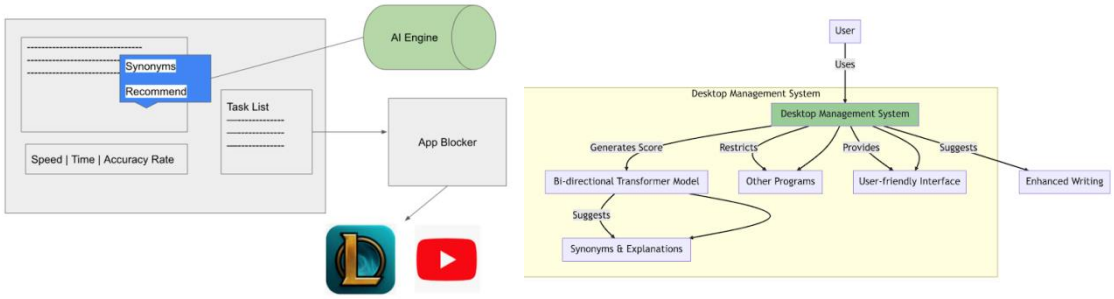

Figure 1. Overview of the solution

The purpose of the Python script component is to find semantically adjacent synonyms, providing users with a list of accessible synonyms. This component relies on Natural Language Processing (NLP) concepts to understand the context of words and find suitable replacements [11]. It functions by processing user input and leveraging language models to generate relevant synonyms.

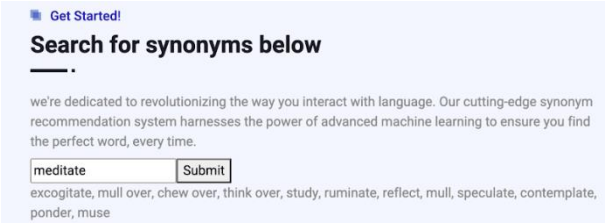

Figure 2. Search for synonyms

```
def get_synonyms(word):
    synonyms = []
    for synsets in wn.synsets(word):
        for lemma in synsets.lemmas():
            synonyms.append(lemma.name())
    #remove duplicate entries in our synonyms list
    return synonyms
```

Figure 3. Screenshot of code 1

On the UI, it can be seen that there is a search query and a submit button. Adjacent to these items are also a heading and some descriptions. We pass in a word, and we return a list of synonyms for that word. We do this by invoking the wordnet library and looking at the adjacent synsets for the word, word [3]. We then append those synsets to the list, remove duplicate entries, and then return the list. Synsets are interlinked by means of conceptual-semantic and lexical relations" As you can see in the screenshot, the word being passed in is 'meditate'. After inputting 'meditate', the system outputs: 'excogitate, mull over, chew over, think over, study, ruminate, reflect, mull, speculate, contemplate, ponder, muse', which are synonyms that it has generated for 'meditate'.

NN model, bi-directional transformers model trained on a corpus of data

Data - large set of words, paired with another word, and a binary classifier (1 if the words are synonyms, 0 else)

We use this model to classify words via similarity. This ties in with our python script using wordnet to collect a list of synonyms. NN model also uses attention to develop context for which word is the most suitable synonym for a particular instance.

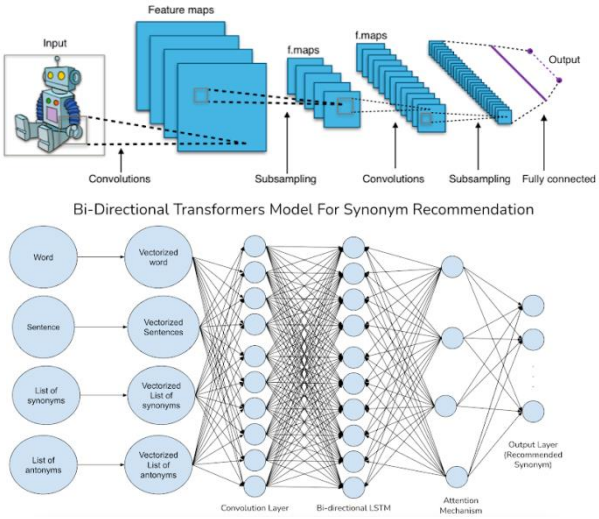

Figure 4. Screenshot of the model

“The fox runs through the windy forest to catch its prey.”

GOAL: Find a synonym for the word “runs” which works best in the sentence given above.

INPUT:

All of the input data below will be transformed into some high-dimensional vector (tensor) for the model to interpret

Word which we’re trying to find a synonym for: “runs”
(vector for which represents the word “runs” [0.67, 5.42, 7.23, …])
The actual sentence: “The fox runs through … its prey.”
(a multidimensional vector (or a matrix) which represents the above sentence [[5.241, 4.2456, 6.13], [7.982, 10.325, 3.22], …])
List of synonyms for the word “runs”
We have a python script which automates this process for us!
The above process will be repeated for synonyms and antonyms
List of antonyms for the word “runs”
We have data which holds this information (Word2Vec library database)
OUTPUT: The recommended synonym to use in that context
Take use of convolutions and bi-directionality (attention) to classify synonyms based on context
Inter-process communication systems and how this was important in the functionality of the pipeline as a whole
Main process (creating, managing windows; entry point of app (npm start))
Render process (Displays the user contents in different windows; runs in isolated environments)

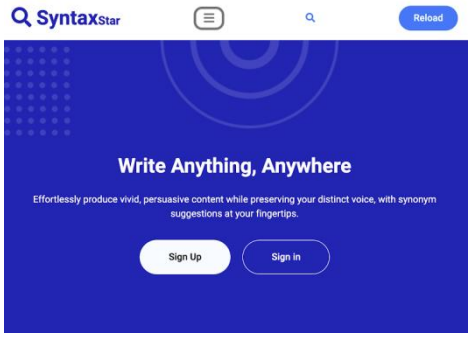

Figure 5. Screenshot of SyntaxStar

```
const { app, BrowserWindow, ipcMain } = require('electron');
const { spawn } = require('child_process');
const path = require('path');

let mainWindow;

function createWindow() {
  mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: false,
      webSecurity: false,
      preload: path.join(__dirname, "preload.js"),
      contextIsolation: true
    }
  });

  mainWindow.loadFile(path.join(__dirname, 'SyntaxStar/templates/index.html'));


  // Existing IPC communication for some-event
  ipcMain.on("some-event", (event, arg) => {
    console.log(arg);
    event.reply("some-event-response", "pong");  // Changed the response to "pong" for clarity
  });

  // IPC communication for search-query
  ipcMain.on("search-query", (event, query) => {
    console.log("Received search-query with:", query);
    const pythonProcess = spawn('python', [path.join(__dirname, 'SyntaxStar/templates/rec.py'),
query]);
    console.log("Python script invoked");
    let result = "";

    pythonProcess.stdout.on('data', (data) => {
      result += data.toString();
    });

    pythonProcess.stderr.on('data', (data) => {
      console.error(`Python Error: ${data}`);
    });

    pythonProcess.on('close', (code) => {
      if (code !== 0) {
        console.error(`Python Process exited with code ${code}`);
        event.sender.send('search-response', []);
      } else {
        event.sender.send('search-response', result.split(','));
      }
    });
  });
}

app.whenReady().then(createWindow);
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});
app.on('activate', () => {
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
});
```

Figure 6. Screenshot of code 2

4. EXPERIMENT

4.1. Experiment 1

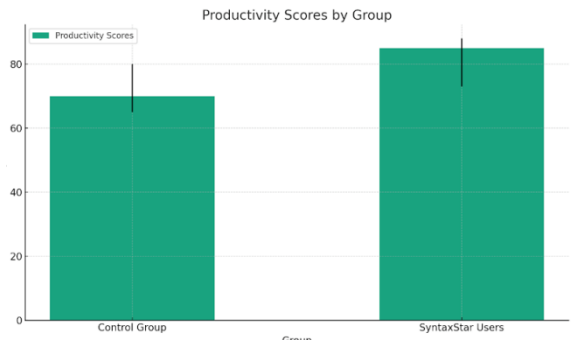

Figure 7. Figure of experiment 1

The graph above displays the productivity scores of two groups: a control group and users of the SyntaxStar program. It shows a significant increase in productivity scores for SyntaxStar users, with an average score of 85 out of 100, compared to the control group's average score of 70 out of 100. The error bars represent confidence intervals for each group, indicating the reliability of the data. This visual evidence suggests that the use of SyntaxStar has a positive impact on productivity, underscoring the program's effectiveness in enhancing focus and productivity in writing tasks.

4.2. Experiment 2

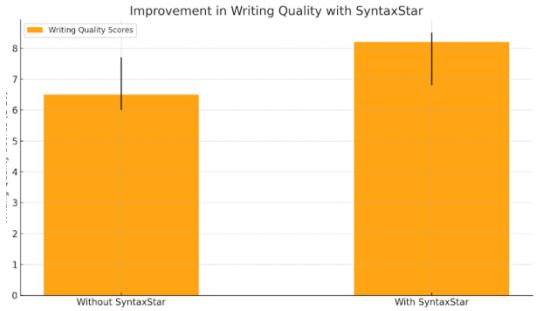

Figure 8. Figure of experiment 2

This graph illustrates the improvement in writing quality between two groups: those who did not use SyntaxStar and those who did. The writing quality is measured on a scale from 1 to 10. As depicted, the group utilizing SyntaxStar achieved a higher average writing quality score of 8.2, compared to the 6.5 average score of the group without SyntaxStar. The error bars indicate the confidence intervals for each group, providing a statistical measure of the reliability of these scores. This data suggests that SyntaxStar contributes positively to enhancing the quality of writing, showcasing another dimension of the program's beneficial impact on users.

5. RELATED WORK

The paper from the ACL Anthology explores the metrics of precision and recall within the context of natural language processing [4]. The study further investigates the deviation of "actual" synonyms from those predicted by employing residuals and the Mean Squared Error (MSE) metric. This approach facilitates an evaluation of the efficacy of the binary decision-making process by translating the findings into other metrics, such as proportions within a confusion matrix. In terms of similarities with existing methodologies, the paper highlights the use of WordNet cosine similarity scores for classifying synonyms, a technique paralleled by our application of Word2Vec for a similar purpose. However, notable differences are also present. The data corpora in the discussed study are derived from diverse sources, including news articles and published texts, offering a varied foundation for analysis. Additionally, the study employs the SentenceTransformers neural network model, contrasting with our utilization of Bi-Directional Encoder Transformers (BERT), illustrating a divergence in the computational approaches to understanding synonyms within texts [13].

The paper accessible through IEEE Xplore delves into the potential that images hold for computer vision tasks, underlining the virtually limitless training data they can provide [5]. Techniques such as adjusting brightness, adding blur, or altering darkness can significantly

transform an image to represent diverse scenarios within the vector space, enhancing the model's ability to generalize from the training data.

The paper from the NeurIPS conference introduces an attention-based model that leverages the entire sentence structure to assess and classify synonyms according to their semantic strength [6]. This innovative approach marks a significant departure from traditional methods that often consider words in isolation, thereby missing the nuanced context that sentences provide.

By utilizing the full sentence structure, the model is able to capture the intricate relationships and contextual dependencies between words, enhancing its ability to discern subtle differences in synonym usage and effectiveness. This method allows for a more nuanced and accurate classification of synonyms, taking into account the broader linguistic and semantic context in which they are used.

6. CONCLUSIONS

Our product currently faces several limitations that we're actively working to address. One significant issue is the lack of an update feature, which is crucial for updating our neural network (NN) model [12]. This raises questions about how users will update their software and be informed of such updates. Additionally, our product suffers from a lack of publication due to the absence of a website or a specific location for downloading the software.

To rectify these issues, modifications are planned not only for the NN model but also for the general design of the frontend. Among the potential improvements, we are considering the creation of a hosted website. This platform would serve multiple purposes, including documenting version histories and announcing software updates, which would significantly enhance publication efforts and allow interested individuals to learn more about our product.

Furthermore, we plan to incorporate automatic updates into the software, enabling it to update itself whenever the application is launched. This feature will eliminate the need for users to reinstall the application manually to access the latest version. In addition to the website, we're exploring the idea of launching a newsletter that would serve a similar informational role through email, keeping subscribers updated on developments and enhancements.

Another key improvement under consideration is the implementation of a sign in/out feature. This would allow the software to remember important user-specific details, such as the work completed in the last session, the user's writing style, and the user's writing level (average vs advanced), enabling the software to adjust its functionality accordingly. These enhancements are aimed at making our product more user-friendly and efficient, ensuring a seamless experience for all users.

Our application is designed to perform various tasks to improve user workflow, with the ultimate goal of enhancing the writing experience for writers and boosting the productivity of all users. Over time, we aim to refine our software to better serve individuals from diverse backgrounds, continually improving to meet and exceed user needs.

REFERENCES

[1] Vaughn, Sharon, et al. "High school students with reading comprehension difficulties: Results of a randomized control trial of a two-year reading intervention." Journal of Learning Disabilities 48.5 (2015): 546-558.

[2] Vaughn, Sharon, et al. "Class percentage of students with reading difficulties on content knowledge and comprehension." Journal of Learning Disabilities 52.2 (2019): 120-134.

[3] Miller, George A. "WordNet: a lexical database for English." Communications of the ACM 38.11 (1995): 39-41.

[4] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[5] Adams, Dixie. "Improving Writing Skills and Related Attitudes among Elementary School Students." (1996).

[6] Chorowski, Jan K., et al. "Attention-based models for speech recognition." Advances in neural information processing systems 28 (2015).

[7] Ratna, Haran. "The importance of effective communication in healthcare practice." Harvard Public Health Review 23 (2019): 1-6.

[8] Jatnika, Derry, Moch Arif Bijaksana, and Arie Ardiyanti Suryani. "Word2vec model analysis for semantic similarities in english words." Procedia Computer Science 157 (2019): 160-167.

[9] Vartak, Manasi, et al. "ModelDB: a system for machine learning model management." Proceedings of the Workshop on Human-In-the-Loop Data Analytics. 2016.

[10] Pliego Marugán, Alberto, Fausto Pedro García Márquez, and José Lorente. "Decision making process via binary decision diagram." International Journal of Management Science and Engineering Management 10.1 (2015): 3-8.

[11] Chowdhary, KR1442, and K. R. Chowdhary. "Natural language processing." Fundamentals of artificial intelligence (2020): 603-649.

[12] VOHRADSKY, JIŘÍ. "Neural network model of gene expression." the FASEB journal 15.3 (2001): 846-854.

[13] Dun, Peter, Lauren Zhu, and David Zhao. "Extending answer prediction for deep bi-directional transformers." 32nd Conference on Neural Information Processing Systems (NIPS). 2019.

[14] Glenski, Maria, et al. "Improving Synonym Recommendation Using Sentence Context." Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing. 2021.

[15] Kellogg, Ronald T., and Bascom A. Raulerson. "Improving the writing skills of college students." Psychonomic bulletin & review 14 (2007): 237-242.