بسم الله الرحمن الرحيم

BIRZEIT UNIVERSITY

Course Project - Arabic and English Spell Checker -

Group : Mousa Salha 1213215
Ali Faqy 1200048
Luai Hawi1200203
Instructor : Dr. Adnan Yahya

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENCS 5342 | INFORMATION RETRIEVAL WITH APPLICATIONS OF NLP

Abstract - This project presents a comprehensive spell checker designed to handle both Arabic and English languages, addressing the unique linguistic challenges inherent to each. The spell checker employs a combination of Natural Language Processing (NLP) techniques, including tokenization, and machine learning algorithms, to accurately identify and correct spelling errors. For Arabic, the system considers the complexities of diacritics, and context-sensitive spelling variations. In English, it addresses common issues such as typographical errors, and contextual word usage.The better data sets we use, the more perfect correction and checking we get. This tool aims to enhance written communication by providing reliable and contextually aware spelling corrections, making it valuable for educational, professional, and personal use.

Discuss - English Spell Checker
We used a machine learning technique in the creation of our English spell checker to guarantee excellent accuracy and efficiency in identifying and fixing spelling mistakes. For spelling correction jobs, we used the **Spello** library, a well acclaimed tool in the NLP field. **Our system's primary function is to train a machine learning model on a large dataset of correct English sentence.**

The model's training phase, which started off taking about 49 seconds, was an important stage. In this stage, the model picks up on the subtleties and patterns of English spelling, which helps it recognize mistakes and provide suitable fixes. Considering the computational complexity of this procedure, we decided to store the finished trained model to disk. This tactical choice greatly improves system performance by removing the requirement for retraining on subsequent uses, **Improved runtime speed by loading the previous trained model.**

The pre-trained model is loaded from disk for every session, allowing us to achieve more fast spell checking. This method preserves computing resources and enhances the user experience by offering quick response, making the system more useful for practical uses.

Our project's inclusion of the **Spello** library was crucial to obtaining these results. It offers strong techniques for mistake detection, tokenization, and recommendation for correction—all necessary components of a trustworthy spell checker. Additionally, our spell checker can adjust to different situations and usage patterns using a data-driven machine learning model, providing accurate and contextually appropriate corrections.

Overall, our English spell checker project's technology advancements and design decisions show a balanced commitment to efficiency and accuracy. We have developed a technology that works well in situations and is scalable and responsive in real-world circumstances by utilizing machine learning and pre-trained model , improve the dataset and make it more strong plays an important role in enhancing performance for spell checker.

examples of the data set used -

The man sprang from his chair and paced up and down the room . . . . . .

No, no, there's life in him!" shouted another. "But . . . . . .

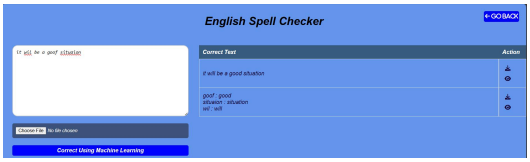
The dataset we used contain news , e books , stories , etc…

Testing our model -

When we tested our model, we wrote a sentence that contained some spelling errors, and the model corrected those errors and identified them for us.

For example our input :
model correction : ***it will be a good situation***
corrections : ***'wil': 'will', 'goof': 'good', 'situaion': 'situation'***

As we see in this screen :



Discuss - Arabic Spell Checker
We encountered particular difficulties in creating an Arabic spell checker and corrector because there aren't many easily accessible machine learning packages designed for Arabic spell checking. We devised a distance-based strategy to close this gap, determining the **Levenshtein distance** between the input word and words in an extensive Arabic corpus.

A popular metric for comparing two sequences is the Levenshtein distance, which counts the least number of single-character alterations **(insertions, deletions, or substitutions)** needed to change one word into another. Using this metric, our spell checker finds misspelled words and recommends repairs based on how close they are to valid terms in the corpus.

The steps of our spell checker is as follows -

Corpus Compilation : Initially, we assembled a comprehensive collection of accurately spelled Arabic terms that covered a broad spectrum of vocabulary and context.

Input Analysis: The system determines whether a word entered by the user occurs in the corpus. If the word is located, it is considered to be correctly spelled and no additional steps are needed.

Distance Calculation: The system determines the Levenshtein distance between each input word and every word in the corpus if the input word is not present in the corpus.

Suggestion Generation: Next, words with the smallest distance to the input word are identified by the system. Any word with a distance of one is instantly returned with the recommended adjustment. The system moves on to words with a distance of 2, offering them as potential recommendations in the event that no term has a distance of 1.

Although there are no dedicated machine learning libraries available for Arabic spell checking, our method exhibits good accuracy and consistency. It efficiently finds and fixes spelling mistakes, improving Arabic content that has been written in a readable and accurate manner.

Our model is effective and useful for real-time applications since it eliminates the need for large amounts of training data and intricate algorithmic models by concentrating on distance metrics. This method bridges the existing technological gap and sets the stage for future improvements, including the possible integration of more sophisticated natural language processing (NLP) techniques as they become available for Arabic.

To sum up, our spell checker and corrector that operates using distance provides a reliable and effective way to correct spelling in Arabic. It makes precise recommendations and raises the standard of written Arabic communication. but the complex orthographic variants of Arabic make it difficult to choose the most corrected word. The inclusion of prefixes, suffixes, and infixes can significantly alter Arabic words, Which makes our model work more difficult to determine the appropriate word, so we must develop our system to become more efficient .

examples of the data set used -
The dataset we used for this algorithm contain

مصطلحات عربية مكتوبة بشكل صحيح :
المثال
والفلسطينية
المنزلقة
مميزا
باحثوها
بتثقيف
وتستشف
والمجازر
بهجة
إنهائه
لايتجزأ

Testing our program -

Discuss - Arabic Spell Checker using bi-grams

We incorporated a bi-gram language model into our system to improve the precision of our spell checker and corrector, especially in situations where more than one option might be appropriate. By taking into account the surrounding words in the text, including the prefix and suffix, this bi-gram model assists in choosing the term that is most appropriate for the context.

In order to ascertain the likelihood of a particular word coming after another, bi-gram models examine pairs of consecutive words in a corpus. By utilizing the linguistic patterns that are present in the language, this probabilistic strategy offers a potent way to forecast and correct words that are sensitive to context.

implement and utilize steps of the bi-gram model in our spell checker -

Corpus Preparation: We build our bi-gram model using a sizable and varied corpus of Arabic text. To guarantee that the model captures a broad spectrum of linguistic patterns, this corpus contains a variety of genres and sentences .

Bi-gram Model build : We identified bi-grams pairs of consecutive words—from the corpus and determined how frequently each bi-gram occurred. Afterwards, the chance of a word coming after a specific previous word was calculated using this frequency data.

Error Detection: Using the previously mentioned distance-based mechanism, the system first verifies if a word entered by the user is spelled correctly. The bi-gram model is used to hone the correction recommendations in cases when the term is either not present in the corpus or is found to be wrong.

Contextual Suggestion Generation: The technique determines possible repairs for a misspelled word by calculating the minimum Levenshtein distance. The context around the misspelled word is next examined, including the terms that come before (prefix) and after (suffix). The bi-gram model looks at the bi-grams created with the prefix and suffix words to determine the likelihood of each adjustment fitting in this context.

Selection of the Most Probable Word: The bi-gram probabilities of the proposed corrections are used by the algorithm to rank them. The term that is most likely to be true is the one that has the highest total probability when both prefix and suffix are taken into account. In the absence of a suffix, the model exclusively utilizes the prefix bi-gram.

The spell checker's capacity to produce contextually relevant corrections is greatly improved by this bi-gram-based method, which also lowers the possibility of mistakes that may occur from selecting suggestions just on the basis of distance measures. The bi-gram model makes sure that the selected word not only fixes the spelling but also blends in nicely with the rest of the text by taking into account the words that surround it.

Advantages:

Context Awareness: The bi-gram model provides context sensitivity, ensuring corrections are grammatically and semantically appropriate.
Improved Accuracy: By using probabilities derived from real text, the model can disambiguate between similarly spelled words based on actual usage patterns.
Scalability: The model can be easily updated with new bi-grams as more text data becomes available, continually improving its performance.

Challenges:

Data Requirements: A large and diverse corpus is necessary to train an effective bi-gram model, which can be resource-intensive to compile and process.
Computational Overhead: Calculating bi-gram probabilities adds computational complexity, which needs to be managed to maintain real-time performance.
take more time to search and correct .

All things considered, our spell checker's integration of a bi-gram model makes use of contextual data to provide a comprehensive solution that improves the system's accuracy and usability for Arabic text correction. This strategy marks a substantial advancement in the creation of reliable NLP tools for the Arabic language.

examples of the data set used -
The dataset we used for this algorithm contain

بإجراء تسمح المادة من الفقرة أن المحاكم إحدى ترى <s>
حتى للطرفين الذاتية النية حول تحقيق
في المشروعة غير العنف أعمال بقمع المتعلق البروتوكول <s>
المكمل الدولي المدني الطيران تخدم التي المطارات

Testing our program -

قراءة الكتب مفيد لكل شخش
تصحيح التطبيق :

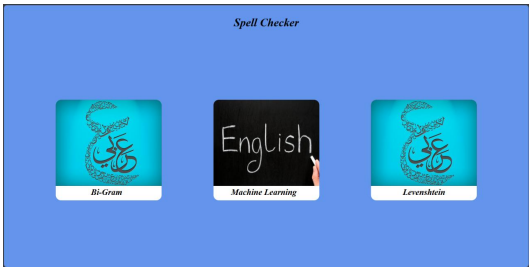
The dataset we used for this algorithms :

1. machine learning in English :
About 120000 lines of sentences

2.Levenshtein distance in Arabic :
About 170000 unique words

3. Bi-grams algorithm in Arabic :
500000 lines of sentences

Our user interface is web page ,
That user can choose which algorithm of this three algorithms want to use and Arabic or English spell checker .
then take the user input text or input text file ,
And check it the correct and return it to user in simple way to show his errors and corrections .



Weaknesses:

Data Intensive: Requires a large and diverse corpus to accurately model bi-gram probabilities.

Computational Complexity: More resource-intensive due to the need for calculating bi-gram probabilities.

Initial Setup: Compiling and processing the necessary corpus can be time-consuming.

Summary:

Machine Learning-Based Model: Best for high-accuracy, efficient English text correction with a large training dataset.

Distance-Based Approach: A simple, language-agnostic method suitable for applications with limited resources or specialized language needs like Arabic.

Bi-Gram Model: Provides the most contextually accurate corrections, suitable for advanced applications where maintaining the linguistic context is crucial.

Each algorithm has distinct advantages and limitations, and the choice of method should be guided by the specific requirements of the application, available resources, and the target language. Combining these approaches where appropriate could also yield a more robust and versatile spell-checking system.

We attached the code and you can try it .