Phonetic Based Fuzzy Matching of Quranic Verse in Distributed Environment

Hassan Rehman
L217302@lhr.nu.edu.pk

FAST, NUCES, Lahore, Pakistan

June 5, 2022

Abstract

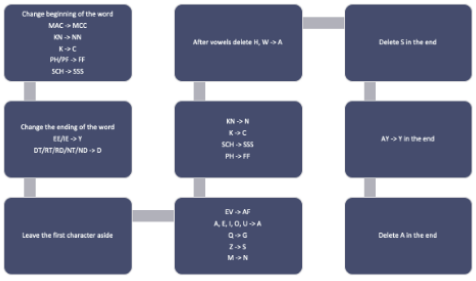
Searching related document is an easy task when it comes to the same language. But when it comes to roman language its very hard to search as in the roman every person has its own style of writing. The spelling vary from person to person and from language to language. For example if someone is writing Urdu in roman it would be different from someone writing Arabic in roman. The searching problem arise specially in Quran, the traditional way of searching the data from Quran is to give exact verse and chapter number. We have tackled this problem by using the transliteration data use that data via phonetic algorithm to achieve this problem. The data has been taken from the online source. The results we got are promising where the same query with different spellings are given to the system and it returns the same output which is good and that we want to achieve. The proposed technique take 4 to 5 words of the verse(if verse is longer) to search more effeciently.

Keywords
Fuzzy, Phonetics, PySpark, NLP, N-Grams

1 Introduction

There is lots of advancement in the field of data mining and data extraction but these advancements are mostly language specific. There ahs been lots of work done in English, Arabic, Chinese, Hindi etc. Many people are still working on it for the better results. Any one can found large corpus related to these language in which data is also labeled where a person just have to apply a better techniques to get good results. But still work on roman languages is not to that extent as compare to other languages. The problem in roman languages is that the syntax and spellings varies from person to person and also it varies from region to region. For example 2 person chatting on a chat application and talking to each other in roman Urdu the texting style of both individual will be very much different. Similarly when it comes to search the data by querying in roman language its hard to search. The same problem arise in digital Quran. A user have to enter the exact verse and chapter number to get desire result. In this paper we have implemented a technique in which a person can get its desired result by fuzzy string matching technique. For achieving this technique we further used two algorithms one metaphone and other one nysiis. The detail of these algorithm is following:

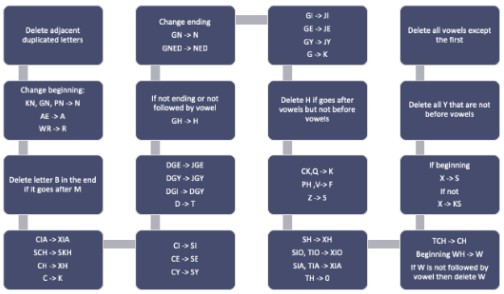
This one was developed in 1970 by the New York State Identification and Intelligence Center (surprise, surprise). The idea is the same as for the Soundex: if there are homophones, you will match them by assigning specific indices for particular sounds. What is good about its results: it is more accurate comparing with Soundex as it returns fewer surnames under the same code.



NYSIIS

Developed by Lawrence Philips in 1990, the Metaphone is also more accurate compared with

the Soundex method as it takes into consideration the groups of letters. The disadvantage shows up when you apply it to reconcile the strings that are not in English, as it is based on the rules of English pronunciation. Following Metaphone, Philips also designed the Double Metaphone. As its name suggests, it returns two codes, so you have more chances to match the items, however, at the same time, it means a higher probability of an error. According to the algorithm, there are three matching levels: a primary key to the primary key = strongest match, a secondary key to the primary key = normal match, the secondary key against the secondary key = weakest match.Sometime later (9 years, to be precise), Philips introduced his gem — Metaphone 3 — the extended matching algorithm, mostly intended to match personal information, such as names and surnames, like its previous versions, however, with a significantly higher degree of precision. What's wonderful, it is adjusted to various language families (endings are hardcoded into the code) which allows it to match most of the European surnames (including, probably, the toughest — the Slavic ones). You can familiarize yourself with the code, it is very detailed, beautiful, and not for free usage. I would even highly recommend checking the code for everyone who has at least some interest in string data matching. By reading it you can follow the author's logic and that may lead you to the most fascinating discoveries, straight to the resolution of your business problem!



Metaphone

We applied these algorithm on the transliteration data that we acquired from the online source and saved the data into spark pickle data format. And for searching query we again load the already saved into the RDD and process it. The results we got are promising we have tried different queries by changing the spellings like writing "V" instead of "W" and even writing a wrong word but the sound that create is same to the actual one, and our system gives the result accordingly. Moreover this system is also faster if we run the code on the same 4 GB RAM of computer the execution time drops drastically.

2 Related Work

As mentioned above the most of the work in data retrieval has been done in the languages like English, Arabic etc but little work has been done on roamn languages. And specially in Arabic little to no work has been done. We have done some of the literature review before actual starting the work and we have found some of the related work.

In paper [1] it is stated that in Natural Language Processing, approximate string matching has a wide range of applications. This study compares and contrasts various approximation string matching techniques. The edit distance between characters in the two strings is used in most algorithms. It also discusses the difficulties of using these algorithms to text retrieval. The authors present a new method for approximate string matching that is more suitable for text retrieval. In this experiment, we use a matrix to compare two strings in order to find commonalities. For each overlap character between two strings, the matrix will be modified. When no overlap character position is encountered, an overlap counter is kept that increments the value for each overlap character position and resets the position to 0. The highest counter value is then used to calculate the degree of similarity using a ratio. Python was used to implement the algorithm. The findings show that the proposed method can be utilised to find lexically comparable words. Lemmatization, text summarization, topic modelling, and data mining solutions will all benefit from this technique.

In [2] it is stated that in business intelligence, data purification is critical. We offer a new phonetic technique for string matching in Russian without converting Cyrillic to Latin characters. It is based on the Russian language's rules for sound creation. Furthermore, we propose an expanded technique for matching Cyrillic strings in which phonetic code letters are supplied as primes, with the code of a string being the sum of these numbers. The results of our experiments reveal that our algorithms are capable of accurately identifying phonetically similar sequences in Russian.

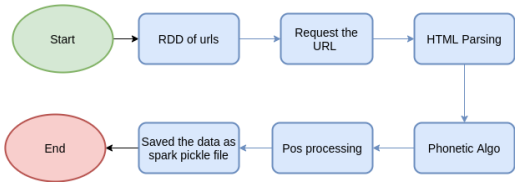
3 Problem statement and Methodology

Most of the digital quran application have no searching ability. The only searching ability they have is to give verse number and chapter number. But most of the time a person just remember few words and he wants to see where

that particular verse resides in Quran. This will make things a lot easy when it comes to searching verse and other information related to it. We have implemented the methodology in pyspark. And our methodology have 2 parts. The first part shows how we acquired the dataset and the second part how we query it

3.1 Data Crawling

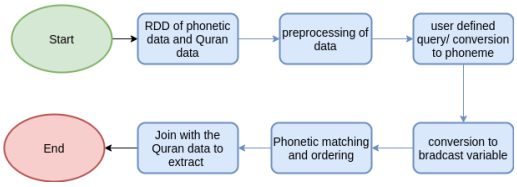
So in this part we first make a list of urls for 114 chapters of quran. After that we have initialized the RDD and crawl the web and extract the verses of Quran and apply post processing on it. The post processing includes removing the invalid spaces, removal of irrelavant words (Our data consist of some words like Juz of and Section of) . So we remove all these words. And finally if the string is greater than 5 words we use the ngram algorithm in which we used 5 grams to chunk the sentence. This will be helpful to get optimum results. And in last we apply phonemes algorithm and store the data in spark pickle format



Data Fetching

3.2 Query of Data

The second part is the most important and the core of this paper. In this we query the results from the Quran Data. Initially we have taken two data set that are in the form spark pickle dataset. One data set is the same data set that we have taken from online source. Other dataset is the actual data consist of "Arabic Verse", " Verse Number"," Translation in Urdu and English" and "Surah Name". This data will be used to represent actual data after retrieval. We will convert both of these data to RDD and load it as pickleFile. Than we initialize the query that we want to search and convert it into phonemes. For better performance we convert both of these phonemes to broadcast variable and than map whole data to the function of match phonemese. The match phonmese function will take the data and bradcast variables and calculate the distance. Here we have used gestalt pattern matching for getting the distance. We only take the distance which has high percentage.



Result Query

3.3 Dataset

The dataset is collected from different source. The phonetic data is collected from the webiste whose url is https://quran411.com/verse-by-verse. The data is fetched by crawling the web pages. The crawler as mentioned above is made using pyspark. The Quran data is acquired from github repository https://github.com/hablullah/data-quran. The preprocessing is done afterward on the dataset.

4 Results

The results we have acquired are promising. We have given a list of queries and see the results which totally matching and also giving the results that are closely related to the query. Also we have tried with different spelling and it also recognize the actual data. Following are some of the results.

4.1 Results searching same data with different ways

Query	Result	Edit Distance
qol hu wallahooo ahaad	قُلْ هُوَ اللَّهُ أَحَدٌ	0.86
kul hoo vellahoo ehed	قُلْ هُوَ اللَّهُ أَحَدٌ	0.87
kul hu wallahu ahad	قُلْ هُوَ اللَّهُ أَحَدٌ	0.92

Result Query

4.2 Other Queries

Query	Result	Edit Distance
wamaliyalaa abudullazi fatarani	مَا لِيَ لَا أَعْبُدُ الَّذِي فَطَرَنِي وَإِلَيْهِ تُرْجَعُونَ	0.88
bismillahir rehmani rahim	إِنَّهُ مِن سُلَيْمَانَ وَإِنَّهُ بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ	0.89
al khabisatoo lilkhabisina walkhabisuna	الْخَبِيثَاتُ لِلْخَبِيثِينَ وَالْخَبِيثُونَ لِلْخَبِيثَاتِ ۖ وَالطَّيِّبَاتُ لِلطَّيِّبِينَ وَالطَّيِّبُونَ لِلطَّيِّبَاتِ ۚ أُولَٰئِكَ مُبَرَّءُونَ مِمَّا يَقُولُونَ ۖ لَهُمْ مَغْفِرَةٌ وَرِزْقٌ كَرِيمٌ	0.91
ma kan muhammahun aba ehadin	مَا كَانَ مُحَمَّدٌ أَبَا أَحَدٍ مِّن رِّجَالِكُمْ وَلَٰكِن رَّسُولَ اللَّهِ وَخَاتَمَ النَّبِيِّينَ ۗ وَكَانَ اللَّهُ بِكُلِّ شَيْءٍ عَلِيمًا	0.88

Result Query

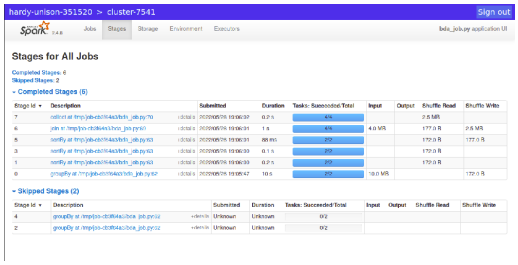
5 DAG Visualization

We use data proc for deployment of code. For clustering we have used the free trial of GCP and we used the following requirement to set cluster:

- Ubuntu 18.05
- 50 GB storage and GCP storage
- 1 Master and 2 Nodes (4GB RAM)
- Anaconda for environment

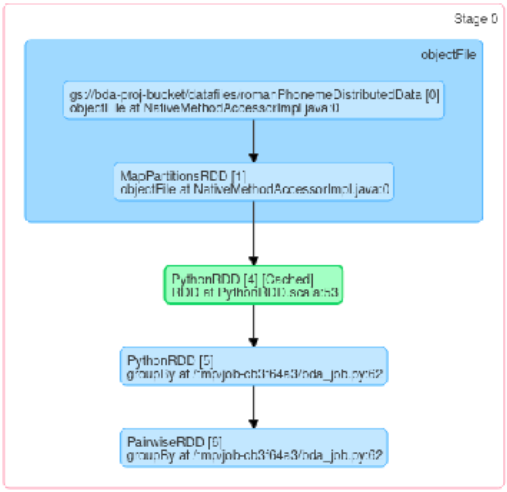
The detail of after job completeion is following:

5.1 Detail Visualization

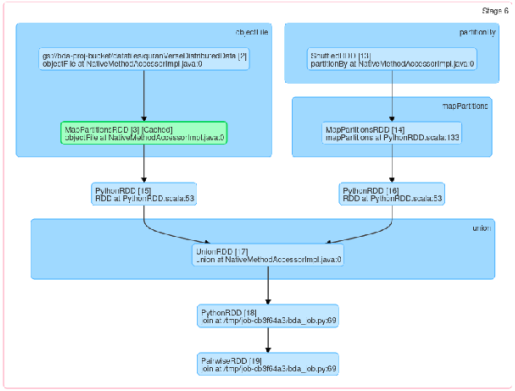


Detail of jobs

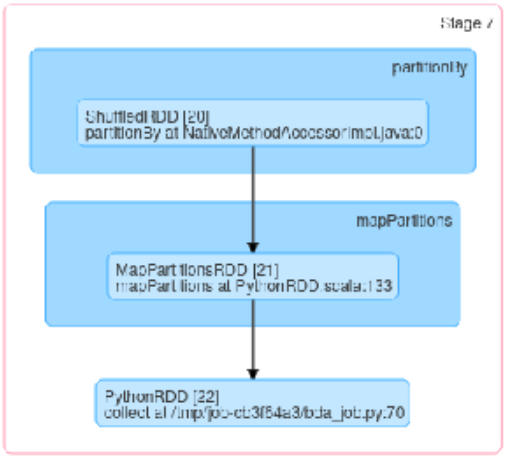
5.2 DAG Visualization



Stage 0 DAG



Stage 6 DAG



Stage 7 DAG

Conclusions

In this study, =we have introduced a technique to query the Quranic verse from transliteration. For achieving this we have used phonetic algorithm and used the edit distance to calculate the percentage of resemblance. The results we got are promising and fetching the data and related data. The computational time is also good as we have done the experiment using 4GB RAM computer and run the code without distribution the time it took was 15 seconds (importing of data exclude). Where as the detail of DAG shows us that time it take to compute the query is about 1.5 seconds(excluding import of data).

References

[1] Krishna Kalyanathaya, Akila D., and Suseendran G. A fuzzy approach to approximate string matching for text retrieval in nlp. *Journal of Computational Information Systems*, 15:26–32, 05 2019.

[2] Viacheslav Paramonov, Alexey Shigarov, Gennagy Ruzhnikov, and Polina Belykh. Polyphon: An algorithm for phonetic string matching in russian language. 10 2016.