**DESIGN AND ANALYSIS OF ALGORITHMS**

Salanatin, Nathaly Pearl F. Professor Jan Eilbert L. Lee

BSCS-NS-2A March 25, 2023

*Trigram Search Algorithm*

**CONTENTS**

1. Introduction …………………………………………………………………………. 1
2. The algorithm ……………………………………………………………………….. 2
3. Pseudocode of the algorithm ………………………………………………………... 2
4. Step by step numerical example of the algorithm ………………………………...… 3
5. Advantage & disadvantage ………………………..………………………………… 4
6. Conclusion …………………………………………………………………………... 5

References …………………………………………………………………………… 5

1. **Introduction**

Trigram Phrase Matching is a technique for discovering phrases that are very likely to be synonyms. The trigram, also known as Trigraph Search, is suitable for text searches when the actual spelling of the target object is unclear. In database technology, it is known as “Trigram indexing”. It identifies objects that match the most three-character strings in the provided search keywords.

The trigram algorithm is a bigram model extension that evaluates the likelihood of the following word simply based on the last word. The trigram model can capture more complicated word relationships and deliver more accurate predictions by integrating the third word.

Trigram search algorithm uses trigrams, which are groups of three adjacent characters or words, to efficiently match and search text. However, trigram index cannot be used for substrings shorter than 3.

Similar to the ‘LIKE’ operator in SQL, the trigram index can be implemented to find items containing a specific substring and regular expression matching. In the case of the LIKE operator, the runtime basically pulls the longest substring from the pattern, eliminating wildcards, and searches for it before performing a regular expression match on all selected objects. The trigram index can also be used for exact match indexing, especially if no other indexes have been specified for the data collection.

1. The Algorithm

The primary idea behind trigram search is to breakdown text into smaller, more digestible chunks that may be compared and searched. The algorithm compares the trigrams of the search query with the trigrams of the target text to detect similarities and matches by producing trigrams from the text.

Trigram search methods are very effective when looking for text with mistakes or misspellings since the algorithm can identify similar trigrams even if the spelling of the words is slightly different.

1. Pseudocode of the algorithm

# Generate the trigrams

for i in range(len(tokens) - 2):

key = (tokens[i], tokens[i+1])

value = tokens[i+2]

if key in trigrams:

trigrams[key].append(value)

else:

trigrams[key] = [value]

# Generate the output text

output = []

start\_index = random.randint(0, len(tokens) - 2)

current = (tokens[start\_index], tokens[start\_index+1])

output.extend(current)

for i in range(n):

if current in trigrams:

possible\_next = trigrams[current]

next\_word = random.choice(possible\_next)

output.append(next\_word)

current = (current[1], next\_word)

else:

break

1. Step by step numerical example of the algorithm

Suppose we have “I love programming. It’s my favorite subject” as an input text.

Step 1: We first tokenize the input text into a list of words.

Example:

[‘I’, ‘love’, ‘programming’, ‘its’, ‘my’, ‘favorite’, ‘subject.’]

Step 2: Generate the trigrams.

Here, we carefully generate a dictionary of trigrams that show a list of possible pairs of third words.

Example:

[‘I’, ‘love’, ‘programming’], [‘its’, ‘my’, ‘favorite’, ‘subject.’],

etc.

Step 3: Generate the output text.

Here, we generate the output text by selecting random pair of thirds words and select one of these third words and add it to the output sequence, then update the current pair of words and repeat the process until the desired number of words is acquired.

Example:

current = (‘I’, ‘love’)

output = [‘I’, ‘love’]

possible\_next = trigrams[current] # [‘its’, ‘my’]

next\_word = random.choice(possible\_next) # ‘its’

output.append(next\_word) # [‘I’, ‘love’, ‘its’]

current = (current[1], next\_word) # (‘love’, 'is'')

and continuously.

1. Advantage & disadvantage

One of the main advantages of the trigram algorithm is its simplicity and ease of implementation. It requires only a relatively small amount of training data, making it an ideal choice for many natural language processing tasks. Additionally, the algorithm can be easily extended to handle n-grams of any size, providing greater flexibility and accuracy in language modeling.

Here are some advantages and disadvantages of the trigram search algorithm:

Advantages:

* Precision: Since it can match incomplete words or characters, trigram search is more efficient than other text search algorithms. As a result, it is handy for searching for misspelled words or incomplete matches.
* Speed: Because it lessens the amount of text that must be searched, trigram search is quicker than other text search algorithms. It only looks for trigrams in the text, which make up a small percentage of the total length of letters or words.
* Scalability: Since trigram search only searches for trigrams, it can handle vast volumes of text. This makes it handy for searching large databases or document collections.

Disadvantages:

* Language-dependent: Trigram search may not perform as effectively in languages with no spaces between words or with complicated scripts that require specific processing.
* Storage requirements: Trigram search needs more storage in order to save the trigrams. With big quantities of text, this can be a huge amount of storage.
* False positives: If the search term contains multiple trigrams that are also present in other words or phrases in the text, trigram search may create false positives. This could result in incorrect search results.

1. Conclusion

In conclusion, the trigram search algorithm is a fast and precise text that uses trigrams to efficiently match and search text. It can handle partial matches and misspellings and is flexible and useful for searching big volumes of text. However, it requires additional storage to retain the trigrams, it may create false positives, and it may not perform as effectively for some languages. Overall, the trigram search algorithm is a valuable text search tool, although its success is dependent on the application and the qualities of the text being searched.

References

* <https://en.wikipedia.org/wiki/Trigram_search>
* <https://lhncbc.nlm.nih.gov/ii/tools/MTI/trigram.html>
* <https://medium.com/@mahmudulhassannavid/trigram-search-using-node-js-postresql-dc15d34328a9>