Can you imagine your life without AUTOCORRECT?

A dive into the world of
SPELLING CORRECTION & PHONETIC CORRECTION

Part1 - Spelling Correction

Goals :

1. Understand the problem of correcting spellings in queries
2. Few interesting ways to solve the problem

Why should one learn this?

1. To correct document(s) being indexed
2. To correct the queries itself to get apt results

How search engines provide spell-correction as a part of UX

- In case of various correct alternatives, choose the nearest one using algorithms. (Hold on the question of how to)

- If two or more correct spellings are tied, select the most common one.

- **What is the most common one?**
  - The correction with the most no. of occurences in the collection (or)
  - The correction which is most common among the user typed queries (Used by most of the search engines on web)

How users are exposed to the functionality of spell correction

1. If query = “carot”, result = all docs. with “carot” along with other spelling corrections like “carrot” and “tarot”.
   a. Only when ‘carot’ is not in the dictionary.
   b. Only when the original query returned fewer than a preset number of documents (say fewer than five documents).
2. When the original query returns fewer than a preset number of documents, the search interface presents a spelling suggestion to the end user.

   Thus, the search engine might respond to the user: “Did you mean carrot ?”

Forms of Spelling correction

- Isolated-term correction
- Context-sensitive correction

Isolated-term correction

- Correct a single query term at a time – even when we have a multiple-term query.

This is the time to answer the "how to"

In what ways can isolated-term correction be performed?

1. Edit distance
2. k-gram overlap

Edit distance

Operations (ops.) :

- Replace a character
- Add a character
- Delete a character

Task :

Convert a string1 to string2 using minimum no. of operations

Example:

string1 = ‘quirky’ string2 = ‘murky’

$$\text{quirky} \xrightarrow[\text{no of ops?}]{} \text{murky}$$

All the colored terms are common in both the strings. Now it breaks down to

$$\text{qi} \xrightarrow[\text{no of ops?}]{} \text{m}$$

The best way is to delete a character and replace the left over character with ‘m’.

No. of ops. = 2 ⇨ ***Edit distance = 2***

[Check out the algorithm for edit distance calculation]()

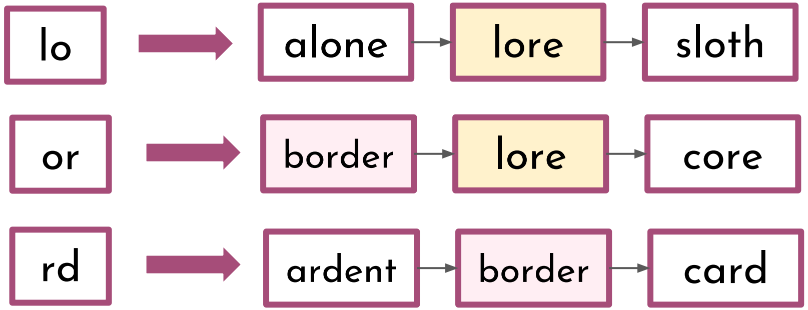
k-gram overlap

- Used to reduce the terms which undergo calculation of edit distance with the query term .

- The dictionary contains all the k-grams of the query term and the posting lists points from a k-gram to terms containing the k-gram.

Example

- **Query** : ‘lord’
- **Task:** To identify words matching 2 of its 3 bigrams
- bigrams of lord = {‘lo’, ‘or’, ‘rd’}



lo → alone → lore → sloth

or → border → lore → core

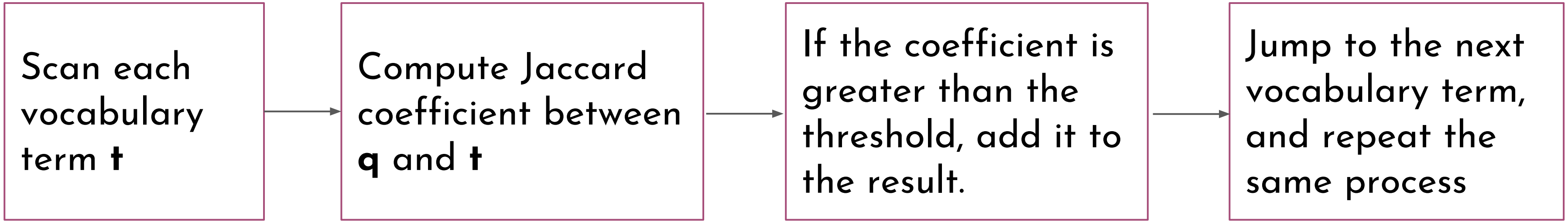
rd → ardent → border → card

- The linear scan intersection can be adapted when the measure of overlap is the Jaccard coefficient.

$$\text{Jaccard coefficient} = \frac{|A \cap B|}{|A \cup B|}$$

Where A and B are two sets

- In our case, the two sets are :
    - Set of k-grams in the query q
    - Set of k-grams in a vocabulary term



Scan each vocabulary term **t**

Compute Jaccard coefficient between **q** and **t**

If the coefficient is greater than the threshold, add it to the result.

Jump to the next vocabulary term, and repeat the same process

Can Isolated-term correction work well in all the cases?

- Consider the case where all the words are correctly spelled, but still there is something wrong.
  Ex : I flew ***form*** New York.

- The answer is **no.**
- To overcome this problem we have **context sensitive spelling correction**.

Context sensitive spelling correction

1. Retrieve dictionary terms close to each query term.
2. Try all possible resulting phrases with one word “fixed” at a time.
    - flew from New York
    - fled form New York
    - flea form New York
3. Suggest the alternative that has lots of hits.(i.e has more number of ccurences in the query logs that contain previous queries by people)

Part2 - Phonetic Correction

- Misspellings that arise because the user types a query that sounds like the target term.

- The main idea here is to generate, for each term, a “phonetic hash”so that similar-sounding terms hash to the same value.

- Algorithms for such phonetic hashing are commonly collectively known as Soundex algorithms.

Scheme of a soundex algorithm

1 Turn every term to be indexed into a 4-character reduced form.

2 Build an inverted index from these reduced forms to the original terms; call this the soundex index.

3 Do the same with query terms.

4 When the query calls for a soundex match, search this soundex index.

Standard Soundex Algorithm

1. Retain the first character

2. Convert each character to digit using the rules in the table.

3. Repeatedly remove one out of each pair of consecutive identical digits.

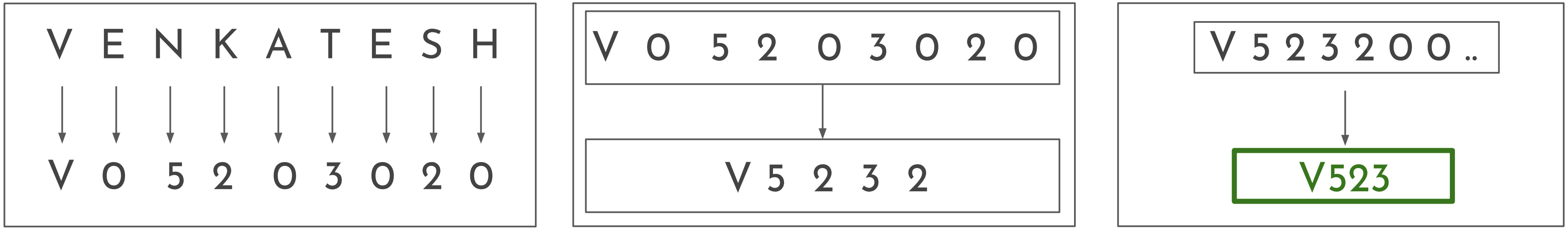
4. Remove all the zeros.

5. Add trailing zeros, and return the first four positions.

Alphabets to be replaced	Digit
A, E, I, O, U, H, W, Y	0
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6

Implementation using an example

Let the term be "Venkatesh".



- Retain the first character
- Convert each character to digit using the rules of the algorithm.

- Repeatedly remove one out of each pair of consecutive identical digits.
- Remove all the zeros.

- Add trailing zeros, and return the first four positions.

The flow

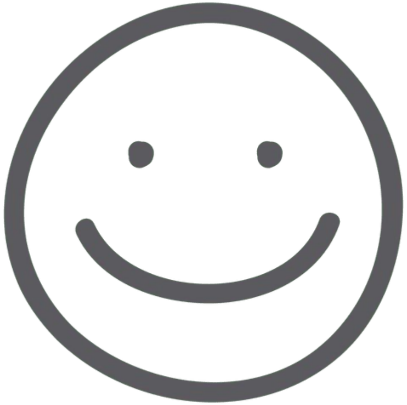
Follow the scheme on page 13, to proceed further after getting the phonetic hash

Test your understanding

- Find two differently spelled proper nouns whose soundex codes are the same.
- Find two phonetically similar proper nouns whose soundex codes are different.

Hope these slides would have made atleast a 0.1% progress in your understanding about

SPELLING CORRECTION & PHONETIC CORRECTION



- By Vinoothna Sai Kinnera