Make it Clean

2023 Information Retrieval Project

Michele Andreata
michele.andreata@studenti.unimi.it

Abstract

This project delves into the problem of spelling and segmentation error correction for text processing in information retrieval systems. Through an exploration of spell-checking and segmentation correction methodologies, including Peter Norvig's probabilistic approaches and SymSpell's techniques, insights are gained into the complexities and limitations of these linguistic challenges. Key findings include the importance of optimizing frequency dictionaries and leveraging advanced methods such as SymSpell for enhanced performance. Furthermore, the project highlights the challenges in joint spelling and segmentation error correction and suggests avenues for future research, including the exploration of more sophisticated approaches such as recurrent neural networks (RNNs) or transformers.

1 Introduction

In the realm of digital communication, the precision of language holds immense significance for effective information retrieval. This project addresses the prevalent challenges posed by spelling and segmentation errors, recognizing their impactful influence on the clarity and accuracy of textual information in information retrieval systems.

As we navigate the ever-expanding volume of digital content, the importance of accurate language usage becomes evident. Spelling errors, though seemingly innocuous, have the potential to introduce confusion and misinterpretation into messages. Simultaneously, segmentation errors, where words are improperly combined or split, add complexity to language processing mechanisms. Within the context of information retrieval systems, these errors can compromise search results, disrupt recommendation algorithms, and contribute to noise within expansive databases.

I began exploring automatic spell checking and segmentation error correction by studying Peter Norvig's insights on spell checker development shared in his blog [4]. This initial exploration laid the foundation for my understanding of the intricacies involved.

In my quest to optimize the spell checking algorithm, I came across the SymSpell repository [3], which introduced the concept of Symmetric Delete spelling correction. This idea significantly improved the efficiency of the spell checker.

To enhance the accuracy of my spell checker, I delved into refining onegram and bigram data. This effort resulted in a noteworthy improvement in accuracy.

While my spell checker performed well in correcting single-word errors and addressing segmentation issues individually, I faced a challenge in devising a unified function capable of rectifying both character and segmentation errors simultaneously.

2 Methods

2.1 The basic approach

Peter Norvig's spell checking algorithm [4] is based on probabilistic approaches and it is a simple and effective method for correcting spelling mistakes in

a given text. The following is a brief overview of the algorithm:

1. A large corpus of text (such as a collection of books, articles, or web pages) is used to build a frequency dictionary. This dictionary contains words and their corresponding frequencies of occurrence in the corpus.
2. For a given input word with a potential spelling mistake, the algorithm calculates the probability of the word being correct. This is done by considering both the frequency of the word in the corpus and the frequency of possible correction candidates.
3. The algorithm generates potential corrections by applying basic edit operations, such as inserting, deleting, substituting, and transposing letters, to the word being corrected. It explores all possible words resulting from one and two of these simple edits, and subsequently filters them by retaining only those found in the dictionary.
4. The algorithm calculates the probability of each candidate correction being the intended word. This probability is based on the frequency of the candidate in the corpus.
5. Finally, the correction is chosen by the following order of priority:
    - The original word, if it is known (i.e. in the dictionary); otherwise
    - The list of known words at edit distance one away, if there are any; otherwise
    - The list of known words at edit distance two away, if there are any; otherwise
    - The original word, even though it is not known.

Norvig's spell checking algorithm is particularly effective because it leverages the statistical properties of language. By relying on the frequencies of words and their variations in a large dataset, it can make reasonable assumptions about the likelihood of a particular spelling correction.

An important drawback of this method is its lack of consideration for the context in which the word is used. It treats the corpus as a mere collection of individual words, disregarding the surrounding context in which words may influence each other. This limitation could be somewhat mitigated by incorporating the occurrence of word pairs, commonly known as bigrams. Considering bigrams allows the algorithm to take into account the relationships between adjacent words, providing a more contextualized approach.

By employing bigram frequencies, it can also be constructed a pretty effective word segmentation function. The algorithm recursively splits the input and calculates the conditional probability of the segments. The result is a list of the most probable segments of the input. To make it somewhat efficient, memoization is used to prevent re-computing the segmentations of the remaining characters in a split.

2.2 Making it faster

SymSpell is an algorithm for spelling correction and approximate string matching developed by Wolfgang Bosma [3]. It guarantees a speed several orders of magnitude greater than the approach from Peter Norvig.

The core idea of the algorithm is to apply only deletion edits on both the term being corrected and the dictionary term. This way, insertion, substitution and transposition edits are completely replaced. This has several benefits:

- The generation of candidates is significantly faster if we are using only deletion edits.
- We can pre-compute all the possible deletions (up to a fixed edit distance) to the dictionary terms and save time on the generation of the candidate corrections.
- The pre-computed deletions are saved in a hash table where the term with deletions is used as the key and the value is the collection of all the dictionary terms that generated that key.
- Furthermore, the hash table's size can be minimized by computing deletions not directly

on the dictionary terms but on their prefixes: words with a length less than the prefix length remain unchanged, while words exceeding the prefix length are truncated to match the prefix length.

The increased performance given by the SymSpell method allows the use of edit distances greater than 2 which can improve accuracy by some margin. This will be evaluated in the results section.

2.3 Joint spelling and segmentation error correction

Overall, the methods discussed so far can handle spelling and segmentation errors individually, but struggle when facing both cases in a sentence. Correcting both of them simultaneously presents several challenges, as these two types of errors require different approaches and often interact in complex ways.

Segmentation errors can actually be of two types, incorrectly removed a white space and incorrectly added a white space. Hence, all the possible combinations of interaction between misspelling and segmentation error are:

- Misspelling in a word
- Added a white space inside of a word
- Split a word and first half has a misspelling
- Split a word and second half has a misspelling
- Split a word and both halves have a misspelling
- Concatenated two words together
- Concatenated two words and first one has a misspelling
- Concatenated two words and second one has a misspelling
- Concatenated two words and both have a misspelling

This representation is a simplification since we are just considering either only one split of a word or one concatenation of two words interacting with one misspelling (either insertion, deletion, substitution, or transpositions of letters). Moreover, we also need to take into consideration the fact that the misspelling and segmentation errors might generate valid words or couple of words in the monogram and bigram dictionary, leading to wrong suggestions.

Even with this simplification in mind I was not able to come up with a satisfactory method that could solve the joined issue. The actual problem lies in how to score and then choose between all the possible types of correction and I could not find a proper algorithm to solve this problem.

3 Datasets

The accuracy of the spell-checking methods heavily relies on the quality and representativeness of the frequency dictionaries employed. It is imperative that these dictionaries accurately reflect the true distribution of words in the language. A notable challenge arises from the fact that the source collection of documents used to compute these frequency dictionaries may contain spelling errors. This poses a potential threat to the accuracy of the algorithm, as it may fail to correct erroneous words if the dictionaries are contaminated with inaccuracies.

Striking a balance becomes crucial in selecting the right corpus for dictionary generation. While opting solely for literature books may provide cleaner text, it risks lacking representation of common speech, potentially omitting everyday language nuances. Therefore, a thoughtful selection of diverse text sources is necessary to ensure that the frequency dictionaries capture the richness and variety of the language while minimizing the impact of potential errors within the source documents.

Initially, this work employed the frequency dictionaries from Peter Norvig's website[1]. The files, namely 'count_1w.txt' and 'count_2w.txt', consist of

[1] https://norvig.com/ngrams/

approximately 333,271 monograms and 286,358 bigrams, respectively.

As highlighted in the results section, these files did not yield high accuracy due to a significant number of spelling errors impacting the quality of correction. To address this, I pruned all words with a count below 100,000, resulting in a notable improvement in accuracy.

Subsequently, I utilized the frequency dictionaries from the SymSpell library [2], which consist of 82,834 words for monograms and 242,342 bigrams. Despite the lower word count, the quality of these dictionaries proved significantly higher, and this enhancement is reflected in the improved results.

Test Set	Correct (%)	Unknown (%)	Speed (words/s)
Wiki	48	5	52
TOEFL	35	13	47
Aspell	26	7	19
Wiki	48	5	2500
TOEFL	35	13	3239
Aspell	25	7	3108

Table 1: Initial Evaluation of SpellCheck vs. SpellCheckFast. 'count_1w.txt' as the frequency dictionary and max edit distance = 2

4 Results

In this section, I will elucidate the outcomes of a series of assessments conducted on my spell-checking system. These evaluations were designed to gauge the system's accuracy, speed, and adaptability to varied misspelling scenarios.

4.1 Word correction without context

The first evaluations are performed on three distinct test sets: a list of commonly misspelled english words from Wikipedia [5], a dataset of Spelling Annotations for English language learner essays written for TOEFL exams [2] and the test set [1] used to evaluate Aspell, the standard spell checker for the GNU operating system. These test sets are all pairs of wrong / right words, hence do not include context in which the error occurred.

I evaluate on these test sets the classical version from Peter Norvig (SpellCheck) and the version which implements the SymSpell algorithm (SpellCheckFast) using different frequency dictionaries and values for the maximum edit distance.

First, let's delve into the results, shown in Table 1, obtained using 'count_1w.txt' as the frequency dictionary and a maximum edit distance set to 2. A striking observation emerges in the remarkable disparity in speed, with SpellCheck and SpellCheckFast exhibiting a difference of two orders of magnitude. Interestingly, the accuracy remains nearly identical, but this aligns with our expectations, considering that SymSpell's approach doesn't alter the selection process for the best suggestion but just enhances the speed of execution. Subsequent evaluations will exclusively focus on SpellCheckFast.

Test Set	Correct (%)	Unknown (%)
Wiki	69	8
TOEFL	52	13
Aspell	40	10
Wiki	80	5
TOEFL	69	11
Aspell	49	7

Table 2: SpellCheckFast using 'count_1w.txt' pruned of words with count less than 100,000 and with SymSpell's frequency dictionary. Max edit distance = 2

Moving on, we examine (Table 2) the accuracies achieved by SpellCheckFast when utilizing 'count_1w.txt' pruned of words with a count less than 100,000, as well as the frequency dictionary from the SymSpell library. In both configurations, the evaluations are conducted with a maximum edit distance set to 2. A noticeable improvement in ac-

[2] https://github.com/wolfgarbe/SymSpell

Test Set	Top-1 (%)	Top-3 (%)	Top-5 (%)	DCG	Speed (words/s)
Wiki	80	90	92	0.87	457
TOEFL	70	81	83	0.77	519
Aspell	52	68	72	0.63	559

Table 3: Extended Evaluation with SymSpell's frequency dictionary and edit distance 3 (SpellCheckFast)

curacies is evident across all test sets. Notably, the act of pruning the frequency dictionary alone results in a substantial enhancement in accuracy. This observation underscores the impact of maintaining a clean and high-quality frequency dictionary, as exemplified by SymSpell's dataset, which entails even better results.

Finally, we get to use max edit distance equal to 3 with the SymSpell's frequency dictionary. As illustrated in Table 3, this particular configuration resulted in the highest accuracy across all test sets. The table also provides insights into the top-3 and top-5 accuracies, accompanied by the Discounted Cumulative Gain (DCG). This measure is calculated for each entry in the test set by obtaining a list of the five most probable suggestions from the spell checker, ordered first by edit distance and then by their frequency in the dictionary. The single DCGs for each entry are then averaged across the entire test set, offering a comprehensive evaluation of the system's performance. We can also see that the speed reduced with a factor of 10x by employing the greater edit distance.

4.2 Word correction with context and word segmentation

We now move to the task of sentence correction and word segmentation. I delved in these two problems separately, first by looking at how well the algorithm performs at correcting sentences with and without bigram data. Then, I tested the word segmenter by concatenating all the words and then trying to resegment them. For this tests I used the book of Frankenstein by Mary Shelley as a test set, properly normalized and divided into small sentences.

The tests on sentence correction yielded intriguing results. I introduced spelling errors with a 10% probability on sentences from the normalized Frankenstein book, implying that each position in the string had a 10% chance of undergoing an edit involving deletion, insertion, substitution, or transposition of characters. Experimenting with maximum edit distances of 2 and 3, I observed consistent accuracies of 82% for the function utilizing monogram data and 83% for the one leveraging bigram data. While I expected a more substantial difference, the outcomes demonstrated a closer performance between the two functions than initially expected.

Finally, the segmentation test, that involved concatenating the words of each sentence and then applying the word segmenter, demonstrated a consistent accuracy of about 90%. Despite the straightforward nature of this function, the results indicate a noteworthy high accuracy. However, it's essential to consider performance implications as well. The segmenter leverages memoization through a cache decorator with an unbounded size limit. This approach rapidly escalates, evidenced by the cache size reaching 4,822,020 elements after testing on Frankenstein sentences. Furthermore, it became necessary to impose a maximum sentence length, as longer phrases quickly surpassed the maximum recursion limit.

Max Edit Dist	Accuracy Monogram Data	Accuracy Bigram Data
2	82%	83%
3	82%	83%

Table 4: Sentence correction using SpellCheckFast with SymSpell dictionaries

5 Conclusion

Throughout this project, I've delved into the world of spelling and segmentation errors in digital communication, especially within information retrieval systems. By exploring various spell-checking and segmentation correction methods, I've gained insights into the complexities and limitations of these language challenges.

Starting with Peter Norvig's work on spell checker development, I got a solid foundation in probabilistic approaches for fixing spelling errors. Then, I dived into SymSpell, a game-changer that sped up spell checking while keeping accuracy high.

The results revealed that pruning frequency dictionaries, such as 'count_1w.txt', by eliminating low-frequency words greatly improved accuracy. Additionally, leveraging SymSpell's frequency dictionaries exhibited superior results, surpassing the performance of other datasets. Whereas, using higher maximum edit distances did not show substantial differences in accuracy.

In conclusion, while these methodologies have yielded promising results, there remains ample room for further exploration. Future endeavors could involve the exploration of more complex approaches, such as those utilizing recurrent neural networks (RNNs) or transformers, particularly in addressing the joint spelling and segmentation error correction challenge. These advancements hold the potential to further refine and enhance the effectiveness of language correction mechanisms in information retrieval systems.

References

[1] *Aspell Spell Checker Test Set*. URL: `http://aspell.net/test/`.

[2] EducationalTestingService. *EducationalTestingService/TOEFL-Spell: Corpus of annotations for misspelings*. URL: `https://github.com/EducationalTestingService/toefl-spell`.

[3] Wolf Garbe. *SymSpell*. June 2012. URL: `https://github.com/wolfgarbe/SymSpell`.

[4] Peter Norvig. "How to write a spelling corrector". In: *http://norvig.com/spell-correct.html* (2007).

[5] Wikipedia contributors. *Commonly misspelled English words*. Jan. 4, 2024. URL: `https://en.wikipedia.org/wiki/Commonly_misspelled_English_words`.