REGULAR ISSUE

IIOAB JOURNAL

ARTICLE

GRAPH-BASED SENTENCE LEVEL SPELL CHECKING FRAMEWORK

Abdullah Talha Kabakus[1*], Resul Kara[2]

[1]IT Center, Abant Izzet Baysal University, Bolu, TURKEY

[2]Faculty of Engineering, Department of Computer Engineering, Duzce University, TURKEY

ISSN 0976-3104

ABSTRACT

Spelling mistakes are very common on the web, especially when it comes to social media, it is much more common since (1) users tend to use an informal language that contains slang, and (2) the character limit defined by some social services such as Twitter. Traditional string similarity measurements (1) do not consider the context of the misspelled word while providing alternatives, and (2) do not provide a certain way to choose the right word when there are multiple alternatives that have the same similarity with the misspelled word. Therefore, we propose a novel sentence level spell checking framework that targets to find "the most frequently used similar alternative word". 146,808 sentences from different corpora are stored in a graph database. The similarity is calculated by using Levenshtein distance algorithm alongside the similarity between two given words. As the experimental results are presented in the discussion, the proposed framework is able to correct misspellings which cannot be corrected by traditional string similarity measurement based approaches. The accuracy of the proposed framework is calculated as 84%. Since the proposed framework uses a slang dictionary to determine misspelled words, it can be used to correct misspellings in the social media platforms.

KEY WORDS
Spell check; string similarity; edit distance; social media; Twitter

Received: 4 March 2017
Accepted: 18 April 2017
Published: 5 June 2017

***Corresponding Author**
Email: talhakabakus@gmail.com
Tel.: +90-374 2534510
Fax: +90-374 2534526

INTRODUCTION

Spelling mistakes are one of the key areas that many software systems have been working on in order to correct them which is critical especially for the systems that are built-in semantics. Natural language processing and sentiment analysis are two research areas that need a proper solution to correct spelling mistakes. A meaningful sentence may become meaningless when a letter of a word that specifies the sentiment is misspelled. According to the recent report by Global Lingo [1], 59 percent of 1,029 participants would not use a company which contains obvious grammatical or spelling mistakes on its website or marketing material. Another result of the report is that 74 percent of the same participants tends to check the quality of spelling while visiting a company's website. There is also economic side of this issue especially when it comes to e-commerce companies that drive their all business through their websites. Spelling mistakes in product catalogs may prevent users to reach their targets since they search products through the search engines or the companies' own websites. According to the research by an online entrepreneur [2], spelling mistakes cost millions in lost online sales. The official reports, documents have no toleration for spelling mistakes since a single letter typo may be crucial and irreversible. For all those reasons, we propose a novel sentence level spell check framework that uses context-aware analysis to correct misspelled words. The main contributions of this paper are listed as follows:

- A novel approach which is based on context-aware analysis is proposed to correct spelling mistakes in order to provide not just syntactically similar replacements but also semantically too. The findings are discussed in order to shed light on the difficulties of spell checking.
- The database the proposed framework is constructed on is specifically selected as a graph database which not just provides constructing better data structure but also improves the overall system performance and makes the knowledge base more scalable.
- The proposed framework can be used to correct misspellings in social media platforms since it uses (1) a slang dictionary to be aware of slang, and (2) it is trained with various datasets which contain formal and informal sentences.

This paper is structured as follows: Section 2 presents the related works. Section 3 describes the material and method. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper with future directions.

RELATED WORK

VGRAM [3, 4] is a technique to improve the performance of approximate string queries based on choosing variable-length grams, which can be used to identify similar strings based on their common grams. Authors improved the proposed technique by providing some filtering and merging algorithms in order to merge inverted lists of grams generated by strings.

Various filtering approaches are proposed such as using relational database management system [5] to use approximate string join capabilities. Cohen [6] proposes a framework to integrate heterogeneous databases based on textual similarity and proposed a logic called WHIRL that reasons explicitly about string similarity using TF-IDF (frequency-inverse document frequency) term weighting. Grossman et al. [7, 8] propose an approach to represent text documents and their associated term frequencies in relational tables. They may boolean and vector-space queries into standard Structured Query Language (SQL) queries. Chaudhuri et al. [9] propose a new primitive operator called SSJoin which can be used to implement similarity joins based on a variety of textual and non-textual similarity functions.

Golding et al. [10] propose WinSpell, a context-sensitive spelling correction algorithm based on combining variants of Winnow, and weighted-majority voting. Since the work is proposed before the rise of social media and microblogs, it cannot be used to correct misspellings in social networking sites and microblogs as is not aware of slang used in these platforms. Also, unlike our work, their work is not expected to perform well for various domains since the training data differs from one domain to other. Unlike that, our knowledge base is specifically designed to contain corpus from different domains in order to propose a general spelling checking method.

Carlson and Fette [11] propose an approach based on memory-based learning techniques and a very large database of token n-gram occurrences in web text as training data. They use GNU Aspell [12], an open source spell checking library, to generate candidates for spelling correction alongside their own method. We generate candidates by solely using our knowledge base through the proposed algorithm which is based on the frequency of co-occurrence in the knowledge base, and our own similarity measurement between the candidate and misspelled words as it is discussed in detail in Section 3. That lets us define our own method to retrieve candidates from our knowledge base instead of a semi black box model which lets us extend our work in the future without any dependencies. The latest version of GNU Aspell, GNU Aspell 0.60.6.1, is released in July 2011 at the time of writing which is outdated especially when the new words that are arisen by the social media are considered. Our knowledge base can also be extended with data from different platforms when it is necessary.

Lapata and Keller [13] propose an approach based on the search engine Altavista (which is not available anymore, purchased by Yahoo! in 2003) to correct spelling mistakes through the retrieved search results. One advantage of this approach is that it removes the necessity to create a large knowledge base. However, Liu and Curran [14] report that this approach gives much lower accuracy than retrieving counts from a collection of web pages.

Unlike the related works, the proposed spell checking framework targets sentence level spell checking and correction which is based on a native graph data model and can also be used for the text in the social networking sites and microblogs since (1) It is trained with the corpora such as telephone dialogues, face-to-face speech to include the words specific to the informal language used in social media, and (2) unlike the related works, it considers slang such as "*ty*", "*lol*", "*cu*" as valid words which are commonly used in social media [15–17].

MATERIALS AND METHODS

Levenshtein distance

Levenshtein distance [18], also known as edit distance, is a metric to measure the similarity between two sequences by calculating how many atomic edits (i.e. insertions, deletions, or substitutions) are required to convert one string to another.

Sentence level spell check framework

The proposed sentence level spell check framework is a context-aware approach that corrects the misspelled word by considering the previous and following words. neo4j, a highly scalable open source native graph database, is used to create the required knowledge base for context analysis. Words of sentences are stored in a directed two-way graph database with using the `NEXT` and `BEFORE` relations. The `NEXT` relation is used to link the following word with the previous one, and the `BEFORE` relation is used to link the previous word with the following one. An example of how sentences are stored in the graph database is presented in [Fig. 1].

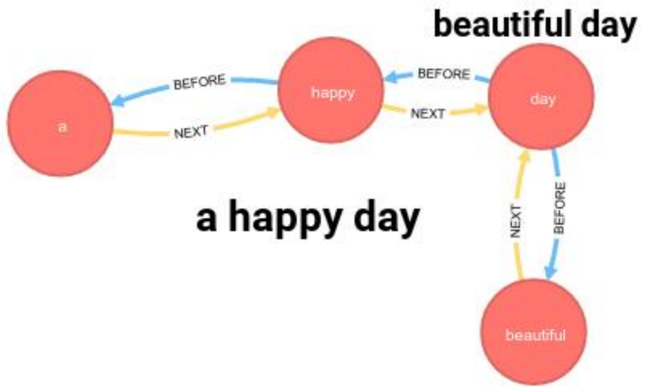


Fig. 1: An example of how sentences are stored in the graph database.

IIOAB JOURNAL ISSN: 0976-3104

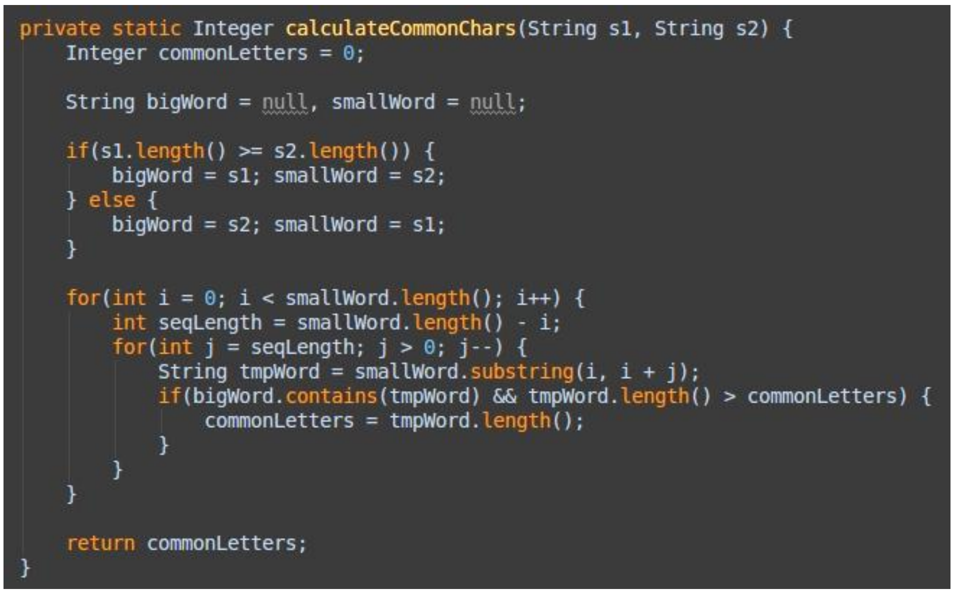
46,808 sentences from Open American National Corpus (OANC) and 100,000 sentences from Leipzig Corpora [19] are imported into the graph database in order to create a general knowledge base that can be used with any domains. As a total, 142,685 words (nodes) and 4,053,733 relations are constructed to present these 146,808 sentences in the graph structure. The algorithm that is used to present sentences in the graph database is presented in [Fig. 2].

```
for each sentence in the corpora
   parse the sentence into the array of words
   find the previous ($w_p$) and the following word ($w_n$) to $w_i$
   for each word ($w_i$) in the array
      if $w_i$ already exists as a node in the graph
         create the relations between $w_p$, $w_i$, and $w_n$
      else
         create the node to represent $w_i$
         create the relations between $w_p$, $w_i$, and $w_n$
      end if
   end for each
end for each
```

Fig. 2: The algorithm that is used to present sentences in graph database.

Spell checking and correcting

The proposed spell checking framework checks each word sentences in order to determine whether it is meaningful or not. This lookup is done by using the WordNet dictionary [20, 21], a lexical database for the English language that contains lexical categories such as nouns, verbs, adjectives, and adverbs, and also a predefined list of slang provided by Wiktionary, a project by the developers of Wikipedia. The latest version of WordNet, WordNet 3.0, defines 147,278 resources. If the word is not found in the WordNet database and the list of slang, it is supposed as a misspelled word and sentence level spell correcting process starts. The misspelled word is found in the graph database and its previous and following words are retrieved (if they exist). The graph database, neo4j, is queried by using the Cypher Query Language, an SQL-like declarative query language, which lets using two-way directed relations instead of writing traverses in the code [22]. Thanks to this ability, the following and previous words can be easily queried by just changing the direction of the relation. The similarity between each alternative and the misspelled word is calculated based on (1) the frequency of the usage of each alternative based on the previous and following words to the misspelled word, (2) string similarity measurement between each alternative and the misspelled word based on Levenshtein distance, and (3) number of common sequential letters between each alternate and the misspelled word. [Fig. 4] presents the method implemented using Java programming language in order to calculate common sequential letter count between two given strings.



```java
private static Integer calculateCommonChars(String s1, String s2) {
    Integer commonLetters = 0;

    String bigWord = null, smallWord = null;

    if(s1.length() >= s2.length()) {
        bigWord = s1; smallWord = s2;
    } else {
        bigWord = s2; smallWord = s1;
    }

    for(int i = 0; i < smallWord.length(); i++) {
        int seqLength = smallWord.length() - i;
        for(int j = seqLength; j > 0; j--) {
            String tmpWord = smallWord.substring(i, i + j);
            if(bigWord.contains(tmpWord) && tmpWord.length() > commonLetters) {
                commonLetters = tmpWord.length();
            }
        }
    }

    return commonLetters;
}
```

Fig. 3: The method implemented using Java programming language in order to calculate common sequential letter count between two given strings.

Let s to be the calculated similarity between w_i and w_a, w_i defines the misspelled word, wa defines each alternative word which is looked up in the knowledge base (the graph database) through the co-occurrence with the previous and following words to w_i. $lev(w_i, w_a)$ defines the Levenshtein distance between w_i

and w_a. com(w_i, w_a) defines the number of common letters between w_i and w_a. The formula to calculate the similarity between the misspelled word and each alternative is presented in [Fig. 4]. As it is seen in the formula, the similarity (s) is directly proportionate to the frequency of co-occurrence (freq), and the number of common sequential letters(com) but it is inversely proportional to the Levenshtein distance.

$$s(w_i, w_a) = \left(\frac{freq(w_i, w_a) \times com(w_i, w_a)^3}{lev(w_i, w_a)^3} \right)$$

Fig. 4: The formula to calculate the similarity between the misspelled word and each alternative.

....

RESULTS AND DISCUSSION

Finding the right word to replace the misspelled word cannot be found by using the methods based on solely string measurement without considering the context. [Table 1] presents some examples of misspelled words with the corrections based on the Levenshtein distance and the proposed method. As it is clearly seen in [Table 1], context-awareness is necessary to replace the misspelled word with the suitable word which is maybe not the most similar one in terms of string similarity. Also, when the string similarity based distances of the candidates are same, there is no certain way to select one of them as the replacement for the misspelled word while the proposed method does spelling correction even though string similarities of candidates are same thanks to the frequency of co-occurrence. The proposed method not just uses the string similarity measurement to correct the misspelled word but also reveals the contextual suitability of each alternative by considering its usage frequency with the previous and the following words to the misspelled word, the number of common letters between alternative and misspelled words, and the calculated Levenshtein distance between alternative and misspelled words.

Table 1: Some examples of misspelled words with the corrections based on the Levenshtein distance and the proposed method

Misspelling	Levenshtein Distance		The Proposed Method	
	Candidate	Levenshtein Distance	Candidate	Levenshtein Distance
Development faze	Faze	1	Phase	2
Crew custome	Custom	1	Costume	2
Radar sencor	Censor, sensor	1,1	Sensor	1
Wheter forecast	Whether	1	Weather	3
Academic celender	Calender	1	Calendar	2
Flaot warning	Float, flood	2,2	Flood	2
Psychiatric petienthe	Patience	2	Patient	3
Christmas presenthe	Present, presence	2,2	Present	2
First-quarter prophit	Prophet	1	Profit	2
Quate village	Quite	1	Quiet	3

Since we could not find a publicly available dataset which fits to evaluate the proposed framework, the proposed framework is evaluated with a list which contains 188 commonly misspelled English words alongside the correct ones shared by Wikipedia based on the three different lists [23–25]. Sample sentences that contain these misspelled words are found from a various of online resources such as microblogs, and social networking sites since they commonly contain spelling mistakes [15, 26]. The proposed framework is evaluated through these sample sentences and the accuracy of it is calculated as 84%. The knowledge base does not contain 14 words such as "*hors d'oeuvres*", "*memento*" of these 188 words. When these words are ignored, the accuracy of the proposed framework increases to 90.8%. Since the proposed framework uses context analysis during spell correction, it is evident that the accuracy of the proposed framework depends on the diversity of its knowledge base. This knowledge base can be extended in order to include more words by importing corpora that contain grammatically and semantically right sentences while keeping homogeneity of the data. When we analyze the false corrections, the following reasons are recognized: (1) Some words are not found in the knowledge base, and (2) some words are less similar but a lot more frequently used within the context of the misspelled word than the right replacement.

The data structure of the knowledge base is as critical as the diversity in order to query the data in the right way to the proposed algorithm. Graph databases are the perfect solution to store the data which contains bi-directional relations between entries as it happens for natural language sentences. Using a graph database not just provides constructing better data structure but also improves the overall system performance as Vicknair et al. report that neo4j performs significantly better than MySQL, a relational database management system, for full-text character searches [27].

CONCLUSION

Correcting misspelled words to make them meaningful is critical for the systems based on semantics. Since the traditional string similarity measurements do not consider the context of the misspelled word, they do not look for a suitable word in terms of contextual accordance. Therefore, we propose a novel approach to correct misspelled words by not just considering string similarity between words but also contextual similarity is also considered. Both previous and following words to the misspelled word are analyzed in order to find the most frequently used similar word to replace the misspelled word with the correct candidate. Thanks to this context-awareness, the proposed framework is able to correct misspellings that cannot be corrected by traditional string measurement based spell correction approaches as the test results presented in the discussion. According to the experimental result, the accuracy of the proposed framework is calculated as 84% which is promising and encourages us for future improvements. The knowledge base is stored in a native graph database in order to provide more flexibility and better performance while querying the context of the misspelled word. Also, using the direction property of the relations between nodes (words), it is possible to query the data in both ways without writing traverses in the code. Since the knowledge base contains corpora from face-to-face speech and telephone dialogues which are more informal compared to books, reviews, the proposed framework is aware of slang and optimized to check the spelling on the social media platforms.

The proposed framework is specifically designed to not have any external dependencies in order to be open for further improvements. As a future improvement, authors would like to use sentiment analyzing methods while finding the suitable alternative word to replace the misspelled one by detecting the overall sentiment of the sentence. Also, the knowledge base can be extended by importing corpora that contain grammatically and semantically right sentences which are common in social media. Finally, we would like to evaluate the proposed work with a bigger dataset.

CONFLICT OF INTEREST
Authors confirm there is no conflict of interest.

ACKNOWLEDGEMENTS
None

FINANCIAL DISCLOSURE
Authors confirm there are no financial disclosures.

REFERENCES

[1] Hesse J. [2013) http://realbusiness.co.uk/article/24623-poor-grammar-on-websites-scares-59-away. Accessed 05 May 2017.
[2] Coughlan S (2011) http://www.bbc.com/news/education-14130854. Accessed 05 May 2017.
[3] Li C, Lu J, Lu Y. [2008] Efficient merging and filtering algorithms for approximate string searches. In: ICDE '08 Proc. 2008 IEEE 24th Int. Conf. Data Eng. Washington, DC, USA, pp 257–266.
[4] Li C, Wang B, Yang X. [2007] VGRAM: Improving Performance of Approximate Queries on String Collections Using Variable-Length Grams. In: VLDB '07 Proc. 33rd Int. Conf. Very large data bases. University of Vienna, Vienna, Austria, pp 303–314.
[5] Gravano L, Ipeirotis PG, Jagadish HV, Koudas N, Muthukrishnan S, Srivastava D. [2001] Approximate String Joins in a Database (Almost) for Free. In: VLDB 2001, Proc. 27th Int. Conf. Very Large Data Bases. Roma, Italy, pp 491–500.
[6] Cohen WW. [1998] Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proc. 1998 ACM SIGMOD Int. Conf. Manag. data. Roma, Italy, pp 201–212.
[7] Grossman D a., Frieder O, Holmes DO, Roberts DC. [1997] Integrating structured data and text: A relational approach. J Am Soc Inf Sci 48:122–132.
[8] Lundquist C, Frieder O, Holmes DO, Grossman D. [1999] A Parallel Relational Database Management System Approach to Relevance Feedback in Information Retrieval. J Am Soc Inf Sci 50:413–426.
[9] Chaudhuri S, Ganti V, Kaushik R. [2006] A Primitive Operator for Similarity Joins in Data Cleaning. In: ICDE '06 Proc. 22nd Int. Conf. Data Eng. Atlanta, GA, USA, pp 5–16
[10] Golding AR, Roth D, Cardie C, Mooney R. [1999] A Winnow-Based Approach to Context-Sensitive Spelling Correction. Mach Learn 34:107–130.
[11] Carlson A, Fette I. (2007) Memory-based context-sensitive spelling correction at web scale. In: Proc. 6th Int. Conf. Mach. Learn. Appl. ICMLA 2007. Cincinnati, Ohio, USA, pp 166–171.
[12] Atkinson K. [1998] GNU Aspell. http://aspell.net. Accessed 05 May 2017.
[13] Lapata M, Keller F (2005) Web-based Models for Natural Language Processing. ACM Trans Speech Lang Process 2:1–31.
[14] Liu V, Curran JR. [2006] Web Text Corpus for Natural Language Processing. Eacl 233–240.
[15] Montejo-Ráez A, Martinez-Cámara E, Martin-Valdivia MT, Ureña-López LA. [2014] Random Walk Weighting over SentiWordNet for Sentiment Polarity Detection on Twitter. Comput Speech Lang 28:93–107.
[16] Go A, Bhayani R, Huang L. [2009] Twitter Sentiment Classification using Distant Supervision. Processing 150:1–6.
[17] Go A, Huang L, Bhayani R. [2009] Twitter Sentiment Analysis. Entropy 17.
[18] Levenshtein VI. [1966] Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys Dokl 10:707–710.
[19] Quasthoff U, Richter M, Biemann C. [2006] Corpus portal for search in monolingual corpora. In: Proc. fifth Int. Conf. Lang. Resour. Eval. (LREC 2006). Genoa, Italy, pp 1799–1802.
[20] Miller GA. [1995] WordNet: a lexical database for English. Commun ACM 38:39–41.
[21] Fellbaum C (1998) WordNet: An Electronic Lexical Database. MIT Press. doi: 10.1139/h11-025.
[22] Batra S, Tyagi C (2012) Comparative analysis of relational and graph databases. IJSCE Int J Soft Comput Eng 2:509–512.
[23] Mastin L. [2008] Some Commonly Misspelled Words. http://www.lukemastin.com/testing/spelling/cgi-bin/database.cgi?action=misspelled. Accessed 05 May 2017.
[24] Nordquist R. [2008] The 201 Most Commonly Misspelled Words in English. In: About.com. http://grammar.about.com/od/words/a/misspelled200.htm. Accessed 05 May 2017.
[25] [2008] 100 Most Often Misspelled Words in English. In: LoveToKnow, Corp. http://grammar.yourdictionary.com/spelling-and-word-lists/misspelled.html. Accessed 05 May 2017.

ISSN: 0976-3104

[26] Saif H, He Y, Alani H. [2012] Alleviating data sparsity for twitter sentiment analysis. In: Proc. WWW'12 Work. "Making Sense Microposts." Lyon, France, pp 2–9.

[27] Vicknair C, Macias M, Zhao Z, Nan X, Chen Y, Wilkins D. [2010] A Comparison of a Graph Database and a Relational Database. In: ACM SE '10 Proc. 48th Annu. Southeast Reg. Conf. Oxford, Mississippi, USA, p 42:1–42:6.