The Home Depot, 2455 Paces Ferry Rd SE, Atlanta, GA 30339

Detecting and Correcting Real-Word Errors in E-Commerce Search *

Jared Moore, Mingzi Cao, and Rongkai Zhao

March 8, 2020

Abstract

When searching for products on an e-commerce site, user input is often subject to error. Simple spelling errors such as "hamer" instead of "hammer" are typically easy to detect and correct, as the original term, "hamer," will not be present in the dictionary that the system is using. A much harder class of errors to detect and correct are Real-Word Errors (RWEs) that occur when all of the words in a phrase are present in a dictionary, but the phrase itself does not make sense, such as "hammer grill" (instead of "hammer drill"). We implemented existing RWE detection methods (based on a confusion set) into our Spell Correction system and calculated accuracy on a hand-labeled dataset. After that, we used our confusion set to come up with an occurrence estimate for RWEs in the search space. Finally, we built a novel classification method based on our customer behavior data that can filter out false positive RWEs queries with high accuracy.

1 Introduction

In the study of e-commerce search (or any form of text search), the developers of the search engine are subject to the search terms (or queries) entered by their users. While most of the search terms may be well-formed, any sort of user input is known to be subject to human error, like spelling errors, which have been well-studied in academia. Many approaches exist for fixing traditional spelling errors, such as "*hamer*" $\rightarrow$ "*hammer*". These errors are typically referred to as Non-Word Errors (NWEs) since the original term is not a word. There have even been studies and patents on different approaches that one should take when correcting spelling on different devices, such as phones[1]. Another area inside of spell correction that has received a lot of attention (and is the focus of our paper) is the domain of *real-word errors* (RWEs). Unlike a NWE, an RWE is a spelling error that is present not because the spelling of any individual token is incorrect (i.e. not present in a our dictionary) but instead because the phrase itself does not make sense. Much research has gone into the area of tokenization in Natural Language Processing (NLP) and we do not attempt to address the topic in this paper. Pinto et. al provides one good summary of existing methods[2] if interested. For our context, we refer to a token as any string of characters separated from another by any amount of whitespace. Consider the phrase "brown much." Both "brown" and "much" are valid words in the English language but together the phrase does not make a lot of sense. Suppose we had a spell correction system that corrected "much" to "mulch" in this instance. Now we have the query "brown mulch" which is a popular home improvement product. However, there are valid cases that the word "much" could be used, so it cannot be replaced by "mulch" in all cases. In this paper, we conduct three experiments with the goal of detecting and correcting RWEs.

1.1 Related Work

Starting as early as 1980, there has been research done in the area of automated spell correction[3][4] by James Peterson. His paper did not specifically call out RWEs, but simply identified spelling errors as missing, extra, wrong, or transposed letters. RWEs can indeed fall into Peterson's classification, but are much harder to detect than the situation where the token is not present in a dictionary. Though they did not adopt the terminology "Real Word Error," Mays et. al extended Peterson's research by acknowledging the fact that such errors existed and proposing a solution to detect them[5]. These early approaches used a probabilistic model to decide if a token belonged in a given bigram or trigram, which will be used when describing our approach to fix RWEs. A different approach by Burnard and McEnery used part-of-speech tagging to determine when a specific word did not belong in a sentence[6]. Both Mays et. al and Peterson noticed that the ability to decide if a phrase contained a spelling error was related to the size of the dictionary that the spell correction system was using, and noted that as size increased, the computational power needed for the system would also increase. Wilcox-O'Hearn et. al extended the research done by Mays et. al to see

*Keywords: spell correction, e-commerce, search engine, real word errors, clustering

what actual performance would be on a real dataset, rather than one with synthetically generated errors[7], and found the F1-score of corrections to be at best 70%, and usually much lower. As mentioned in Section 2.2, The Home Depot (THD) needs a high-accuracy algorithm if we were to implement any approach.

Around 2005, literature started appearing that was formally calling out RWEs[8] as Real Word Errors. Pedler and Mitton described them as "Cupertinos," referring to the behavior of an old version of Microsoft Word when a user typed in "cooperation" instead of "co-operation"[9]. Many studies, such as the one done by Wilcox-O'Hearn, involve using a synthetic dataset instead of real data since it is much easier to generate labeled data than to find RWEs in real text[10]. Two interesting studies using real data was done by Max and Wisniewski in 2010 and Hesch in 2012, where they used Wikipedia revision history in an attempt to build a model that could detect when a word was used in the wrong context[11][12].

What we have seen in the existing studies is one main shortfall that is very difficult to overcome in an academic setting: the research was not done on a large corpus of real user data. Sometimes the dataset is synthetic, other times it is very small. Regardless, what we are contributing to the space of RWE detection and correction is how these algorithms that are being discussed might perform in an enterprise setting. We believe our research will start to shine a light on the algorithms that are worth further pursuit and highlight other factors such as runtime that are not usually considered when doing academic research on the topic of spell correction.

2 Detecting Real-Word Errors

The first experiment we ran was to produce empirical results based on our analysis of user-behavior data for RWEs using an algorithm we developed for our spell correction (SC) system based off the approaches used in current literature. As Wilcox-O'Hearn noted, detecting RWEs is the hardest part of the SC problem: once an RWE can be detected, fixing them is very similar to fixing any other spelling error[10]. Our approach to correcting RWEs is similar to the one proposed by Semanta and Chaudhuri[13], but it contains some important differences. We present the first empirical study on real search data, rather than a synthetic dataset or a different form of text, and the largest study on RWEs to date. Peddler and Mitton pointed out that while there have been many studies using confusion sets, they are typically the same set, or a very small set of data[9]. Even Peddler and Mitton's study, which used a larger set of words than had been studied before, only examined a document with 21524 total (non-unique) words and they identified 833 RWEs. The results from Peddler and Mitton's work imply that about 4% of all tokens are RWEs. But will this apply to our industry and to search terms instead of blocks of text? While such studies are indeed useful in the abstract sense, it is difficult to see if the approach will work for a specific domain or with larger datasets.

2.1 Our Domain THD has a dictionary of over 50,000 words and processes millions of searches every single day. Our search terms are typically 2-3 tokens in length and last year we saw over 10 million unique tokens. With such volume, will the approaches that worked for 20,000 words work for us? Our results are especially important to others working in the domain of e-commerce search. Further to this point, until now, there have been very few major published studies regarding RWE correction in the search space. Most studies that have been conducted are instead for text in general. For reasons described in Section 2.4, the search space is fundamentally different than general text input, and is important to study independently, although there are still many similarities.

2.2 Considerations regarding the Feasibility of Approaches At THD, we have a collection of services that work together in order to provide a high-quality search experience to the customer. Because it's role in this collection, our SC service only has a very limited timeframe to respond to the service that calls it. While there have been many different approaches to fixing RWEs, such as [5], [14], [8], [7], [10], [13], and [15], we only focused on those that we believe can fit into the total spell correction runtime requirement (hereby referred to as the Service Level Objective, or SLO). While we wholeheartedly believe more accurate solutions could be developed, algorithms that could not fit within our SLO were not considered in our experiments. Another point to keep in mind is that some algortihms, such as the one proposed by Hirst and Budanitsky[8], provide an estimated accuracy of around 20%, which would make it difficult to justify the algorithm if it was the only one used. That being said, any algorithm that improves our overall accuracy (i.e. fixes more errors than it breaks) without sacrificing runtime is a candidate to move to production.

2.3 Considerations regarding the existing THD Spell Correction Service While this paper will discuss approaches to identifying and correcting RWEs, it is important to acknowledge that the THD SC service consists of an ensemble of algorithms that we have de-

signed to work together to fix all spelling errors it sees. It is outside of the scope of this paper to discuss exactly how those algorithms work, though in Section 4, performance before and after adding our RWE-correction algorithm to the ensemble will be shown.

2.4 The Search Space As mentioned earlier in Section 2, the search space is not the same as when a user is writing a sentence or series of sentences. The research done by Max and Wisniewski and Hesch examined the occurrence of spelling errors in Wikipedia, which is a collection of full paragraphs written by users. As mentioned earlier, we have user queries that have a length of 2-3 tokens on average. While other researchers have been able to detect the occurrence of RWEs in longer sentences (Golding and Roth even defined the problem as requiring a sentence[14]), would they even happen in a shorter phrase like a search query? Knowing the domain, it is easy to come up with hypothetical examples: consider a user searching for "hammer grill" instead of "hammer drill". But do these queries happen? If so, how often do they happen? Singh and Singh claimed that RWEs represent 35% of all spelling errors (in Hindi, but perhaps this applies to other languages as well[1])[17]. If this is actually the case, fixing all RWEs (or at least trying) could be a huge lift for THD. As mentioned in Section 2, there have been surprisingly few studies on how often RWEs occur, usually studies focus on how to detect and/or fix them. See Section 4.1 to see our estimate of how often RWEs occur in our query log.

It is important to note that our experiments discuss performance on datasets that are specific to THD user search queries, though we hope that the results will be applicable in the general sense. As we implied in Section 2, the THD spell correction system has its own dictionary based on the search terms that we get. Our dictionary will likely be different than any other dictionary used, especially those in academia, as we have to handle words like brand names that are not present in the English language. Further, not all words in the English language are relevant to our domain, and thus we do not need to include them in our dictionary. Since our data is proprietary, the datasets used cannot be released publicly.

3 Implementation

In this paper, we provide three main findings: first, the accuracy improvement we saw when we tried to fix them, second, the estimated occurrence of RWEs in our query log by volume, and third, our effort to make an RWE classifier using customer behavior data. While the majority of the approaches described in Section 1.1 typically rely on synthetic datasets in order to calculate accuracy, at THD we have the advantage of having access to a vast dataset of user-entered queries. Because of this dataset, we are able to check and see how often any suspected RWE query actually happens on our website, if at all. In order to come up with potential RWE candidates, we started by generating a confusion set, similar to the ones done by Mays et al. and by Fossati and Di Eugenio in their respective studies[5][18]. Mays et al. used (Levenshtein) edit distance to find similar words for each word in a dictionary that were within a single letter edit. Fossati and Di Eugenio used phonetic and orthographic methods to find similar words, but, in the abstract sense, both approaches were based on the process of "given each word in a dictionary, find all of the valid similar words that it could be mistaken for." Peddler and Mitton used a similar process to Mays et. al, taking all candidates for a given word and ranking them using the process Mitton developed, in his 1996 research on developing a spell corrector[9][19], to find the best replacement for a token, given a set of candidates. Instead of only finding a single replacement, when we started investigating RWEs in our query log, we did not want to filter out any potential candidates and start with the largest set possible.

3.1 Initial Candidate List To come up with the initial confusion set, we used a combination of the approaches taken by Mays et. al and by Fossati and Di Eugenio. Suppose we have a dictionary D that is a set of words in our space, where $|D| > 1$. Suppose $L(x, y)$ is a function that will return the Levenshtein distance between x and y, $P(x)$ returns a set of tokens with the same phonetic code[20] as x, and $S \leftarrow S \cup x$ denotes adding an element x to the set S. The algorithm described in Algorithm 1 shows this process.

Running Algorithm 1 results in a large confusion set, mapping unigrams to sets of unigrams. Now, as pointed out in other research, even as early as Peterson's first paper, there are two pieces to the spell correction process: identification of errors (spell checking) and fixing the errors (spell correction). In other research, such as Pedler's thesis[21], we see the confusion set being used to see if the probability of there being an error decreases when a word is replaced with one from the confusion set, which makes sense intuitively. However, as we mentioned in Section 2.2, this is not how the THD SC system is built and we need a solution that will fit into existing system and comply with our strict runtime

[1] Fashola et. al did some early research on the patterns of spelling errors for students learning a new language (specifically English to Spanish)[16], but to this date there have not been any studies on the similarities of spelling errors across different languages.

Result: $M < u, S >$ a map, mapping a unigram u to a set S of candidates u could be replaced with
$M \leftarrow \emptyset$;
[2] **foreach** $word \in D$ **do**
  $S \leftarrow \emptyset$;
  foreach $other \in D$ **do**
    if $word \neq other \wedge L(word, other) \leq 2$ **then**
      $S \leftarrow S \cup other$;
  end
  foreach $other \in P(word)$ **do**
    if $L(word, other) \leq 3$ **then**
      $S \leftarrow S \cup other$;
  end
  $M \leftarrow M \cup < word, S >$;
end
return M

Algorithm 1: Generating Unigram Confusion Candidates

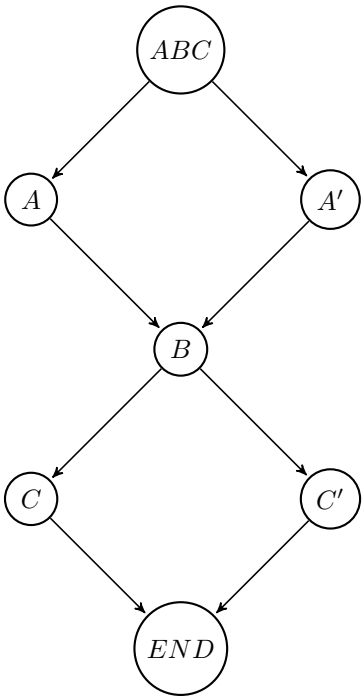

Figure 1: Sample of aggregation process. Here the candidates $ABC, A'BC, A'BC', ABC'$ will be sent to the ranker.

requirement.

3.1.1 Aside on the THD Spell Corrector At THD, we have split the spell correction problem into three pieces: candidate generation, aggregation, and ranking. For simplicity, let us assume that our SC system has already decided that the input query is spelled incorrectly (or at least has the potential to be improved). During the candidate generation phase, we take each token, and generate possible replacements for that token (this does not include the RWE confusion set). During aggregation, we take every combination of the candidates, then pass those to a ranking system and choose the highest-ranked suggestion. Suppose we started with the tokens ABC, that A had a candidate A', and C had a candidate C'. The resulting strings that are sent to the ranker are $ABC, A'BC, A'BC', ABC'$. Clearly, if each unigram had many possible RWE replacements, the total candidates could explode and take a lot of time to be ranked. See Figure 1 for a visualization of the candidate aggregation process.

3.1.2 Bigrams instead of Unigrams Looking at Figure 1, one can easily see how adding many potential candidates could explode the total query strings we send to the ranker. While others such as Fossati and Di Eugenio, Mays et. al, and Peddler and Mitton generated candidates for individual words, we instead decided to generate candidates for bigrams. The goal of such a confusion set is to ultimately be able to decide, given an input bigram, if there was an RWE and, if so, the ideal replacement for the bigram. To do this, we took every permutation of 2 tokens from our dictionary and put them into a set with every possible RWE replacement for each unigram. For each of the sets we generated, we took the bigram with the highest occurrence in our query log and marked it as the "correct" bigram, and the rest as possible RWEs. One check we did was to ensure there were no chains of mappings, for instance if bigram AB was mapped to CB and then CB was mapped to DC. Any cases that were found with such a chain simply set the last node in the chain as the correct bigram for each node preceding it (we did not find any cycles while doing this). Some examples from our confusion set can be seen in Figure 2.

3.2 Picking the Correct Replacement After aggregating all of the candidates for the query, we then iterated through all of the tokens in each candidate using a sliding window of size 2. For each element in the window, we check the bigram against our RWE candidate set and then add the replacement as a new candidate in the query candidates. When we move to the next window, we mark the previous tokens as "final" and no longer check for new RWEs. However, new queries added to our query candidates will be checked for additional RWEs starting at the first non-final token. There

Correct Bigram	RWE Bigrams
brita faucet	["britta faucet", "britta facet", "britt faucet", "brita facet"]
roof hammer	["room hammer", "rough hammer", "ruff hammer", "roof hummer"]
tile drill	["tale grill", "tite drill", "till drill", "tile grill", "tile dill", "title grill"]

Figure 2: Selected examples of bigrams from our confusion set

will be no such instance where one RWE replacement leads to another (per design in the confusion set), so that will not generate additional candidates. Once this is done, we then pass the data to our ranking service, which chooses the best correction. In the context of this study, we only changed the candidate generation process and the ranker was not touched. For simplicity, one can think of the ranker simply as a function that takes in how often the candidates appear in the THD query log as well as some other features and returns a scalar value (and we select the highest value as the correction).

4 Results of Fixing RWEs on Hand-Labeled Datasets

How did the change we made perform compared to how we were doing before? While other studies examined the performance against well-known datasets, these datasets are not necessarily relevant to THD. As mentioned in Section 2.4, our SC system has been designed to correct THD-specific search terms and may not be applicable to general datasets, so the percentages below may or may not apply in the general sense. More research would need to be done to examine the similarities of spelling errors between search and other forms of text. To properly measure the accuracy of our implementation, we created a hand-labeled dataset of around 1500 RWEs from our query log. Every single example was an actual RWE that we expected to be changed to some ground truth correction. We also ran a test against our standard regression suite, consisting of a random sample of 2200 queries from our query log, hand-labeled as correct or incorrect, with the best correction for the query if it is incorrect. None of the errors in the regression set have been classified as RWE or non-RWE, but since it is a random sample from our query log, the dataset likely contains some RWEs. In our regression set, about

Algorithm	RWE Set	Regression
Base	4.80%	79.18%
Base + RWE	19.99%	76.01%

Figure 3: Results of our tests on the RWE and Regression datasets.

57% of the queries are spelled correctly, and the rest are incorrect, so an accuracy value of 57% would be the trivial approach of not changing any queries. We ran a test against both the regression set and the RWE set and the results of our experiment can be seen in Figure 3. As expected, the RWE fix did increase the accuracy on the RWE dataset, by 15.19%. We saw a slight drop in the overall regression set but the lift in RWE performance may offset this reduction, depending on the occurrence of RWEs in our query log (discussed in Section 4.1). To explain the drop in the regression set, we believe that the RWE corrector was adding additional candidates that got ranked higher than the other ones but were not the best replacement for the original query. More work would need to be done to tune the algorithm so it does not interfere with the existing correctors.

4.1 Occurrence of RWEs As promised in Section 2, we also spent some effort to determine the frequency of RWEs occurring in our query log. If it turns out that these errors do not happen very frequently, maybe the effort to fix them is not worth the investment. Luckily, this was not the case. To figure out an estimate of RWE occurrence we tried the following. To start, we took our bigram confusion set and looked up how often the original queries (suspected errors) occurred in our query log. We simply looked for the presence of the bigram inside of a search term, no investigation was done to see if the presence of other tokens on either side of the bigram changed the phrase enough for it to no longer be an error. These original queries comprise about 4% of our total query volume, nowhere near the 35% estimated by Singh and Singh. Interestingly enough, this is very close to the percentage of words that were RWEs that Peddler and Mitton found to be RWEs in their document, though whether words in a document and search terms have the same error rate in general remains to be seen.

Now, we had a suspicion that this list of suspected RWEs was not 100% so we wanted to see an approximate accuracy of our identification process. We took a random sample of 1000 queries from the candidate set and hand-labeled each query as an RWE or not an RWE. What we saw was that about 70% of our suspected RWEs were actually RWEs, which we were quite

pleased about as it is higher than what Wilcox O'Hearn found in her paper when she performed a similar experiment. We then multiplied the accuracy value by the occurrence, giving us the estimate of $4\% \times 80\% = 2.8\%$ of our annual query volume containing an RWE. Given that we now have labeled RWE data, can we use this dataset to do better at detecting RWEs?

5 Building a (non-)RWE classifier

After seeing the accuracy of our confusion set, we wondered if we could do better using additional data signals. Since we have access to a vast amount of customer data, we wondered if we can use the information about how our customers behave when searching the different phrases to help us determine whether or not a phrase is an RWE.

5.1 Methodology What are the best customer metrics that we can use to enhance our understanding of RWEs? Looking at our customer data, we have many related data points, for example: product clicks, cart adds, and purchases. For simplicity, we assume a purchase needs to have a cart add first, and a cart add needs a click first. As expected, we have less purchases than cart adds, and less cart adds than clicks. In order to have the best signal of search term effectiveness, we decided to go with clicks as our first feature. The second feature we used was already incorporated in a way in our confusion set: annual search volume. The intuition here is the more often a term is searched, the more likely it is to be correct (as found by Whitelaw et. al in 2009 [22]). Third, we integrated the usage of another feature of our website: type-ahead (TA). Our TA system recommends search terms roughly based on a combination of occurrence and revenue (so it should be slightly related to the first two features), but the way it was built, there should never be any form of spelling error in the suggestions. We suspected that the more TA usage we saw for a query, the less likely it would be an RWE. A plot of a sample of our RWE candidate set and some high-volume queries can be seen in Figures 4 and 5. We decided to add high-volume queries as well since the most popular queries on our website are almost definitely not RWEs, and any classifier we make should definitely not flag them as RWEs.

In the graph in Figures 4 and 5, we see that the points in the bottom left corner are a mix of RWEs and non-RWEs, which seem to be difficult to separate given the current set of features. However, what we also see is that the further from that corner a point is, the less likely it is to be an RWE. Figure 5 especially shows the separation as all of our RWE candidates are in the corner while more popular queries are further away in all

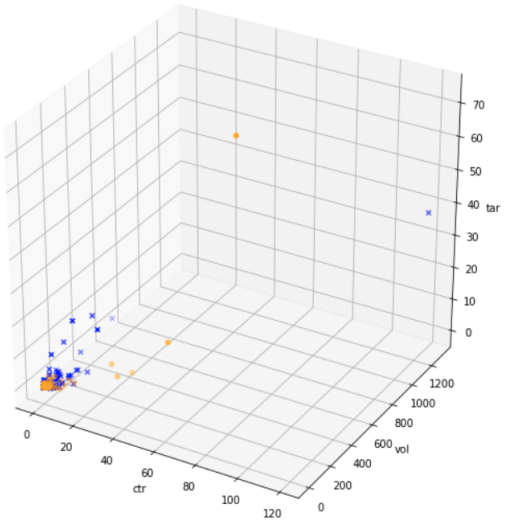

Figure 4: A visualization of RWEs and non-RWEs in our feature space. The blue points are not RWEs and the orange points are RWEs.

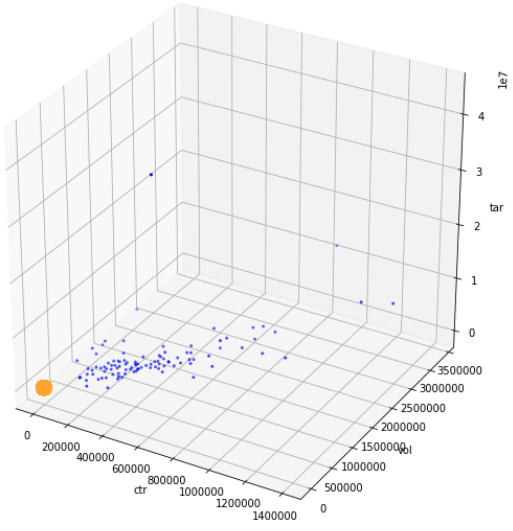

Figure 5: A visualization of RWEs, non-RWEs, and high-volume queries in our feature space. The RWE and non-RWE cluster from Figure 4 is represented by the orange circle in the bottom right.

Query	Is RWE? (Y/N)	Corrected Query
bio soil	N	N/A
front tires	N	N/A
dark tiles	N	N/A
floor fridge	N	N/A
antic subway	Y	antique subway

Figure 6: Sample of the queries that are classified as "non-RWE" by our classifier.

dimensions. To build a classifier off of our observation, we wanted to make a way to tell if a query belonged in a given cluster. We could have used the distance from the center of the cluster, but one look at our graphs show that our clusters are not spherical. Instead we used the average inter-point distance between all of the points in the cluster to every other point to calculate the mean and standard deviation for distances between points. Using these measurements, we made a simple classifier that would label a point as non-RWE if it was more than 1 standard deviation from the mean after calculating the average inter-point distance.

5.2 Classifier Results While we did originally build the classifier with the intent of detecting RWEs, we were only able to achieve the opposite. As seen in Figures 4 and 5, the RWE and non-RWE points from our confusion set are mixed in the bottom cluster. As we expected, the high-volume terms were completely separate from the other cluster. A sample of the queries classified as "non-RWE" from our classifier can be seen in Figure 6. After manually inspecting the results, we found that 100% of the high-volume queries were classified as non-RWE. Of the queries from the confusion set that were outside of 1 standard deviation, 93% were correctly labeled as not RWEs, giving us very high confidence in our classifier.

6 Conclusions

What we have discovered in our study is that while RWEs do occur in a significant percentage of our query volume, fixing them is not easy to do without breaking other fixes. We built a confusion set with potential replacements similar to the work done in previous research, but what we found was that this confusion set was only able to give us a minor improvement fixing RWEs. To improve our ability to detect RWEs, we set out to develop a way to classify a search term as an RWE or not an RWE. Using customer behavior features such as search volume, type-ahead usage, and clicks, we were able to develop a novel classification method to help identify queries that are not likely RWEs. While our classifier was not able to easily separate RWEs from all non-RWEs, it is able to identify queries that are not RWEs with high confidence, allowing us to further filter down our confusion set and ideally achieve higher accuracy fixing RWEs.

7 Acknowledgements

We would like to acknowledge Nagaraj Palanichamy for helping with part of the implementation of the algorithms described in this paper.

References

[1] Nicole Coddington, 05 2014.

[2] Alexandre Pinto, Hugo Gonçalo Oliveira, and Ana Oliveira Alves. Comparing the Performance of Different NLP Toolkits in Formal and Social Media Text. In Marjan Mernik, José Paulo Leal, and Hugo Gonçalo Oliveira, editors, *5th Symposium on Languages, Applications and Technologies (SLATE'16)*, volume 51 of *OpenAccess Series in Informatics (OASIcs)*, pages 3:1–3:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[3] James L. Peterson. Computer programs for detecting and correcting spelling errors. *Commun. ACM*, 23(12):676–687, December 1980.

[4] James L. Peterson. A note on undetected typing errors. *Commun. ACM*, 29(7):633–637, July 1986.

[5] Eric Mays, Fred J. Damerau, and Robert L. Mercer. Context based spelling correction. *Information Processing & Management*, 27(5):517 – 522, 1991.

[6] Lou Burnard and Tony McEnery. *Rethinking Language Pedagogy from a Corpus Perspective*. Peter Lang, Bern, Switzerland, 2000.

[7] L. O'Hearn, Graeme Hirst, and Alexander Budanitsky. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. pages 605–616, 02 2008.

[8] Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87–111, 2005.

[9] Jennifer Pedler and Roger Mitton. A large list of confusion sets for spellchecking assessed against a corpus of real-word errors. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta,

Malta, May 2010. European Language Resources Association (ELRA).

[10] L. Amber Wilcox-O'Hearn. Detection is the central problem in real-word spelling correction. *CoRR*, abs/1408.3153, 2014.

[11] Aurélien Max and Guillaume Wisniewski. Mining naturally-occurring corrections and paraphrases from Wikipedia's revision history. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[12] Torsten Zesch. Measuring contextual fitness using error contexts extracted from the Wikipedia revision history. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538, Avignon, France, April 2012. Association for Computational Linguistics.

[13] Pratip Samanta and Bidyut B. Chaudhuri. A simple real-word error detection and correction using local word bigram and trigram. In *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013)*, pages 211–220, Kaohsiung, Taiwan, October 2013. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

[14] Andrew R. Golding and Dan Roth. A winnow-based approach to context-sensitive spelling correction. *CoRR*, cs.LG/9811003, 1998.

[15] Sumit Sharma and Swadha Gupta. A correction model for real-word errors. *Procedia Computer Science*, 70:99–106, 12 2015.

[16] Olatokunbo S. Fashola, Priscilla A. Drum, Richard E. Mayer, and Sang-Jin Kang. A cognitive theory of orthographic transitioning: Predictable errors in how spanish-speaking children spell english words. *American Educational Research Journal*, 33(4):825–843, 1996.

[17] S. Singh and S. Singh. Review of real-word error detection and correction methods in text documents. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1076–1081, March 2018.

[18] Davide Fossati and Barbara Di Eugenio. I saw tree trees in the park: How to correct real-word spelling mistakes. In *LREC*, 2008.

[19] Roger Mitton. *English spelling and the computer*. Longman, 1996.

[20] Lawrence Philips. Hanging on the metaphone. *Computer Language Magazine*, 7(12):39–44, December 1990. Accessible at `http://www.cuj.com/documents/s=8038/cuj0006philips/`.

[21] Jennifer Pedler. *Computer correction of real-word spelling errors in dyslexic text*. PhD thesis, Birkbeck, London University, 2007.

[22] Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Ged Ellis. Using the Web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899, Singapore, August 2009. Association for Computational Linguistics.