Revisiting Known-Item Retrieval in Degraded Document Collections

Jason Soo and Ophir Frieder; Information Retrieval Lab; Georgetown University; Washington, DC: {soo;ophir}@ir.cs.georgetown.edu

Abstract

Optical character recognition software converts an image of text to a text document but typically degrades the document's contents. Correcting such degradation to enable the document set to be queried effectively is the focus of this work. The described approach uses a fusion of substring generation rules and context aware analysis to correct these errors. Evaluation was facilitated by two publicly available datasets from TREC-5's Confusion Track containing estimated error rates of 5% and 20% . On the 5% dataset, we demonstrate a statistically significant improvement over the prior art and Solr's mean reciprocal rank (MRR). On the 20% dataset, we demonstrate a statistically significant improvement over Solr, and have similar performance to the prior art. The described approach achieves an MRR of 0.6627 and 0.4924 on collections with error rates of approximately 5% and 20% respectively.

Introduction

Documents that are not electronically readable are increasingly difficult to manage, search, and maintain. Optical character recognition (OCR) is used to digitize these documents, but frequently produces a degraded copy. We develop a search system capable of searching such degraded documents. Our approach sustains a higher search accuracy rate than the prior art as evaluated using the TREC-5 Confusion Track datasets. Additionally, the approach developed is domain and language agnostic; increasing its applicability.

In the United States there are two federal initiatives underway focused on the digitization of health records.

First, the federal government is incentivizing local and private hospitals to switch from paper to electronic health records to improve the quality of care [3]. Second, the Veteran's Affairs (VA) has an initiative to eliminate all paper health records by 2015 [2]. Both processes require converting paper records to digital images, and – hopefully – indexing of the digitized images to support searching. These efforts either are leveraging or can leverage OCR to query the newly created records to improve quality of service.

These are but a few of the many examples demonstrating the importance of OCR.

An OCR process is composed of two main parts. First is the conversion of an imagine to text by identifying characters and words from images [8, 17]. Second, the resulting text is post-processed to identify and correct errors during the first phase. Techniques in this process can range from simple dictionary checks to statistical methods. Our research focuses on the latter phase.

Some work in the second phase has attempted to optimize the algorithm's parameters by training algorithms on portions of the dataset [16]. However, such an approach does not generalize to other OCR collections. Other work focuses on specialized situations: handwritten documents [15]; signs, historical markers/documents [13, 9]. While other works hinge on assumptions: the OCR exposes a confidence level for each processed word [7]; online resources will allow the system to make hundreds-of-thousands of queries in short bursts [6, 12]; or the ability to crawl many web sources to create lexicons [28]. We focus on the generalized case of post-processing of OCR degraded documents without training or consideration of document type.

Historically, there was a flurry of research in this area, particularly around the time TREC released an OCR corrupted dataset [10]. Entries to the TREC competition fell into 2 categories: attempts to clarify or expand the query and attempts to clarify or correct the documents themselves. Results submitted from the latter category have higher mean reciprocal ranks (MRR). Therefore, we continue work in this direction.

Taghva et al. published many results in this area [26]. They have designed specialized retrieval engines for OCR copies of severely degraded documents [25] and found their tested OCR error correction methods had little impact on precision/recall vs an unmodified search engine [24]. This result suggests that Solr is a good enough solution to searching OCR corrupted collections. Their most related work to this research was the creation of a correction system for OCR errors. This system uses statistical methods to make more accurate corrections, but requires user training and assistance [27]. More recent work from this lab has been focused on similar supervised approaches [18]. In contrast, our objective is the development of a solution requiring no user intervention or training data.

Our contributions are:

- Given a minimally corrupted dataset ($\sim$5% error rate), we show that a fusion based method has a statistically significantly ($p<0.05$) higher MRR than prior art, and higher MRR than individual methods for correcting corrupt words.
- Given a moderately corrupted dataset ($\sim$20% error rate), we show the same method's MRR is roughly equal to the prior art's.
- We evaluate the impact of context when correcting corrupted terms in a corrupted document.
- We demonstrate the tradeoffs of occurrence frequency thresholds for corrupt words. Thresholds set too high and too low negatively impact MRR.
- We evaluate filtering methods to increase the accuracy of identifying corrupt words.
- We reinforce the assumption that use of domain keywords

improve correction rates by showing their impact on MRR.

Methods

Dataset

Document Set

The experiments performed are based on the publicly available TREC-5 Confusion Track collection: 395 MB containing approximately 55,600 documents. The documents are part of the Federal Register printed by the United States Government Printing Office. A list of 49 queries and the best resulting document are provided for evaluation. Since each query seeks only a single document, MRR is reported. TREC created two corrupted datasets from the original collection with an estimated 5% error rate and 20% error rate.

Real Words Dictionary

We create an exhaustive English dictionary of real words using the following three datasets: 1) 99,044 words from the English dictionary[1]; 2) 94,293 sir names in the United States[2]; 3) 1,293,142 geographic locations within the United States[3]. Collectively, this dictionary is referred to as *real words*.

To measure the impact of a domain specific dictionary, we supplement the *real words* dictionary with additional terms obtained from the 1996 Federal Register [1]. By selecting the publications from 1996 – 2 year after our test set – we ensure minimal possible overlap of temporal topics. To accurately attribute the impact of these domain terms, we report our results both with and without this dataset.

Baselines

We used Solr to construct a baseline measure. Using version 4.6.1, we indexed the 5% and 20% collections with the system defaults. We ran the 49 queries against these indexes and measured the MRR to obtain a baseline. As described earlier, using the findings from [24] – a multi-year study of searching collections with OCR errors – allows us to assert Solr is roughly equal to their studied OCR error correction methods.

More directly, the best reported MRRs on the TREC-5 Confusion Track (5%: 0.5737; 20%: 0.4978) are by the Swiss Federal Institute of Technology (ETH for short) whose method attempts to correct corrupted words in the document by replacing each corrupted term by a vector of candidate correction terms [5]. Candidate terms for the vector are chosen using a probabilistic technique that estimates feature frequencies. The result of this method is the expansion – and hopeful clarification – of a corrupted document. Our method differs from the ETH method in the retrieval strategy for estimating the likelihood a corrupted term maps to a candidate term.

We considered implementing the ETH method on our version of Solr, but their approach is based upon a customized search engine known as SPIDER [5]. Therefore, without the intimate knowledge of their parser, tokenizer, and other low-level mechanisms, we cannot hope to accurately reproduce their process.

[1] ftp://ftp.gnu.org/gnu/aspell/aspell-0.60.6.1.tar.gz

[2] http://www.census.gov/genealogy/www/data/1990surnames/dist.all.last

[3] http://download.geonames.org/export/dump/US.zip

Experiments

We now describe the experiments performed on different components of the post-processing phase. To evaluate the impact of each experimental change, we issue 49 queries associated with our TREC dataset, and measure the changes in MRR.

Filtering

The filtering component is tasked with separating real words from the OCR errors. Using our *real words* dictionary, we can define *corrupted words* as the relative compliment of *real words*. We evaluate methods to reduce the noise within *corrupted words*, using the following heuristics constructed from empirical observation:

- Drop numerics from words that lead with at least one numeric followed by all characters or lead with at least one character followed by all numerics. Often whitespace or the initial/final characters of a word may be interpreted as or combined with numerics. Stripping such numerics reduces noise.
- Drop words that are less than 4 characters, these are typically noise and correcting them is of little use.
- Drop words that match the stem of a *real word*, so that we do not attempt to correct a real-word.
- Drop words that do not contain any characters, since we do not concern ourselves with attempting to numeric OCR errors.
- Drop words that are below a frequency threshold, because a corrupted term with an low frequency will provide little value if corrected. Furthermore, the likelihood of successfully correcting such a term is much lower than the likelihood of selecting an incorrect word, causing document drift.

Correcting

The correcting component is the core of post-processing. This experiment seeks to identify a reliable method for correcting the identified OCR errors. The process works as follows. For each *corrupted word*, we use a correction method to obtain a candidate's ranked-list. We then select the top k (also called TopK) from each list and replace all instances of the corrupted word in the document set with this list. Once all corrupted words are "corrected", we measure and record results, repeating this process for the each correction method.

Furthermore, we take the best performing method and evaluated its performance as we vary how its correction candidates are selected. For example, the fusion method makes suggestions first without considering the surrounding words (context-free), then makes suggestions which take into account the surrounding words (context-dependent). The suggests from these two sources are later merged. We evaluate what happens when we adjust the size of each set. This gives us an understanding of the role context plays in the OCR post-processing task.

Domain Knowledge

To understand the impact of domain keywords, we supplement the *real words* dictionary used in by the correction approaches with terms from the 1996 Federal Register and then measure the change in MRR.

Frequency Threshold

Finally, we measure the change in MRR when varying our frequency threshold for corrupted words. By raising the threshold, we have less words to correct, but may miss an opportunity to

```
if q ∈ I then
    return q;
end
κ ← ∅;
for i ∈ {1..6} do
    κ ← append(κ, r_i);
end
H ← ∅;
for s ∈ κ do
    Search I for s;
    if H[s] == ∅ then
        H[s] ← Φ(r);
    else
        H[s] ← H[s] + Φ(r);
    end
end
if C(H) < 0.3 then
    N ← n-grams result set;
    if C(H) < C(N) then
        H ← N;
    end
end
return H;
```

Algorithm 1: Segments Process

correct true-positive OCR errors. We report the impact of this varying threshold.

Evaluated Correction Approaches

We evaluated five retrieval strategies for finding and ranking correction candidates for corrupted words. These strategies are:

1. Segments
2. Segments re-ranked using edit distance
3. Word-level bigrams using Segments
4. Word-level bigrams using n-grams
5. Fusion of Segments and word-level bigrams using Segments

Each of these methods takes a corrupted word and returns a ranked-list of candidates. These approaches evaluate many styles. Segments and edit distance evaluate substring rules, bigrams evaluate context, and the fusion approach evaluates the combination of the two.

Segments

For completeness, Segments is a system that takes an input string, and using 6 substring rules, returns a list of possible correction candidates derived from a lexicon, ranked by similarity. A detailed Segments description is found in [21, 22, 20, 23]. Recent research has reaffirmed the potential of segmenting strings by using said segments to perform authorship attribution [19]. Our approach implicitly considers similar string segments.

Throughout this paper, $\Theta(q)$ will run the Segments process on the input string q, and return a ranked-list of candidates.

To begin, Segments takes an input string q. If an exact match of q exists in our lexicon, we return it, otherwise we generate substrings of q, by applying 6 rules, 2 of which are recursive (r_5 and r_6). These rules are described using the following for the sake of convenience:

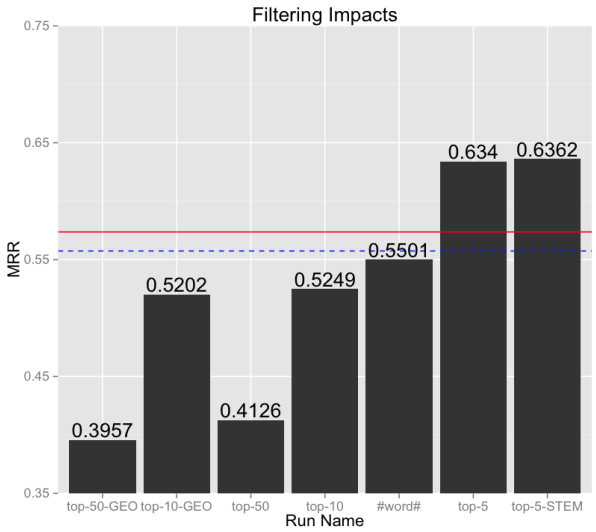

Figure 1. *MRR of filtering methods applied using the 5% dataset.*

- λ represents the length of string q
- $q[0..5]$ returns the zeroth to fourth characters in q
- r_i will immediately return if $\lambda \leq 3$
- $\delta := 0$ represents an assignment of 0 to the variable δ. δ's scope is limited to the rule where it is declared.
- Cases where a floor or ceiling may be required are ignored
- A semicolon is used in place of a line break
- $*$ represents a wildcard of zero or more characters

Using these conveniences, we present the descriptions of each rule below.

1. $r_1(q) =$ return $q[0..\lambda/2]*$
2. $r_2(q) =$ return $*q[(\lambda/2)+1..\lambda]$
3. $r_3(q) =$ return $*q[2..\lambda-2]*$
4. $r_4(q) =$ return $*q[1..\lambda-1]*$
5. $r_5(q) = \delta := q[0..(\lambda/2)-1]*q[(\lambda/2)+1..\lambda]; f_5(\delta);$ return δ
6. $r_6(q) = \delta := *q[1..\lambda-1]*; f_6(\delta);$ return δ

These substrings form the set κ. For each $s \in \kappa$, search the lexicon and add each hit to a hashset H. H is keyed on hit strings and has a similarity score float value. Each time an $s \in \kappa$ hits a string, the string is added to H and assigned a value using the following similarity score function: $\Phi(r) = 1/(r+1)$, where r is the number of recursive function calls required to generate s from q. If the string already exists in H, the value is simply incremented by $\Phi(r)$. For example, if a substring rule $s \in \kappa$ was generated from $f_1(q)$, that implies $r = 0$ since no recursive calls to $f_1(q)$ were required to generate s (after all, $f_1(q)$ is not a recursive function), thus any candidate found by s will add a value of 1 ($\Phi(0) = 1$) to the corresponding entry in H. This puts greater weight on candidates found when the $s \in \kappa$ closely resemble q, and decays the weight as the substrings resemblance to q fades.

Next, H is ordered by similarity scores and a confidence is computed:

$$C(H) = \frac{max_{c \in H}\{c.value\}}{\sum_H value} \quad (1)$$

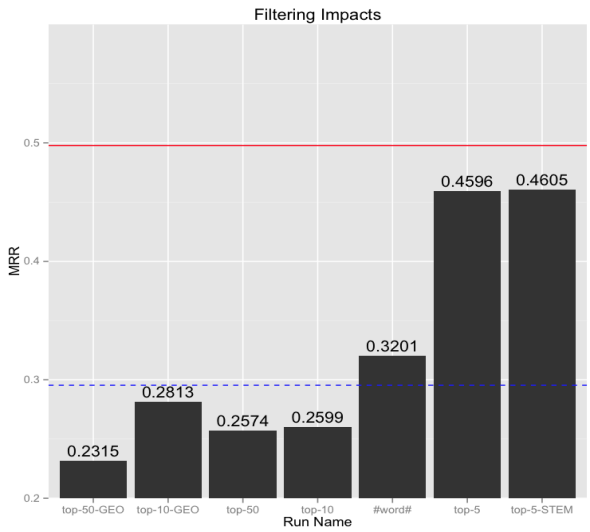

Figure 2. *MRR of filtering methods applied using the 20% dataset.*

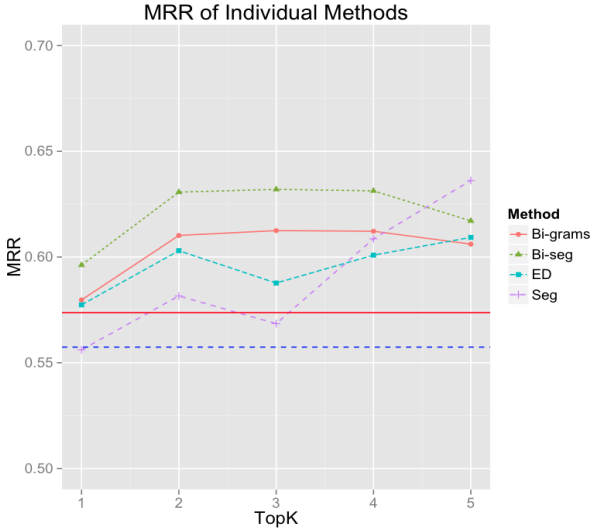

Figure 3. *MRR of individual methods evaluated with a frequency threshold of 4 or more using the 5% dataset.*

If $C(H) \geq 0.3$ – suggesting a tight distribution of votes around a single term – return the ordered list. Otherwise, the list has a high variance, so we do a final check with vanilla 3-grams. Using a standard 3-grams search process, we order the list of found candidates, and return the result set with higher confidence.

Segments Re-Ranked Using Edit Distance

This method works the same as the previously described method, but will re-rank the result set using edit distance.

Word-level Bigrams Using Segments

This approach uses a sliding window – inspired by an approach used in [14] for passage detection – to select which bigrams to use for in the correction process. For each word in our *corrupted words* set, we find the word that occurs before, and the word that occurs after the corrupted word (if any). We do not cross line feed boundaries, but do cross punctuation boundaries since OCR errors often introduce punctuation (for example, "|)" for D).

Each bigram contains a known corrupted word w_c, and the word directly preceding w_p and follow it w_f, if any. Then, using a list of English bigrams from Wikipedia W [4], we return the following bigrams set: $\forall w_i \in \Theta(w_c), \{w_p, w_i\} \in W \cup \{w_i, w_f\} \in W$, where w_i is a correction candidate for w_c as suggested by Segments. These bigrams are ranked by the hit count. Finally we remove w_p and w_f

Note, it is possible that w_p or w_f are corrupted. We make no attempts to correct for this, and simply allow such occurrences to fail to find any matches.

Word-level Bigrams Using 3-grams

This process works in the same way as the previously described process, but with 3-grams for candidate retrieval in place of Segments. This test is included to evaluate the usefulness of the Segments substring rules.

Fusion Approach

Previous research has shown that fusion approaches can be quite powerful [11]. Therefore we sought to evaluate this approach within the corrupted document domain. These methods merge the result sets from two previously independent methods: either 1) Segments results and word-level bigrams using Segments results, or 2) Segments re-ranked using edit distance results and word-level bigrams using Segments results.

Results

In the results figures, where present, horizontal dashed and solid lines represent Solr's baseline of 0.5574 and 0.2954 and ETH's score of 0.5737 and 0.4978, on the 5% and 20% datasets, respectively.

The highest reported MRR from all measured approaches is 0.6627, reported by the "FR-Fusion" with a candidate vector size of 2. The statistical t-test was used to verify the increase in MRR is statistically significant ($p<0.05$).

Filtering

Figure 1 shows the results of the filtering approaches applied to the 5% dataset, and Figure 2 show the same filtering applied to the 20% dataset. The approaches evaluated are, from left to right: top 50 suggestions including the geography dictionary; same as the previous, but with top 10 suggestions; top 50 suggestions without the geography dictionary; same as previous with top 10 suggestions; same as previous, but all corrupted words with leading or trailing numbers are prepended to the suggestions list with those numbers removed; same as previous, but with top-5 suggestions only; same as previous, but with corrupted words that match a stemmed version of a word in our *real words* dictionary prepended to the suggestions list. These results were processed using a frequency threshold ≥ 4.

From Figure 1 and Figure 2, we can see that filtering is important. Prior to filtering, the *corrupted word* set contained 329,346 entries. After setting a frequency threshold of 4 (which aligns with the 3rd quartile), we were left with 82,953 entries. Af-

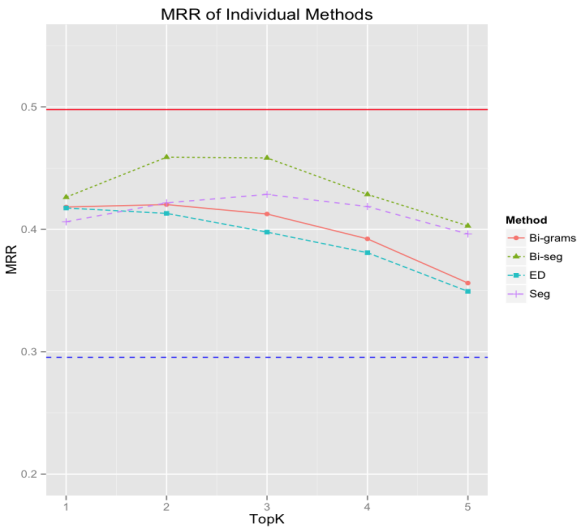

Figure 4. *MRR of individual methods evaluated with a frequency threshold of 4 or more using the 20% dataset.*

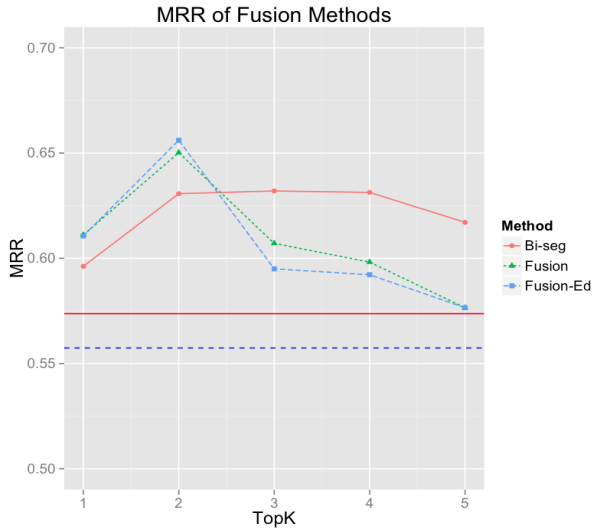

Figure 5. *MRR of fusion methods evaluated with a frequency threshold of 4 or more using the 5% dataset.*

Table 1: Impact of varying sets for fusion method on the 5% dataset

Context-Free Suggestions	Context-Dependent Suggestions	MRR	Rank
2	3	0.6627	1
1	2	0.6577	2
2	2	0.6481	3
3	2	0.6419	4
2	1	0.6330	5
0	3	0.6320	6
0	2	0.6307	7
1	3	0.6289	8
1	1	0.6221	9
3	3	0.6100	10
2	0	0.6093	11
3	1	0.6003	12
0	1	0.5962	13
3	0	0.5943	14
1	0	0.5783	15

Table 2: Impact of varying sets for fusion method on the 20% dataset

Context-Free Suggestions	Context-Dependent Suggestions	MRR	Rank
2	3	0.4924	1
2	2	0.4885	2
3	2	0.4864	3
2	1	0.4761	4
3	1	0.4698	5
3	3	0.4634	6
2	0	0.4601	7
0	2	0.4540	8
1	3	0.4460	9
3	0	0.4356	10
1	2	0.4293	11
1	1	0.4278	12
0	1	0.4109	13
0	3	0.4087	14
1	0	0.4042	15

ter all filtering was complete, our *corrupted word* size was 44,640. For the 20% dataset, 285,572 entries were reduced to 143,385. During this reduction, we learned several things.

First, as shown by top-50-GEO and top-10-GEO, the geography set was detrimental to our suggestions because it is over 6.5x larger than the other two (dictionary and names) sets combined. This means that corrupt words were significantly more likely to have their substrings match an entry in the geography set. Once we removed the geographic component of our dictionary, our MRR was improved, as shown by the 4 rightmost columns.

Second, the number of suggestions is very important. Replacing a corrupted word with a large word vector causes the document's content to drift, as shown by comparing top-50 and top-10. The larger the vector, the more pronounced the drift becomes. A small vector performs better. Of all the filtering, this seems to be the most important.

Third, it was observed that there were many corrupted words of the form "1depart" or "depart1". The "#word#" column in Figure 1 shows that by prepending the vector with the corrupted word without any digits, we increased our MRR by 0.0252. Note that there were no checks performed to see if the corrupted word appeared in our *real words* dictionary after the transformation.

Last, a small subset of the *corrupted words* were noted to simply be stemmed versions of words appears in our *real words* set. By stemming our *real words* set and removing the intersection with the *corrupted words*, we were slightly increased again. Top-5-STEM shows this impact.

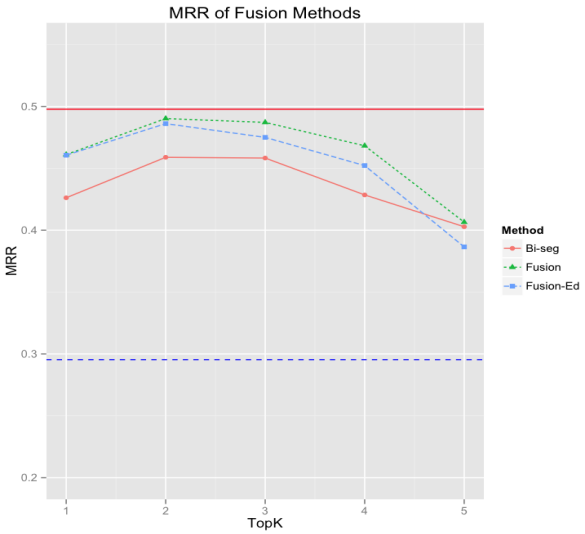

Figure 6. *MRR of fusion methods evaluated with a frequency threshold of 4 or more using the 20% dataset.*

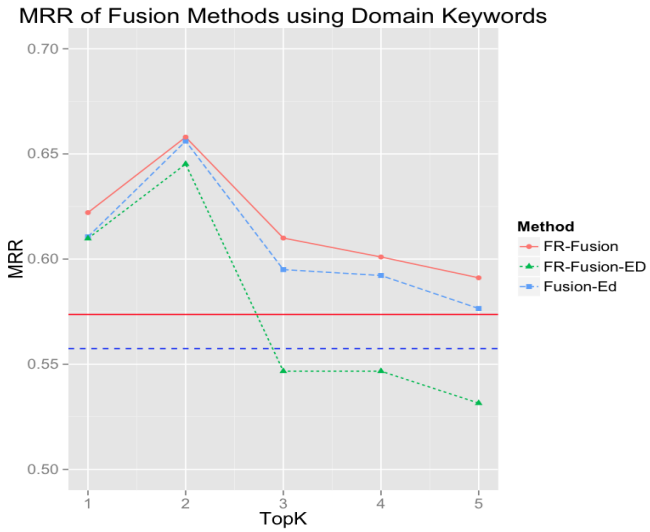

Figure 7. *Impact of including domain specific keywords in our lexicon using the 5% dataset.*

Methods

After selecting top-5-STEM for filtering, we evaluated our individual methods in Figure 3 and Figure 4. The results of the Segments (Seg); Segments re-ranked using edit distance (ED); word-level bigrams using Segments (Bi-Seg); and word-level bigrams using n-grams (Bi-gram) are shown.

Segments

As demonstrated in Figure 3, unlike other methods, the Segments tail begins to rise, with its performance at k = 5. This performance is better than any other *individual* method. This suggests that Segments is the best at retrieving candidates, but has a poor ranking function. With a non-trivial portion of the correct terms residing at the 5th position of the vector, an improved ranking function could make Segments a powerful approach. However, Segments's ability to rank is limited by its lack of context knowledge. When context is important, Segments performs poorly [20]. Once k = 5, the ranking becomes less important (which Segments is poor at), and selection becomes more important (which Segments is good at). This realization supports our hypothesis that while context is important, substring selection is also important to good performance, as demonstrated by the k = 5 Segments results.

The 20% dataset results shown in Figure 4 show Segments maintain roughly stable results. This is to be expected, as the increased corruption doesn't impact it as dramatically as other approaches, as it has no reliance on contextual words which may also be corrupted.

Segments with Edit Distance

Since Segments performs the initial selection process of the top 50 candidates for a corrupted word, this method has similar performance to Segments. It is infeasible to have edit distance make such an initial selection due to its time complexity – optimistically approximated by $O(n^2)$ on a typical implementation, where n is the size of the dictionary. Edit distance will rank words based on their integer value scores. However, in our work we found that many more than 5 words per corrupted word received an edit distance score of 1. At such a point, the question becomes how to rank these words. No consideration is given to the old word structure. Instead, all characters are considered equally in transformation operations which ultimately cannot be overcome. **Bigrams** The bigrams method has consistently good results on the 5%. This is to be expected as context is quite powerful. For example, one observed corrupted word was *prodect*. In this scenario, Segments suggested the following 5 candidates: *PROJECT, PROTECT, PROSPECT, PROVECT, PRODUCT*. Without context, all of these are valid candidates.

As seen in Figure 3, the comparison of Bigrams using Segments to retrieve candidates (Bi-seg) and Bigrams using 3-grams to retrieve candidates (Bi-grams) demonstrates the power of Segments of a regular implementation of n-grams, as Segments consistently outperforms the retrieval of 3-grams. The performance of our bigrams method further supports our hypothesis that, not only are good substring rules important in correcting OCR errors (as shown by Segments at k = 5), but so is context.

Another supporting result to our hypothesis is the bigram results on the 20% dataset shown in Figure 4. The bigrams approach has a quick decline due to the role of corruption in the contextual words. Thus, there is a need for correction methods to select their candidates in a context-free fashion, as Segments does.

Fusion

Figure 5 shows the results of our fusion methods on the 5% dataset and Figure 6 shows the similar results using the 20% dataset. Both figures include the best performing individual method for comparison. In this figure, we show bigrams using Segments (Bi-Seg); Fusion method of bigrams and Segments (Fusion); and Fusion method of bigrams and Segments re-ranked using edit distance (Fusion-Ed). Each method was evaluated on the same dataset with all 49 queries, using a corrupted word frequency threshold of at least 4.

On both datasets, the fusion methods have the best performance of all the methods at k = 1 and k = 2. However, this is slightly misleading. By definition, the fusion methods combine the results of other methods. In this case, it combines the vectors of Segments fused with bigrams (Fusion), or Segments re-ranked with edit distance fused with bigrams (Fusion-Ed). At k = 1, the

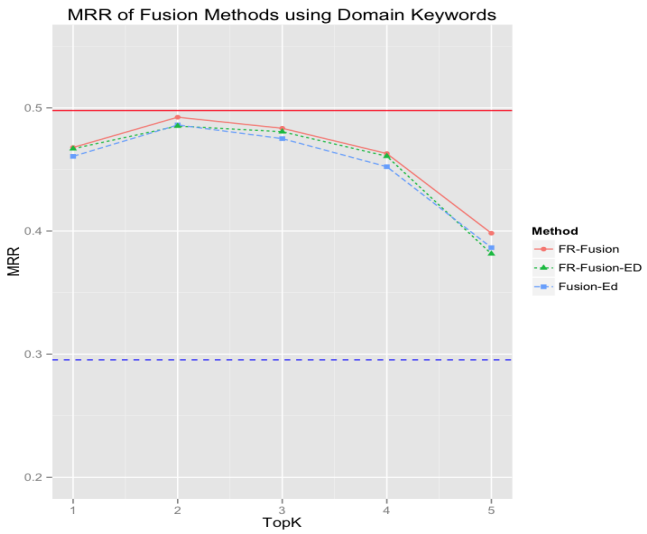

Figure 8. *Impact of including domain specific keywords in our lexicon using the 20% dataset.*

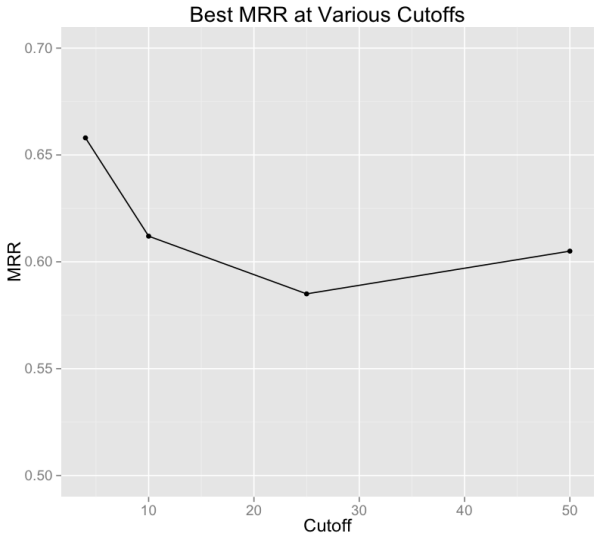

Figure 9. *Impact of frequency thresholds on MRR using the 5% dataset.*

fusion methods combine the top candidate as suggested by Segments and bigrams. This actually results in, at most, 2 suggestions (there is no guarantee the candidates are unique). Similarly, for k = 2, the **top 2 candidates** from Segments and top 2 from bigrams are merged and used by the fusion method. It is for this reason you see a dramatic decline in MRR at k = 3 for the fusion methods – they actually contains, at most, 6 suggested candidates. This is where document drift comes in as we discussed earlier. The trend continues for the remainder of the graphs.

An interesting observation on the 5% dataset is that when taken individually, Segments re-ranked with edit distances outperforms just Segments for low values of k. However, when these two individual methods are combined into a single method in the fusion methods, they are roughly equivalent, with one having a slightly higher MRR than the other at various values of k. This can be attributed to the impact of the bigrams suggestions. For

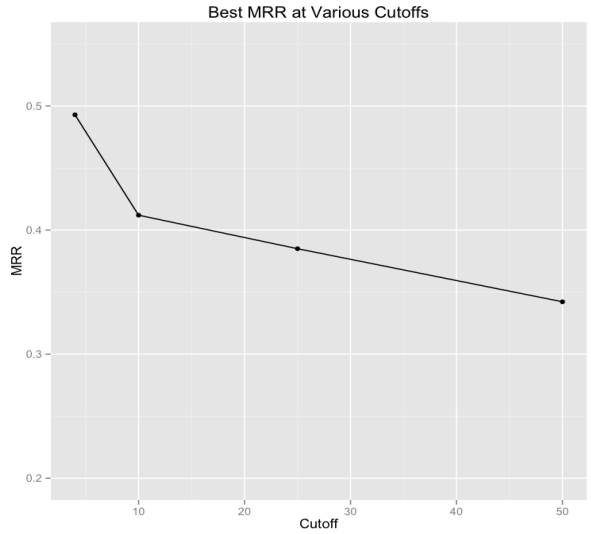

Figure 10. *Impact of frequency thresholds on MRR using the 20% dataset.*

higher values of k, the Segments method outperforms the edit distance method, following the same trend as when they are used individually.

Similar trends can be found in Figure 6 for our 20% dataset. However, one difference here is that all of the fusion methods follow a very similar trend. We believe this is because of the difficulty in trying to make corrections in a document with a 20% error rate.

A final support of our hypothesis is offered at k = 1 and k = 2 of the fusion methods. The combination of substring rules and context results in superior performance as compared to other methods. Eventually, the tail drops off but is still better than each individual method measured at k = 6, which would be an apt comparison.

A consideration for future research is the selection process for the fusion methods. In our work we simply merged the two results, as we didn't want to overfit our approach to our dataset, but instead observe overall trends. However, a more intricate selection process that considers rules, substring matches, frequency analysis, and other attributes may likely perform better.

Context-Dependent VS Context-Free

When correcting OCR errors, with reasonable expectation, the terms surrounding the corrupted term may also be corrupted. This degrades the performance of context-dependent approaches, such as word-level bigrams and trigrams, and leads one to believe context-free corrections may perform better in such an environment. Table 1 and 2 explore this by varying the size of candidates from context-free and context-dependent sets used by our fusion method. First, the 5% collection, which has less opportunity for contextual terms to be corrupted, performs well with a larger set of context-dependent corrections. The top 4 results all heavily rely on the context-dependent set. Furthermore, the worst 5 MRR in this experiment is when the fusion method uses 1 or less correction terms from the context-dependent set.

The results of the 20% collection, shown in Table 2, suggest a slightly different trend. While still strong in combination, heavily context-free suggestion results outperform the context-

dependent results. This upholds our hypothesis that, when faced with corrupted documents, a balance between context-free and context-dependent corrections will yield good results.

Domain Knowledge

Figure 7 and Figure 8 show the impact of using domain keywords. The methods evaluated using these keywords are: Segments fused with bigram results (FR-Fusion) and Segments reranked with edit distance and fused with bigram results (FR-Fusion-ED). We also include the best performing fusion method without knowledge of domain keywords for comparison (Fusion-ED).

By indexing keywords from a collection of documents published from the same domain two years after our test collection, Figure 7 and Figure 8 show that overall performance increased. The natural suggestion from these results is that lexicon is important. However, we see that the maximum MRR is only improved slightly by the fusion method that uses keywords (FR-Fusion) above the method that does not use keywords specifically derived from this domain (Fusion-ED). This increase is more pronounced for other k.

Frequency Thresholds

Should corrupt words that appear only a handful of times be kept, or discarded? We evaluate this question by applying the best performing method, "FR-Fusion" at k = 2, to other frequencies, and measure the MRR. As shown in Figure 9, we find that results degrade as more terms are dropped. As the frequency threshold moves above 4, the MRR decreases. with an odd, but still lower, result at a frequency of 50. Similar degregation as the threshold increases can be see for our 20% dataset results in Figure 10. Here we see a monotonic decline, likely due to the number of split words that resulted from the increase error rate

Limitations

We would be interested to compare the performance of our work not just to prior academic research, but to commercial offerings. We are not able to evaluate any modern OCR offerings because images of the original paper documents from the TREC-5 Confusion Track that were used for scanning are not available. Furthermore, we are interested to see how well a web-scale system performs. For example, Google has query logs from which they can learn, infer context, and perform other useful actions. Ideally we would submit word-level 3-grams to Google as a query, and then use their "did you mean" correction feature to fix corruption. Wikipedia has a similar feature. However, due to bandwidth and throttling, we were unable to submit hundreds-of-thousands of queries.

Conclusion

This work set out to evaluate and find methods for reliably correcting OCR errors. Using two publicly available datasets, we evaluated methods that operate on substrings, consider context, and a fusion of both. Our fusion method's MRR of 0.6627 on the 5% dataset statistically significantly outperforms the prior art's MRR of 0.5737 and Solr's MRR of 0.5574. Furthermore, our MRR of 0.4924 on the 20% dataset statistically significantly outperforms Solr's MRR of 0.2954 and has similar performance to the prior art's MRR of 0.4978. These findings support our hypothesis that both context and substring rules are important to correcting OCR errors, and therefore a fusion method is the best approach.

Our findings also suggest that the ideal size of candidate vectors to replace a corrupted word is between 2 and 5, low-frequency words are valuable, and that keywords from the domain of the document collection can further increase the MRR.

We believe future work in this area should focus on fusion based methods.

References

[1] 1996 United States Federal Register Publications. http://www.gpo.gov/fdsys/pkg/FR-1996-[MONTH]-[DAY].zip.

[2] America's vets: Returning home to a broken system. http://www.npr.org/2013/05/26/186710926/americas-vets-returning-home-to-a-broken-system.

[3] Feds unveil incentive plan for electronic medical records. http://www.npr.org/blogs/health/2009/12/hhs_lays_out_standards_for_ele.html.

[4] Wikipedia Bigram Open Datasets. https://github.com/rmaestre/Wikipedia-Bigram-Open-Datasets/blob/master/datasets/bigram_EN.dat.gz.

[5] J. P. Ballerini, M. Büchel, R. Domenig, D. Knaus, B. Mateev, E. Mittendorf, P. Schäuble, P. Sheridan, and M. Wechsler. SPIDER Retrieval System at TREC-5. In *TREC*, 1996.

[6] Y. Bassil and M. Alwani. OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion. *Journal of Emerging Trends in Computing and Information Sciences*, 3:90–99, 2012.

[7] S. Chen, D. Misra, and G. R. Thoma. Efficient automatic OCR word validation using word partial format derivation and language model. In *Document Recognition and Retrieval XVII*, pages 75340O–1 – 75340O–8, 2010.

[8] R. Clawson, K. Bauer, G. Chidester, M. Pohontsch, D. A. Kennard, J. Ryu, and W. Barrett. Automated recognition and extraction of tabular fields for the indexing of census records. In *Document Recognition and Retrieval XX*, 2013.

[9] D. Garrette, H. Alpert-Abrams, T. Berg-Kirkpatrick, and D. Klein. Unsupervised code-switching for multilingual historical document transcription. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1036–1041, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[10] P. B. Kantor and E. M. Voorhees. The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text. *Information Retrieval*, 2:165–176, 2000.

[11] B. Larsen, P. Ingwersen, and B. Lund. Data fusion according to the principle of polyrepresentation. *Journal of the American Society for Information Science and Technology*, 60(4):646–654, 2009.

[12] Y. Li, H. Duan, and C. Zhai. CloudSpeller: Spelling Correction for Search Queries by Using a Unified Hidden Markov Model with Web-scale Resources. In *Special Interest Group on Information Retrieval*, pages 0–4, 2011.

[13] W.-s. W. Lian. *Heuristic-Based OCR Post-Correction for Smart Phone Applications*. PhD thesis, University of North Carolina at Chapel Hill, 2009.

[14] S. Mengle and N. Goharian. Passage Detection Using Text Classification. *Journal of the American Society for Information Science and*

Technology, 60(4):814–825, Apr. 2009.

[15] N. Naji and J. Savoy. Information Retrieval Strategies for Digitized Handwritten Medieval Documents. In M. Salem, K. Shaalan, F. Oroumchian, A. Shakery, and H. Khelalfa, editors, *Information Retrieval Technology*, volume 7097, pages 103–114. Springer Berlin Heidelberg, 2011.

[16] J. Parapar, A. Freire, and A. Barreiro. Revisiting N-Gram Based Models for Retrieval in Degraded Large Collections. In M. Boughanem, C. Berrut, J. Mothe, and C. Soule-Dupuy, editors, *31th European Conference on Information Retrieval Research*, volume 5478, pages 680–684. Springer Berlin Heidelberg, 2009.

[17] J. Parker, O. Frieder, and G. Frieder. Robust Binarization of Degraded Document Images Using Heuristics. In *Document Recognition and Retrieval XXI*, 2014.

[18] S. Poudel. Post processing of optically recognized text via second order hidden markov model. *Masters Thesis, University of Nevada, Las Vegas*, 08 2012.

[19] U. Sapkota, S. Bethard, M. Montes, and T. Solorio. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[20] J. Soo. A non-learning approach to spelling correction in web queries. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 101–102. International World Wide Web Conferences Steering Committee, 2013.

[21] J. Soo, R. Cathey, O. Frieder, M. Amir, and G. Frieder. Yizkor books: a voice for the silent past. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1337–1338. ACM, 2008.

[22] J. Soo and O. Frieder. On foreign name search. In *European Conference on Information Retrieval*, pages 483–494. Springer Berlin Heidelberg, 2010.

[23] J. Soo and O. Frieder. On searching misspelled collections. *Journal of the Association for Information Science and Technology*, 2014.

[24] K. Taghva, J. Borsack, and A. Condit. Evaluation of model-based retrieval effectiveness with ocr text. *ACM Transactions on Information Systems*, 14:64–93, 1996.

[25] K. Taghva, J. Coombs, and I. Science. Hairetes: A search engine for ocr documents. In *In Proc. of 5th Intl. Workshop on Document Analysis Systems, Lecture Notes in Computer Science*, pages 412–422. Springer-Verlag, 2002.

[26] K. Taghva, T. Nartker, and J. Borsack. Information access in the presence of ocr errors. In *Hard Document Processing*, 2004.

[27] K. Taghva and E. Stofsky. Ocrspell: an interactive spelling correction system for ocr errors in text. *International Journal of Document Analysis and Recognition*, 3:2001, 2001.

[28] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. 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, 2009.

Author Biography

Jason Soo is a 4th year PhD candidate at Georgetown University. Jason focuses on language independent spelling correction approaches, searching corrupted documents, and information retrieval in other adverse conditions.

Ophir Frieder holds the McDevitt Chair in Computer Science and Information Processing at Georgetown University. He is also Professor of Biostatistics, Bioinformatics and Biomathematics in the Georgetown University Medical Center. Previously he served as the Chair of the Department of Computer Science. In addition to his academic positions, he is the Chief Scientific Officer for UMBRA Health Corporation. He is a Fellow of the AAAS, ACM, IEEE, and NAI.