Clustering of Short Strings in Large Databases

Michail Kazimianec
Faculty of Computer Science
Free University of Bozen-Bolzano
E-mail: kazimianec@inf.unibz.it

Arturas Mazeika
Department 5
Max-Planck-Institut für Informatik
E-mail: amazeika@mpi-inf.mpg.de

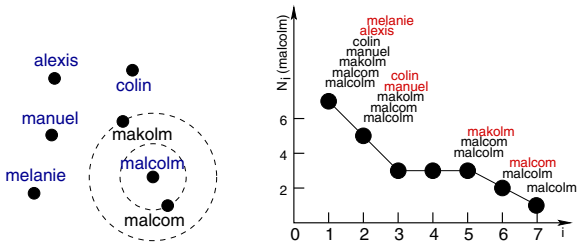
***Abstract*—A novel method CLOSS intended for textual databases is proposed. It successfully identifies misspelled string clusters, even if the cluster border is not prominent. The method uses q-gram approach to represent data and a string proximity graph to find the cluster. Contribution refers to short string clustering in text mining, when the proximity graph has multiple horizontal lines or the line is not present.**

***Index Terms*—clustering, q-grams, short strings**

I. Introduction

Cleansing of string databases and clustering of misspellings is an important, common, and challenging task. Examples of such databases are entered proper names, scrobbling of songs (aggregation of the song titles the user plays), social bookmarking, annotation, and tagging of content. Database records contain many correct strings (proper nouns, song titles, keywords) but also a few arbitrary or systematic misspelled strings. Cleansing and clustering of misspelled strings is of great importance in these databases, since exact search queries return only a part of the results.

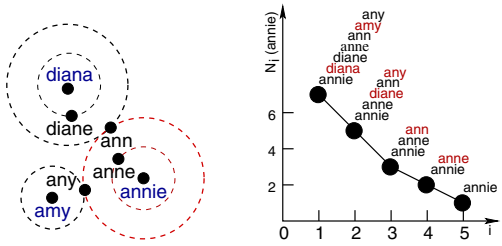
Different methods for cleansing and clustering of dirty string databases exist. Common to these methods is that all of them work only with relatively long strings (longer than 10–15 characters). The clustering quality of these methods decreases as the databases contain strings of the mixed length (both short and long strings) and decay for databases of only short strings. This issue is illustrated in Figures 1–2 for the Proximity Graph Cleansing (GPC) method [1].



(a) String Clustering (b) Proximity Graph

Fig. 1. Clustering with a Prominent Border

GPC determines the clusters of misspellings in the following way. The method takes a center string and examines the size of the neighborhood (the number of strings that are within some similarity threshold from the center string) as the similarity decreases. It is illustrated in Figure 1 for the center string malcolm. N_7(malcolm) in Figure 1(b) indicates the number of strings that matches the center (overlap is 7, i.e. the number of database strings that share 7 2-grams with malcolm), N_6(malcolm) indicates the number of strings that share one 2-gram less, etc. A horizontal line (cf. Figure 1(a)) in the proximity graph (PG) is used by the GPC method to detect the border of the cluster and works well for long strings. However for short strings there are usually no such lines (Figure 2(b)), since the length of the strings is very short compared to the number of misspellings. Nevertheless the data points fall into clearly visible clusters (cf. Figure 2(a)).



(a) Complicated String Clustering (b) Proximity Graph

Fig. 2. Clustering with Unclear Border

The paper contributes with CLOSS (Clustering of Short Strings), a robust clustering method for databases of short strings. The method uses PG to cluster strings, presents a novel border detection algorithm, sorts strings for more accurate results, applies smoothing for PGs with multiple horizontal lines. The border detection algorithm is based on the smallest jump rather than the horizontal line. The ordering of strings increases accuracy for the databases with both long and short strings. The smoothing uniquely and accurately resolves clustering with two or more horizontal lines in PGs. Our analytical and experimental evaluation shows that the method is applicable for both short strings and short and long string databases and outperforms the GPC method.

The paper is organized in the following way. Section II overviews state-of-the-art achievements in cleansing and clustering of textual data. The framework is described in Section III. Section IV presents the CLOSS method. The algorithm is introduced in Section V. Evaluation experiments with real-world data are given in Section VI.

II. Related Work

The methods intended to cluster misspellings refer mainly to specific types of errors and long strings.

GPC is introduced by Mazeika and Böhlen in [1]. It computes string proximity graph for a center string and detects the border based on the horizontal line. It provides good results for strings of similar length and is not applicable for databases of mixed length strings and databases of short strings.

Chen Li *et al.* [2] developed *VGRAM*, a technique to improve the performance of the algorithms that use q-grams for approximate string matching. Specifying q_{min} and q_{max} the method shows good results on datasets with long strings (more than 30 characters long).

The Ed-Join algorithm was proposed by Chuan Xiao *et al.* [3]. It is based on the analysis of mismatching q-grams for edit similarity joins and allows to improve substantially the clustering time for datasets with medium and long strings.

III. Background

In this section we introduce the concepts for the bag of q-grams, overlap between two strings, size of the neighborhood for a center string and overlap threshold, and PG.

The concepts of the **bag of q-grams** and the **overlap** establish the similarity between strings. Let s be a database string. Let $\bar{s}$ be a string obtained by prefixing s with $q-1$ occurrences of '#' and by suffixing s with $q-1$ occurrences of '$'. Then the bag of q-grams of string s is the bag of all substrings of the length q of string $\bar{s}$. For example, the bag of 2-grams for string annie is $B(\text{annie}) = \{\#a, an, nn, ni, ie, e\$\}$. Let s and s' be two strings. The overlap between the strings is the number of q-grams they share in common: $o(s, s') = |B(s) \cap B(s')|$. For example, for $q = 2$ $o(\text{annie}, \text{anne}) = |\{\#a, an, nn, e\$\}| = 4$ and $o(\text{annie}, \text{diane}) = 2$. A higher overlap between two strings denotes higher proximity between them. All database strings that have at least overlap i with a given string s form the neighborhood. The **size of the neighborhood** is denoted by $N_i(s) = |\{s' \in D : o(s, s') \geq i\}|$. Thus in Figure 2(a) the size of the neighborhood with center string $s = \text{annie}$ and overlap threshold $i = 2$ is $N_2(\text{annie}) = |\{\text{annie}, \text{anne}, \text{ann}\}| = 3$.

The **proximity graph** records the decrease in the size of the neighborhood as the overlap threshold i increases. For the center string s PG is defined as a set of pairs: $PG(s) = \{(i, N_i(s)), i = 1, \ldots, \text{length}(s) + q - 1\}$. Figure 2(b) illustrates $PG(\text{annie})$ for the dataset in Figure 2(a). In this case, $PG(\text{annie}) = \{(1, 7), (2, 5), (3, 3), (4, 2), (5, 1)\}$.

IV. CLOSS

In this section we describe a novel approach for cluster border detection that uses the concept of PG jump. We also introduce PG smoothing and sorting of strings to the quality of clustering.

A. Border Detection

We find a cluster border by computing the minimal difference between sizes of the adjacent neighborhoods in PG. The border corresponds to an overlap point i_b that separates PG into a slightly and steeply decreasing part (cf. overlaps 5–3 and 3–1 in Figure 2(b)). The slight decrease is caused by a small number of misspellings that have a high overlap with the correct string, while smaller overlaps include a great number of strings from other clusters and result in a steeper PG decrease. We compute the border in two steps. First, we identify border interval, where the neighborhoods consist of both misspellings and alien strings. Second, within the interval we find the smallest decrease of the neighborhood size.

Border interval $[i', i'']$ is found in two steps: (i) PG is interpolated with a polynomial function (cf. f below), and (ii) the interval is computed using the maximal curvature of f.

Given f the border interval is computed in the following way. Starting point i' is set to the overlap value, where the curvature is maximal

$$i' = \text{argmax}_i\{\mu(i)\} = \text{argmax}_i\Big\{\frac{|f^{(2)}(i)|}{[1 + (f^{(1)}(i))^2]^{\frac{3}{2}}}\Big\}.$$

Here μ is the curvature [4] of f and $f^{(1)}$ and $f^{(2)}$ are the first and second derivatives of f. Ending point i'' is set to be $k \cdot q$ number of q-grams away from the maximal overlap

$$i'' = \text{length}(s) + q(1 - k) - 1,$$

where k defines how many substituted, inserted or deleted characters (mistakes) strings may have if they are to the right of i''. Thus, if $k = 1$ then misspellings beyond the interval have only one mistake.

We use the polynomial

$$f(x) = \sum_{j=0}^{|B(s)|-1} a_j x^j, \tag{1}$$

where coefficients a_j are obtained from the system of linear equations $f(i) = N_i(s)$, $i = 1, \ldots, |B(s)|$.

Example 1: **[Border Interval]** Let us find the border interval for the PG presented in Figure 2(b). The system of equations follows from (1):

$$\sum_{j=0}^{4} a_j i^j = N_i(s),\ i = 1, \ldots, 5,$$

$$N_1(s) = 7, N_2(s) = 5, N_3(s) = 3, N_4(s) = 2, N_5(s) = 1.$$

Solving it we get $a_0 = 6$, $a_1 = 4$, $a_2 = -3.92$, $a_3 = 1$, $a_4 = -0.08$. The curvature for the PG expressed by the polynomial is:

$$\mu(i) = \frac{|2a_2 + 6a_3 i + 12a_4 i^2|}{[1 + \{a_1 + 2a_2 i + 3a_3 i^2 + 4a_4 i^3\}^2]^{\frac{3}{2}}}. \tag{2}$$

Substituting $a_0, \cdots, a_4$ and $i = 2, 3, 4$ into (2) we get $\mu(2) \approx 0.01$, $\mu(3) \approx 0.20$, $\mu(4) \approx 0.10$. We ignore $\mu(1)$ and $\mu(5)$ that correspond to the extreme points of the PG, because it has false curvatures there caused by the polynomial that approximates overlap axis to the left from $i = 1$ and to the right from $i = 5$. Thus $\mu(i') = \mu(3)$ that means $i' = 3$.

Since s = annie is a very short string, we assume $k = 1$ and get $i'' = 4$. Finally, the border interval is $[3, 4]$.

The following definition brings the cluster border $i_b \in [i', i'']$ into correlation with the minimal decrease of the neighborhood size.

Definition 1: **[Cluster Border]** Let $[i', i'']$ be the border interval. Then $i_b \in [i', i'']$ is the cluster border iff

(i) $i_b = \text{argmin}_i\{\Delta_i(s), i = i', i' + 1, \cdots, i''\}$, where $\Delta_i(s) = N_i(s) - N_{i+1}(s) \geq 0$ is the PG neighborhood decrease (PG jump) at the overlap point i;

(ii) $\nexists j < i_b,\ j \in [i', i'']$: $\Delta_j(s) = \Delta_{i_b}(s)$.

Example 2: **[Cluster Border]** For the PG in Figure 2(b) and the border interval $[i', i''] = [3, 4]$ $\min_i\{\Delta_i(s),\ i = 3, 4\} = \Delta_3(s) = \Delta_4(s) = 1$. Then $\Delta_{i_b}(s) = \Delta_3(s) = 1$. Thus $i_b = 3$.

CLOSS is the most effective method when PGs have no horizontal lines. In this case the method identifies i_b correctly enough. A criterion of their absence relates to the property $\Delta_i(s) > 0$, $i = 1, \cdots, |B(s)|$. Supposing conversely that at some point i $\Delta_i(s) = 0$, we get $\Delta_i(s) = N_i(s) - N_{i+1}(s) = 0$, or $N_i(s) = N_{i+1}(s)$. That is, the horizontal line exists if $\Delta_i(s) = 0$ at least in one point.

According to Definition 1 cluster border always exists independently of PG shape. Choosing i_b as the extreme left overlap point of the minimal neighborhood decrease we may add alien strings to the cluster. Therefore it is relevant to know when cluster border corresponds to one minimal PG jump. Sufficient conditions for it are formulated in Theorem 1.

Before that we denote polynomial functions $\Delta^a(x, s)$, $\Delta^m(x, s)$, $\Delta(x, s) = \Delta^a(x, s) + \Delta^m(x, s)$ of the continues variable x such that if $x = i$ then $\Delta(i, s) = \Delta_i(s)$, $\Delta^a(i, s) = \Delta^a_i(s)$, $\Delta^m(i, s) = \Delta^m_i(s)$, where $\Delta^a_i(s)$ and $\Delta^m_i(s)$ are neighborhood decreases of opposite behavior wrt. i (Figure 3) caused accordingly by alien and misspelled strings.

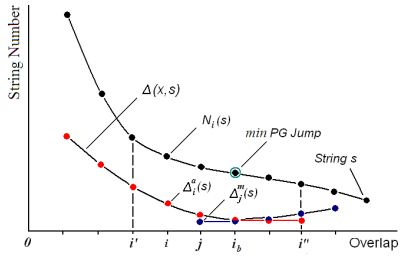


Fig. 3. Neighborhood Functions

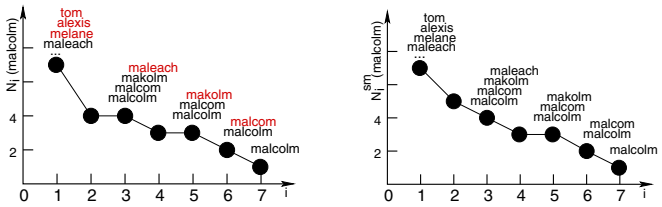
Theorem 1: **[Uniqueness of the Cluster Border]** Let $PG(s)$ be a proximity graph with the border interval $[i', i'']$. Let (i) $\Delta_{i'}(s) > \Delta_{i''}(s)$, and (ii) $\Delta^a(x, s)$ have strictly decreasing and $\Delta^m(x, s)$ have strictly increasing first derivatives in $[i', i'']$. Then $PG(s)$ has one minimal PG jump in $[i', i'']$ that defines the cluster border i_b.

Proof: From Definition 1 follows that cluster border corresponds to extreme left minimal PG jump computed in $[i', i'']$. Let us show that this minimum is only one. Function $\Delta(x, s)$ has local minimum if $d\Delta(x, s)/dx = d\Delta^a(x, s)/dx + d\Delta^m(x, s)/dx = 0$. Since in $[i', i'']$ the first derivatives of $\Delta^a(x, s)$ and $\Delta^m(x, s)$ are strictly monotonic functions, the function modules intersect in one point or do not intersect at all. In the first case minimum of $\Delta(x, s)$ exists in the intersection point and is only one. In the second case $\Delta(x, s)$ is a monotonic function in $[i', i'']$. Since $\Delta(i', s) = \Delta_{i'}(s) > \Delta(i'', s) = \Delta_{i''}(s)$, the minimum of $\Delta(x, s)$ is achieved in the point i''. If $x_{min} = \text{argmin}_x\Delta(x, s)$ is not an integer value within $[i', i'']$ then we choose such i_b that is closest to x_{min}. We choose $i_b = x_i$ if $x_{min} = \frac{x_{i+1} - x_i}{2}$. Thus the minimal PG jump is only one and uniquely defines the cluster border i_b. ■

If the neighborhood decrease functions $\Delta^a(i, s)$ and $\Delta(i, s)$ are not monotonic wrt. i then $\Delta(i, s)$ may have few minimums. This moves the cluster border i_b to smaller overlap points extending the cluster at the expense of alien strings. Moreover, if $\Delta_i(s) = 0$ is true at some overlap points i within $[i', i'']$ then clustering becomes complicated even more, because of the false horizontal intervals that appear in PG. For these cases we propose to identify cluster border by a smoothed PG function.

B. Smoothing

Smoothing replaces PG with the moving averages. This allows to identify the border correctly for PGs with a few horizontal lines that take place in the databases containing short and long strings. The issue is illustrated in Figure 4(a). Here PG has two line intervals: $[2, 3]$ and $[4, 5]$. Overlaps $i = 2$ and $i = 4$ result in the smallest PG jump and overlap $i = 2$ is returned (incorrectly) as the cluster border. The smoothed PG is shown in Figure 4(b). Here the PG values are replaced by moving averages (with $r = 1$ window size) that allows to identify the border correctly ($i = 4$).



(a) PG without Smoothing (b) Smoothed PG

Fig. 4. PG with Horizontal Lines

Horizontal lines appear in PG only if the function $\Delta(i, s)$ has zero values. Depending on their distribution at the overlap axis PGs may differ by line number and length. Theorem 2 formulates conditions for the presence of more than one horizontal line in PG.

Theorem 2: **[Multiple Horizontal Lines]** $PG(s)$ has multiple horizontal lines at least if: $\exists i, j,\ i, j = 1, \ldots, |B(s)|$: $\Delta_i(s) = \Delta_j(s) = 0$, $|i - j| > 2$.

Proof: If there is one $\Delta_i(s) = 0$ then PG has only one horizontal line (see subsection A). For $\Delta_i(s) = \Delta_j(s) = 0$, $|i - j| > 2$ the following is true: $N_i(s) = N_{i+1}(s) =$

$\sum_{h=i+1}^{|B(s)|} \Delta_h(s)$ and $N_j(s) = N_{j+1}(s) = \sum_{h=j+1}^{|B(s)|} \Delta_h(s)$. Since $|i-j| > 2$, there are non-zero values between $\Delta_{i+1}(s)$ and $\Delta_j(s)$, $i+1 < j$, or between $\Delta_{j+1}(s)$ and $\Delta_i(s)$, $j+1 < i$. This implies two horizontal lines in PG. Existence of more than two lines is proved in a similar manner. ■

The problem of multiple equal lines is not resolved within GPC. In CLOSS it is approved for short ($2-4$ points) horizontal lines by smoothing PG with moving averages (Figure 4). The last one is defined at the overlap point i as

$$N_i^{sm}(s) = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} N_j(s). \quad (3)$$

The following theorem allows an efficient computation of PG jump.

Theorem 3: **[Smoothing Equivalence]** Let PG jump $\Delta_i^{sm}(s)$ be computed from smoothed $PG^{sm}(s)$ at overlap point i:

$$\Delta_i^{sm}(s) = N_i^{sm}(s) - N_{i+1}^{sm}(s) \quad (4)$$

Then PG jump $\Delta_i^{sm}(s)$ is equal to PG jump $\Delta_i(s)$ computed from $PG(s)$ and then smoothed at the same overlap point i:

$$\Delta_i^{sm}(s) = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} \Delta_j(s),\ i \in [i', i''].$$

Proof: The theorem is proved by substituting (3) into the expression of the smoothed PG jump (4). ■

The algorithm for the computing of moving averages with adaptive window is given in Figure 5. It inputs the proximity graph $PG(s)$, window size r and starting and ending overlap indexes i_{st}, i_{en}. First the algorithm finds the center index i_c in the interval $[i_{st}, i_{en}]$. Then it averages the sizes of the neighborhoods $\{N_i(s), i = 1, \ldots, |B(s)|\}$ of the PG in the intervals $[i_{st}, i_c)$ and $[i_c, i_{en}]$. If the bound of the sliding window goes beyond the starting or ending overlap index, the algorithm acts recursively by calling itself with $r = r-1$ (adaptive window) until all sizes of neighborhoods are averaged.

Input:
$PG(s)$: proximity graph, r: size of smoothing window,
i_{st}: index of the starting point (i_{st}, $N_{st}(s)$),
i_{en}: index of the ending point (i_{en}, $N_{en}(s)$).
Output:
$PG^{sm}(s) = \{(1, N_1^{sm}(s)), ..., (l, N_l^{sm}(s))\}$: smoothed PG
Body:
1. Find the center overlap index in $PG(s)$
   $i_c = round(|\{i_{st}, ..., i_{en}\}|/2)$;
   $i = i_c - 1$; $j = i_c$;
2. While $i - r \geq st$ Do $N_i^{sm}(s) = \frac{1}{2r+1} \sum_{h=i-r}^{i+r} N_h(s)$; i--;
3. While $i + r \leq en$ Do $N_j^{sm}(s) = \frac{1}{2r+1} \sum_{h=j-r}^{j+r} N_h(s)$; j++;
4. If $i - r < st$ Then $Smooth(PG(s), st, i, r-1)$;
5. If $j + r > en$ Then $Smooth(PG(s), j, en, r-1)$;
6. Return $PG^{sm}(s)$;

Fig. 5. $Smooth(PG(s), st, en, r)$ Algorithm

C. Ordering of the Database Strings

CLOSS sizes strings in the decreasing order of length to ensure better clustering. If clustering starts with the string s that is not the longest one, all strings having the overlap with s higher than the border i_b will be included into the cluster of the string s. It is clear that in large databases many of them are not misspellings of s. Since the cluster is removed from the clustering process, other clusters may loose a part of their strings. The more times the clustering is initialized by a short string the more longer strings are clustered falsely and the more clusters are found incorrectly. At the same time if the clustering starts from the longest string then only strings of similar length and sufficiently high overlap with the initial string are excluded from the following consideration. Thus for the data shown in Figure 2(a) $o(\text{any}, \text{annie}) = o(\text{any}, \text{amy}) = 2$. If the clustering starts with the short string $s =$ any then longer strings, such as $s' =$ annie, sharing two 2-grams with s will be clustered together with the misspellings of s.

V. Algorithm

In this section we introduce the CLOSS algorithm. It takes strings as an input and outputs a clustered set of strings (Figure 6). CLOSS has similar complexity to the GPC method and runs in time $O(l \log l \cdot n^2)$ in the worst case. The smoothing and interpolation do not affect the algorithm complexity, because the PGs consist of sufficiently small number of overlap indexes for short strings, i.e. $i_{max} = i_{l+q-1}$, $l = \text{length}(s)$, is small.

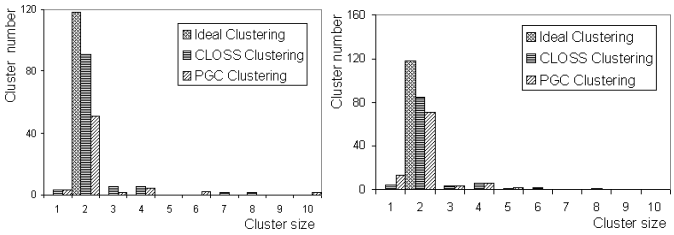
Input:
$D = \{s_1, s_2, ..., s_n\}$: database of ordered strings,
q: size of q-grams, r: range of smoothing.
Output:
$Clusters = \{C_1, ..., C_d\}$: clusters of strings.
Body:
1. Initialize the clustered strings
   $Clustered\_Strings = \emptyset$; $Clusters = \{\emptyset\}$.
2. Scan database strings. For each $s \in D$ Do
   2.1. If $s \notin Clustered\_Strings$ Then
      2.1.1. Compute the proximity graph
         $PG(s) = \{(1, N_1(s)), ..., (l, N_l(s))\}$ (see [1]).
      2.1.2. Compute the smoothed proximity graph
         $PG^{sm}(s) = Smooth(PG(s), 1, l, r)$ (see Figure 5).
      2.1.3. Find the interval $[i', i'']$ (see eq. (**??**)-(**??**)).
      2.1.4. Find the border $i_b \in PG^{sm}(s)$, $i_b \in [i', i'']$,
         by computing the PG jump $\Delta_{i_b}^{sm}(s)$:
         $\Delta_{i_b}^{sm}(s) = \min_i\{\Delta_i^{sm}(s)\}$, where
         $\Delta_i^{sm}(s) = N_i^{sm}(s) - N_{i+1}^{sm}(s)$.
   2.2 Update the clustered strings:
      $Clustered\_Strings = Clustered\_Strings \cup C_{i_b}$,
      C_{i_b} is such that $\forall\ s' \in C_{i_b}$: $o(s, s') \geq i_b$.
   2.4 Insert a new cluster to the set of clusters:
      $Clusters = Clusters \cup \{C_{i_b}\}$.
3. Return $Clusters$.

Fig. 6. CLOSS Algorithm

VI. Experiments

We evaluated CLOSS by applying it to the dataset of the tropical cyclone names of 3–9 character length. The dataset was artificially enriched by one misspelling for each name. Figure 7(a) demonstrates clustering of names with

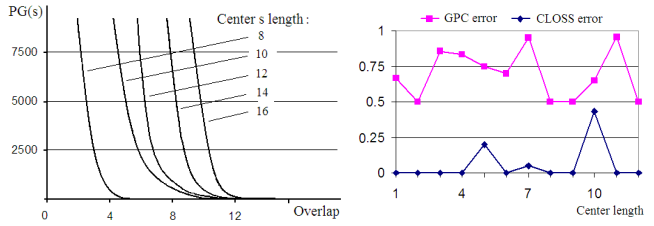
one orphografical mistake. In Figure 7(b) the number of mistakes is greater (up to four). They were obtained by string shortening up to 5 characters. As one can see the results that are essentially closer to the ideal clustering represented in histograms by one bar at the overlap point 2.



(a) One Mistake (b) Many Mistakes

Fig. 7. Comparision of CLOSS and GPC on the Dataset of Names

The use of the CLOSS method in real world tasks was considered on the example of the Oxford text archive. The task refers to text retrieval in social bookmarking when misspellings serve to find additional information. The file 'birkbeck', containing 36133 recorded misspellings of 6136 words, was considered as a misspelling source. CLOSS was chosen as a filtering tool because many words are corrupted in the file to such extent that it is impossible to assign them to any cluster even approximately. We looked for word clusters that could be used as representatives for keywords in tags. That is, if a word from the cluster is chosen, then all strings from the cluster take part in information retrieval. We also used this dataset to compare CLOSS and GPC on the large dataset.



(a) PG Shapes (b) Clusterring Error

Fig. 8. Clustering of Misspellings of the Oxford Dataset

Experimental results for this data are illustrated in Figure 8. In Figure 8(a) we show PGs for the selected cluster collections, while in Figure 8(b) we give the clustering error expressed as $\delta = 1 - |C^{\text{method}}|/|C|$, where C^{method} is a cluster of strings correctly found by CLOSS or GPC, C is the correct cluster of misspellings. Figures confirm that PGs obtained for the names 'resource', 'restaurant', 'acquiescence', 'recommendation', 'transportability' with lengths from 8 to 16 have a horizontal line and GPC can be used. However for strings shorter than 16 characters the CLOSS method suits better. The errors show that CLOSS is a lot better for words of different length, namely for 'goose', 'pigeon', 'mystery', 'resource', 'hydraulic', 'accustomed', 'magnificent', 'acquiescence', 'surreptitious', 'recommendation', 'congratulations', and 'transportability'.

Smoothing was tested by filtering out alien strings from the cluster. An example is given in Figure 9 ($r = 1$). The cluster was obtained for the word 'uninitelligible' and contained another correct word 'unintelligent'. After the smoothing the cluster border shifted from the point A to the point B that removed the string 'unintelligent' from the cluster defined by the point A.

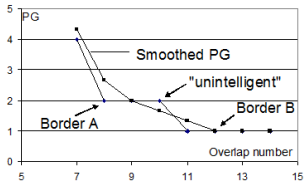


Fig. 9. Example of the PG Smoothing

From the experiments we conclude that the spread of error values increase as string length increases. It is because the number of misspellings for longer strings is greater, and therefore strings are less similar within the cluster. Moreover, the longer the center of a cluster is the more similar misspellings are to the strings of other clusters. For example, misspelling 'recomdation' having 3 overlap dissimilarity with the center 'recommendation" (Figure 8(b), $l = 14$, 30 misspellings) is more similar to the word 'recondition' from which it is moved away only by 2 overlaps. Similarly, misspelling 'recomentation' (dissimilarity is 4 overlaps) is more similar to 'cementation' (dissimilarity is 2 overlaps), etc. Misspellings that are closer to other correct words must be included into the corresponding clusters and eliminated from the considered one. Trying to exclude these strings CLOSS decreases the border i_b. As a result a part of own misspellings stays outside of the cluster and increases the error δ_{CLOSS}. Since not all strings conduct themselves in that way (cf. with the word 'surreptitious' (Figure 8(b), $l = 13$, 1 misspelling) error behavior has a jump form.

Future Work

Further research is related to multiple horizontal lines. If they are short ($3-4$ overlap values) smoothing gives positive results. If they are longer, what is typical for middle and long strings, the lines may not be removed by moving averages. For this case another smoothing technique should be proposed.

References

[1] A. Mazeika and M. H. Böhlen. Cleansing databases of misspelled proper nouns. In *CleanDB*, 2006.

[2] C. Li, B. Wang, and X. Yang. Vgram: Improving performance of approximate queries on string collections using variable-length grams. In *VLDB*, pages 303–314, 2007.

[3] C. Xiao, W. Wang, and X. Lin. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *PVLDB*, 1(1):933–944, 2008.

[4] F. Mokhtarian and A. K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.