Accuracy of Approximate String Joins Using Grams

Oktie Hassanzadeh
University of Toronto
10 King's College Rd.
Toronto, ON M5S3G4, Canada
oktie@cs.toronto.edu

Mohammad Sadoghi
University of Toronto
10 King's College Rd.
Toronto, ON M5S3G4, Canada
mo@cs.toronto.edu

Renée J. Miller
University of Toronto
10 King's College Rd.
Toronto, ON M5S3G4, Canada
miller@cs.toronto.edu

ABSTRACT

Approximate join is an important part of many data cleaning and integration methodologies. Various similarity measures have been proposed for accurate and efficient matching of string attributes. The accuracy of the similarity measures highly depends on the characteristics of the data such as the amount and type of the errors and length of the strings. Recently, there has been an increasing interest in using methods based on q-grams (substrings of length q) made out of the strings, mainly due to their high efficiency. In this work, we evaluate the accuracy of the similarity measures used in these methodologies. We present an overview of several similarity measures based on q-grams. We then thoroughly compare their accuracy on several datasets with different characteristics. Since the efficiency of approximate joins depends on the similarity threshold they use, we study how the value of the threshold (including values used in recent performance studies) affects the accuracy of the join. We also compare different measures based on the highest accuracy they can achieve on different datasets.

1. INTRODUCTION

Data quality is a major concern in operational databases and data warehouses. Errors may be present in the data due to a multitude of reasons including data entry errors, lack of common standards and missing integrity constraints. String data is by nature more prone to such errors. *Approximate join* is an important part of many data cleaning methodologies and is well-studied: given two large relations, identify all pairs of records that *approximately match*. A variety of similarity measures have been proposed for string data in order to match records. Each measure has certain characteristics that make it suitable for capturing certain types of errors. By using a string similarity function $sim()$ for the approximate join algorithm, all pairs of records that have similarity score above a threshold θ are considered to approximately match and are returned as the output.

Performing approximate join on a large relation is a notoriously time-consuming task. Recently, there has been an increasing interest in using approximate join techniques based on q-grams (substrings of length q) made out of the strings. Most of the efficient approximate join algorithms (which we describe in Section 2) are based on using a specific similarity measure, along with a fixed threshold value to return pairs of records whose similarity is greater than the threshold. The effectiveness of the majority of these algorithms depends on the value of the threshold used. However, there has been little work studying the accuracy of the join operation. The accuracy is known to be dataset-dependent and there is no common framework for evaluation and comparison of accuracy of different similarity measures and techniques. This makes comparing their accuracy a difficult task. Nevertheless, we argue that it is possible to evaluate relative performance of different measures for approximate joins by using datasets containing different types of known quality problems such as typing errors and differences in notation and abbreviations.

In this paper, we present an overview of several similarity measures for approximate string joins using q-grams and thoroughly evaluate their accuracy for different values of thresholds and on datasets with different amount and types of errors. Our results include:

- We show that for all similarity measures, the value of the threshold that results in the most accurate join highly depends on the type and amount of errors in the data.
- We compare different similarity measures by comparing the maximum accuracy they can achieve on different datasets using different thresholds. Although choosing a proper threshold for the similarity measures without a prior knowledge of the data characteristics is known to be a difficult task, our results show which measures can potentially be more accurate assuming that there is a way to determine the best threshold. Therefore, an interesting direction for future work is to find an algorithm for determining the value of the threshold for the most accurate measures.
- We show how the amount and type of errors affect the best value of the threshold. An interesting result of this is that many previously proposed algorithms for enhancing the performance of the join operation and making it scalable for large datasets are not effective enough in many scenarios, since the performance of these algorithms highly depends on choosing a high value for the threshold which could result in a very

VLDB '07, September 23-28, 2007, Vienna, Austria.

low accuracy. Our evaluation shows the effectiveness of those join algorithms that are less sensitive to the value of the threshold and opens another interesting direction for future work which is finding algorithms that are both efficient and accurate using the same threshold.

The paper is organized as follows. In Section 2, we overview related work on approximate joins. We describe in detail the approximate join algorithms we will evaluate and compare in Section 3, along with the similarity measures used. Section 4 presents a thorough evaluation of these algorithms and measures and finally, Section 5 concludes the paper and explains future directions.

2. RELATED WORK

Approximate join also known as *similarity join* or *record linkage* has been extensively studied in the literature. Several similarity measures for string data have been proposed [14, 4, 5]. A recent survey [9], presents an excellent overview of different types of string similarity measures. Recently, there has been an increasing interest in using measures from the Information Retrieval (IR) field along with q-grams made out of strings [10, 6, 2, 18, 5]. In this approach, strings are treated as documents and q-grams are treated as tokens in the documents. This makes it possible to take advantage of several indexing techniques as well as various algorithms that have been proposed for efficient *set-similarity joins*. Furthermore, these measures can be implemented declaratively over a DBMS with vanilla SQL statements [5].

Recent work addresses the problem of efficiency and scalability of the similarity join operations for large datasets [6, 2, 18]. Many techniques are proposed for set-similarity join, which can be used along with q-grams for the purpose of (string) similarity joins. Most of the techniques are based on the idea of creating signatures for sets (strings) to reduce the search space. Some signature generations schemes are derived from dimensionality reduction for the similarity search problem in high dimensional space. One efficient approach uses the idea of *Locality Sensitive Hashing* (LSH) [13] in order to hash similar sets into the same value with high probability and therefore is an approximate solution to the problem. Arasu et al. [2] propose algorithms specifically for set-similarity joins that are exact and outperform previous approximation methods in their framework, although parameters of the algorithms require extensive tuning. Another class of work is based on using indexing algorithms, primarily derived from IR optimization techniques. A recent proposal in this area [3] presents algorithms based on novel indexing and optimization strategies that do not rely on approximation or extensive parameter tuning and outperform previous state-of-the-art approaches. More recently, Li et al. [15] propose VGRAM, a technique based on the idea of using variable-length grams instead of q-grams. At a high level, it can be viewed as an efficient index structure over the collection of strings. VGRAM can be used along with previously proposed signature-based algorithms to significantly improve their efficiency.

Most of the techniques described above mainly address the scalability of the join operation and not the accuracy. The choice of the similarity measure is often limited in these algorithms. The signature-based algorithm of [6] also considers accuracy by introducing a novel similarity measure called *fuzzy match similarity* and creating signatures for this measure. However, accuracy of this measure is not compared with other measures. In [5], several such similarity measures are benchmarked for approximate selection, which is a special case of similarity join. Given a relation R, the approximate selection operation, using similarity predicate $sim()$, will report all tuples $t \in R$ such that $sim(t_q, t) \geq \theta$, where θ is a specified numerical *similarity threshold* and t_q is a query string. While several predicates are introduced and benchmarked in [5], the extension of approximate selection to approximate joins is not considered. Furthermore, the effect of threshold values on accuracy for approximate joins is also not considered.

3. FRAMEWORK

In this section, we explain our framework for similarity join. The similarity join of two relations $R = \{r_i : 1 \leq i \leq N_1\}$ and $S = \{s_j : 1 \leq j \leq N_2\}$ outputs a set of pairs $(r_i, s_j) \in R \times S$ where r_i and s_j are *similar* records. Two records are considered similar when their similarity score based a similarity function $sim()$ is above a threshold θ. For the definitions and experiments in this paper, we assume we are performing a self-join on relation R. Therefore the output is a set of pairs $(r_i, r_j) \in R \times R$ where $sim(r_i, r_j) \geq \theta$ for some similarity function $sim()$ and a threshold θ. This is a common operation in many applications such as entity resolution and clustering. In keeping with many approximate join methods, we model records as strings. We denote by $\mathbf{r}$ the set of *q-grams* (sequences of q consecutive characters of a string) in r. For example, for t='db lab', $\mathbf{t}$={'db ' ,'b l',' la', 'lab'} for tokenization using 3-grams. In certain cases, a *weight* may be associated with each token that reflects the commonality of the token in the relation.

The similarity measures discussed here are those based on q-grams created out of strings along with a similarity measure that has been shown to be effective in previous work [5]. These measures share one or both of the following properties:

- High scalability: There are various techniques proposed in the literature as described in Section 2 for enhancing the performance of the similarity join operation using q-grams along with these measures.

- High accuracy: Previous work has shown that in most scenarios these measures perform better or equally well in terms of accuracy when compared with other string similarity measures. Specifically, these measures have shown good accuracy in name-matching tasks [8] or in approximate selection [5].

3.1 Edit Similarity

Edit-distance is widely used as the measure of choice in many similarity join techniques. Specifically, previous work [10] has shown how to use q-grams for an efficient implementation of this measure in a declarative framework. Recent work on enhancing performance of similarity join has also proposed techniques for scalable implementation of this measure [2, 15].

Edit distance between two string records r_1 and r_2 is defined as the transformation cost of r_1 to r_2, $tc(r_1, r_2)$, which is equal to the minimum cost of edit operations applied to r_1 to transform it to r_2. Edit operations include character

copy, *insert*, *delete* and *substitute* [11]. The edit similarity is defined as:

$$sim_{edit}(r_1, r_2) = 1 - \frac{tc(r_1, r_2)}{\max\{|r_1|, |r_2|\}} \quad (1)$$

There is a cost associated with each edit operation. There are several cost models proposed for edit operations for this measure. The most commonly used measure called Levenshtein edit distance, which we will refer to as edit distance in this paper, uses unit cost for all operations except copy which has zero cost.

3.2 Jaccard and Weighted Jaccard

Jaccard similarity is the fraction of tokens in r_1 and r_2 that are present in both. Weighted Jaccard similarity is the weighted version of Jaccard similarity, i.e.,

$$sim_{WJaccard}(r_1, r_2) = \frac{\sum_{t \in \mathbf{r}_1 \cap \mathbf{r}_2} w_R(t)}{\sum_{t \in \mathbf{r}_1 \cup \mathbf{r}_2} w_R(t)} \quad (2)$$

where $w_R(t)$ is a weight function that reflects the commonality of the token t in the relation R. We choose RSJ (Robertson-Sparck Jones) weight for the tokens which was shown to be more effective than the commonly-used Inverse Document Frequency (IDF) weights [5]:

$$w_R(t) = log\left(\frac{N - n_t + 0.5}{n_t + 0.5}\right) \quad (3)$$

where N is the number of tuples in the base relation R and n_t is the number of tuples in R containing the token t.

3.3 Measures from IR

A well-studied problem in information retrieval is the problem of given a query and a collection of documents, return the most *relevant* documents to the query. In the measures in this part, records are treated as documents and q-grams are seen as words (tokens) of the documents. Therefore, the same techniques for finding relevant documents to a query can be used to return *similar* records to a query string. In the rest of this section, we present three measures that have been shown to have higher performance for the approximate selection problem [5].

3.3.1 Cosine w/tf-idf

The *tf-idf cosine* similarity is a well established measure in the IR community which leverages the vector space model. This measure determines the closeness of the input strings r_1 and r_2 by first transforming the strings into unit vectors and then measuring the angle between their corresponding vectors. The *cosine* similarity with tf-idf weights is given by:

$$sim_{Cosine}(r_1, r_2) = \sum_{t \in \mathbf{r}_1 \cap \mathbf{r}_2} w_{r_1}(t) \cdot w_{r_2}(t) \quad (4)$$

where $w_{r_1}(t)$ and $w_{r_2}(t)$ are the normalized *tf-idf* weights for each common token in r_1 and r_2 respectively. The normalized *tf-idf* weight of token t in a given string record r is defined as follows:

$$w_r(t) = \frac{w'_r(t)}{\sqrt{\sum_{t' \in \mathbf{r}} w'_r(t')^2}} \quad , \quad w'_r(t) = tf_r(t) \cdot idf(t)$$

where $tf_r(t)$ is the term frequency of token t within string r and $idf(t)$ is the inverse document frequency with respect to the entire relation R.

3.3.2 BM25

The BM25 similarity score for a query r_1 and a string record r_2 is defined as follows:

$$sim_{BM25}(r_1, r_2) = \sum_{t \in \mathbf{r}_1 \cap \mathbf{r}_2} \hat{w}_{r_1}(t) \cdot w_{r_2}(t) \quad (5)$$

where

$$\hat{w}_{r_1}(t) = \frac{(k_3+1) \cdot tf_{r_1}(t)}{k_3 + tf_{r_1}(t)}$$

$$w_{r_2}(t) = w_R^{(1)}(t) \frac{(k_1+1) \cdot tf_{r_2}(t)}{K(r_2) + tf_{r_2}(t)}$$

and $w_R^{(1)}$ is the RSJ weight:

$$w_R^{(1)}(t) = \log\left(\frac{N - n_t + 0.5}{n_t + 0.5}\right)$$

$$K(r) = k_1\left((1-b) + b\frac{|r|}{avg_{rl}}\right)$$

where $tf_r(t)$ is the frequency of the token t in string record r, $|r|$ is the number of tokens in r, avg_{rl} is the average number of tokens per record, N is the number of records in the relation R, n_t is the number of records containing the token t and k_1, k_3 and b are set of independent parameters. We set these parameters based on TREC-4 experiments [17] where $k \in [1, 2]$, $k_3 = 8$ and $b \in [0.6, 0.75]$.

3.3.3 Hidden Markov Model

The approximate string matching could be modeled by a discrete Hidden Markov process which has been shown to have better performance than Cosine w/tf-idf in the IR literature [16] and high accuracy and low running time for approximate selection [5]. This particular Markov model consists of only two states where the first state models the tokens that are specific to one particular "String" and the second state models the tokens in the "General English", i.e., tokens that are common in many records. Refer to [5] and [16] for a complete description of the model and possible extensions.

The HMM similarity function accepts two string records r_1 and r_2 and returns the probability of generating r_1 given r_2 is a similar record:

$$sim_{HMM}(r_1, r_2) = \prod_{t \in \mathbf{r}_1} (a_0 P(t|GE) + a_1 P(t|r_2)) \quad (6)$$

where a_0 and $a_1 = 1 - a_0$ are the transition states probabilities of the Markov model and $P(t|GE)$ and $P(t|r_2)$ is given by:

$$P(t|r_2) = \frac{\text{number of times } t \text{ appears in } r_2}{|\mathbf{r}_2|}$$

$$P(t|GE) = \frac{\sum_{r \in R} \text{number of times } t \text{ appears in } r}{\sum_{r \in R} |\mathbf{r}|}$$

3.4 Hybrid Measures

The implementation of these measures involves two similarity functions, one that compares the strings by comparing

their word tokens and another similarity function which is more suitable for short strings and is used for comparison of the word tokens.

3.4.1 GES

The generalized edit similarity (GES) [7] which is a modified version of *fuzzy match similarity* presented in [6], takes two strings r_1 and r_2, tokenizes the strings into a set of words and assigns a weight $w(t)$ to each token. GES defines the similarity between the two given strings as a minimum transformation cost required to convert string r_1 to r_2 and is given by:

$$sim_{GES}(r_1, r_2) = 1 - \min\left(\frac{tc(r_1, r_2)}{wt(r_1)}, 1.0\right) \quad (7)$$

where $wt(r_1)$ is the sum of weights of all tokens in r_1 and $tc(r_1, r_2)$ is the minimum cost of a sequence of the following transformation operations:

- *token insertion:* inserting a token t in r_1 with cost $w(t).c_{ins}$ where c_{ins} is the insertion factor constant and is in the range between 0 and 1. In our experiments, $c_{ins} = 1$.
- *token deletion:* deleting a token t from r_1 with cost $w(t)$.
- *token replacement:* replacing a token t_1 by t_2 in r_1 with cost $(1 - sim_{edit}(t_1, t_2)) \cdot w(t)$ where sim_{edit} is the edit-distance between t_1 and t_2.

3.4.2 SoftTFIDF

SoftTFIDF is another hybrid measure proposed by Cohen et al. [8], which relies on the normalized *tf-idf* weight of word tokens and can work with any arbitrary similarity function to find the similarity between word tokens. In this measure, the similarity score, $sim_{SoftTFIDF}$, is defined as follows:

$$\sum_{t_1 \in C(\theta, r_1, r_2)} w(t_1, r_1) \cdot w(\arg\max_{t_2 \in r_2}(sim(t_1, t_2)), r_2) \cdot \max_{t_2 \in r_2}(sim(t_1, t_2)) \quad (8)$$

where $w(t, r)$ is the normalized *tf-idf* weight of word token t in record r and $C(\theta, r_1, r_2)$ returns a set of tokens $t_1 \in r_1$ such that for $t_2 \in r_2$ we have $sim(t_1, t_2) > \theta$ for some similarity function $sim()$ suitable for comparing word strings. In our experiments $sim(t_1, t_2)$ is the Jaro-Winkler similarity as suggested in [8].

4. EVALUATION

4.1 Datasets

In order to evaluate the effectiveness of different similarity measures described in previous section, we use the same datasets used in [5]. These datasets were created using a modified version of the UIS data generator, which has previously been used for the evaluation of data cleaning and record linkage techniques [12, 1]. The data generator has the ability to inject several types of errors into a clean database of string attributes. These errors include commonly occurring typing mistakes (edit errors: character insertion, deletion, replacement and swap), token swap and abbreviation errors (e.g., replacing `Inc.` with `Incorporated` and vice versa). The data generator has several parameters to control the injected error in the data such as the size of the dataset to be generated, the distribution of duplicates (uniform, Zipfian or Poisson), the percentage of erroneous duplicates, the extent of error injected in each string, and the percentage of different types of errors. The data generator keeps track of the duplicate records by assigning a cluster ID to each clean record and to all duplicates generated from that clean record.

Group	Name	Percentage of			
		Erroneous Duplicates	Errors in Duplicates	Token Swap	Abbr. Error
Dirty	D1	90	30	20	50
	D2	50	30	20	50
Medium Error	M1	30	30	20	50
	M2	10	30	20	50
	M3	90	10	20	50
	M4	50	10	20	50
Low Error	L1	30	10	20	50
	L2	10	10	20	50
Single Error	AB	50	0	0	50
	TS	50	0	20	0
	EDL	50	10	0	0
	EDM	50	20	0	0
	EDH	50	30	0	0

Table 1: Datasets Used in the Experiments

For the results presented in this paper, the datasets are generated by the data generator out of a clean dataset of 2139 company names with average record length of 21.03 and an average of 2.9 words per record. The errors in the datasets have a uniform distribution. For each dataset, on average 5000 dirty records are created out of 500 clean records. We have also run experiments on datasets generated using different parameters. For example, we generated data using a Zipfian distribution, and we also used data from another clean source (DBLP titles) as in [5]. We also created larger datasets. For these other datasets, the accuracy trends remain the same. Table 1 shows the description of all the datasets used for the results in this paper. We used 8 different datasets with mixed types of errors (edit errors, token swap and abbreviation replacement). Moreover, we used 5 datasets with only a single type of error (edit errors, token swap or abbreviation replacement errors) to measure the effect of each type of error individually. Following [5], we believe the errors in these datasets are highly representative of common types of errors in databases with string attributes.

4.2 Measures

We use well-known measures from IR, namely precision, recall, and F1, for different values of the threshold to evaluate the accuracy of the similarity join operation. We perform a self-join on the input table using a similarity measure with a fixed threshold θ. *Precision (Pr)* is defined as the percentage of *similar* records among the records that have a similarity score above the threshold θ. In our datasets, *similar* records are marked with the same cluster ID as described above. *Recall (Re)* is the ratio of the number of *similar* records that have similarity score above the threshold θ to the total number of *similar* records. Therefore, a join that returns all the pairs of records in the two input tables as output has low (near zero) precision and recall of 1. A join that returns an empty answer has precision 1 and zero recall. The F_1 measure is the harmonic mean of precision and

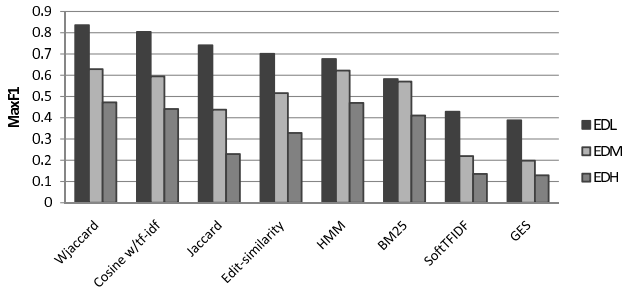


Figure 3: Maximum F_1 score for different measures on datasets with only edit errors

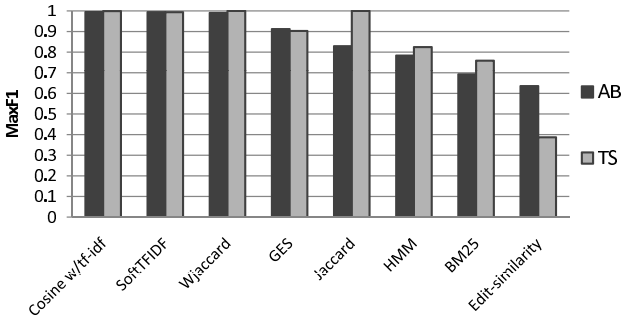


Figure 4: Maximum F_1 score for different measures on datasets with only token swap and abbreviation errors

recall, i.e.,

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re} \tag{9}$$

We measure precision, recall, and F_1 for different values of the similarity threshold θ. For comparison of different similarity measures, we use the maximum F_1 score across different thresholds.

4.3 Results

Figures 1 and 2 show the precision, recall, and F_1 values for all measures described in Section 3, over the datasets we have defined with mixed types of errors. For all measures except HMM and BM25, the horizontal axis of the precision/recall graph is the value of the threshold. For HMM and BM25, the horizontal axis is the percentage of maximum value of the threshold, since these measure do not return a score between 0 and 1.

Effect of amount of errors As shown in the precision/recall curves in Figures 1 and 2, the "dirtiness" of the input data greatly affects the value of the threshold that results in the most accurate join. For all the measures, a lower value of the threshold is needed as the degree of error in the data increases. For example, Weighted Jaccard achieves the best F_1 score over the dirtiest datasets with threshold 0.3, while it achieves the best F_1 for the cleanest datasets at threshold 0.55. BM25 and HMM are less sensitive and work well on both dirty and low-error datasets with the same value of the threshold. We will discuss later how the degree of error in the data affects the choice of the most accurate measure.

Effect of types of errors Figure 3 shows the maximum F_1 score for different values of the threshold for different measures on datasets containing only edit-errors (the `EDL`, `EDM` and `EDH` datasets). These figures show that weighted Jaccard and Cosine have the highest accuracy followed by Jaccard, and edit similarity on the low-error dataset `EDL`. By increasing the amount of edit error in each record, HMM performs as well as weighted Jaccard, although Jaccard, edit similarity, and GES perform much worse on high edit error datasets. Considering the fact that edit-similarity is mainly proposed for capturing edit errors, this shows the effectiveness of weighted Jaccard and its robustness with varying amount of edit errors. Figure 4 shows the effect of token swap and abbreviation errors on the accuracy of different measures. This experiment indicates that edit similarity is not capable of modeling such types of errors. HMM, BM25 and Jaccard also are not capable of modeling abbreviation errors properly.

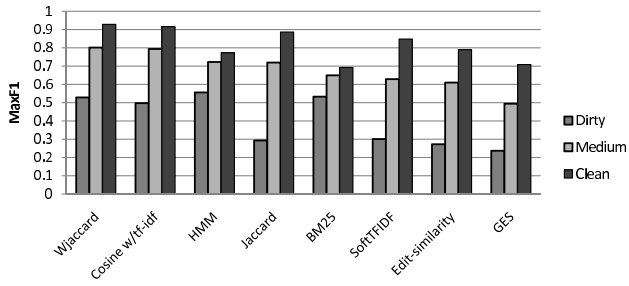


Figure 5: Maximum F_1 score for different measures on dirty, medium and low-error group of datasets

Comparison of measures Figures 5 shows the maximum F_1 score for different values of the threshold for different measures on dirty, medium and low-error datasets. Here, we have aggregated the results for all the dirty data sets together (respectively, the moderately dirty or medium data sets and the low error data sets). The results show the effectiveness and robustness of weighted Jaccard and cosine in comparison with other measures. Again, HMM is among the most accurate measures when the data is extremely dirty and has relatively low accuracy when the percentage of error in the data is low.

Remark As stated in Section 2, the performance of many algorithms proposed for improving scalability of the join operation highly depends on the value of similarity threshold used for the join. Here we show the accuracy numbers on our datasets using the value of the threshold that makes these algorithms effective. Specifically we address the results in [2] although similar observations can be made for results of other similar work in this area. Table 2 shows the F_1 value for thresholds that results in the best accuracy on our datasets and the best performance in experimental results of [2]. PartEnum and WtEnum algorithms presented in [2] significantly outperform previous algorithms for 0.9 threshold, but have roughly the same performance as previously proposed algorithms such as LSH when threshold 0.8 or less is used. The results in Table 2 show that there is a big gap between the value of the threshold that results in the most accurate join on our datasets and the threshold that results in the effectiveness of PartEnum and WtEnum in the studies in [2].

	Jaccard Join		Weighted Jaccard Join	
	Threshold	F_1	Threshold	F_1
Dirty	0.5 (Best Acc.)	0.293	0.3 (Best Acc.)	0.528
	0.8	0.249	0.8	0.249
	0.85	0.248	0.85	0.246
	0.9 (Best Performance)	0.247	0.9 (Best Performance)	0.244
Medium Error	0.65 (Best Acc.)	0.719	0.55 (Best Acc.)	0.776
	0.8	0.611	0.8	0.581
	0.85	0.571	0.85	0.581
	0.9 (Best Performance)	0.548	0.9 (Best Performance)	0.560
Low Error	0.7 (Best Acc.)	0.887	0.55 (Best Acc.)	0.929
	0.8	0.854	0.8	0.831
	0.85	0.831	0.85	0.819
	0.9 (Best Performance)	0.812	0.9 (Best Performance)	0.807

Table 2: F_1 score for thresholds that result in best running time in previous performance studies and highest accuracy on our datasets for two selected similarity measures

5. CONCLUSION

We have presented an overview of several similarity measures for efficient approximate string joins and thoroughly evaluated their accuracy on several datasets with different characteristics and common quality problems. Our results show the effect of the amount and type of errors in the datasets, along with the value of the similarity threshold used for the similarity measures, on the accuracy of the join operation. Considering the fact that the effectiveness of many algorithms proposed for enhancing the scalability of approximate join rely on the value chosen for the similarity threshold, our results show the effectiveness of those algorithms that are less sensitive to the value of the threshold and opens an interesting direction for future work which is finding algorithms that are both efficient and accurate using the same threshold. Finding an algorithm that determines the best value of the threshold (regardless of the type and amount of errors) for the similarity measures that showed higher accuracy in our work is another interesting subject for future work.

6. REFERENCES

[1] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE'06*, page 30.

[2] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB'06*, pages 918–929.

[3] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *WWW'07*, pages 131–140.

[4] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

[5] A. Chandel, O. Hassanzadeh, N. Koudas, M. Sadoghi, and D. Srivastava. Benchmarking declarative approximate selection predicates. In *SIGMOD'07*, pages 353–364.

[6] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD'03*, pages 313–324.

[7] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *ICDE '06*, page 5.

[8] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb'03*, pages 73–78.

[9] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE TKDE*, 19(1):1–16, 2007.

[10] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *VLDB'01*, pages 491–500.

[11] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997.

[12] M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[13] Indyk, Motwani, Raghavan, and Vempala. Locality-preserving hashing in multidimensional spaces. In *STOC'97*, pages 618–625.

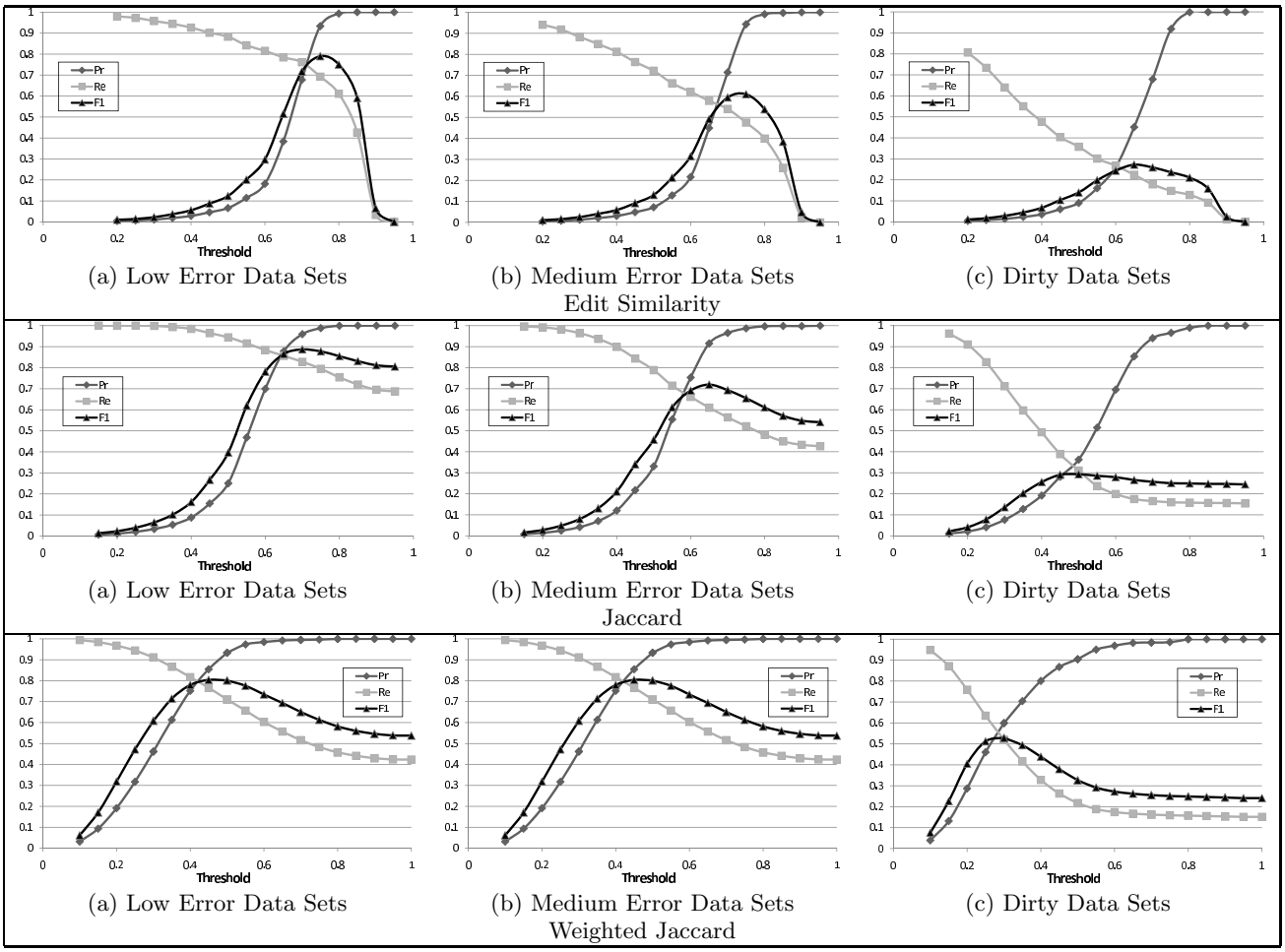
[14] N. Koudas and D. Srivastava. Approximate joins: Concepts and techniques. In *VLDB'05 Tutorial*, page 1363.

[15] C. Li, B. Wang, and X. Yang. Vgram: Improving performance of approximate queries on string collections using variable-length grams. In *VLDB'07*, pages 303–314.

[16] D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden markov model information retrieval system. In *SIGIR'99*, pages 214–221.

[17] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at trec-4. In *TREC'95*.

[18] S. Sarawagi and A. Kirpal. Efficient set joins on similarity predicates. In *SIGMOD'04*, pages 743–754.



(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

Edit Similarity

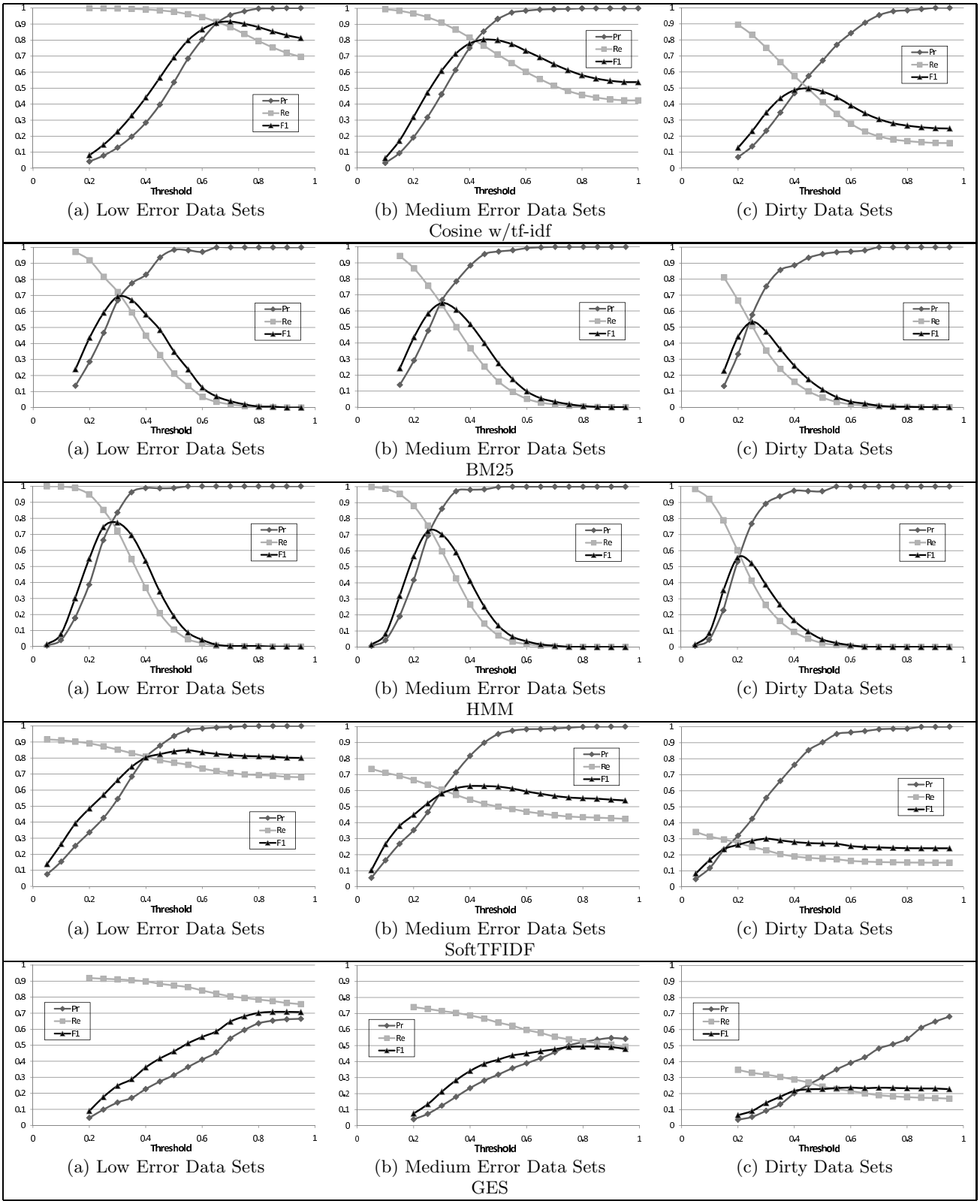
(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

Jaccard

(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

Weighted Jaccard

Figure 1: Accuracy of Edit-Similarity, Jaccard and Weighted Jaccard measures relative to the value of the threshold on different datasets



(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

Cosine w/tf-idf

(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

BM25

(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

HMM

(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

SoftTFIDF

(a) Low Error Data Sets (b) Medium Error Data Sets (c) Dirty Data Sets

GES

Figure 2: Accuracy of measures from IR and hybrid measures relative to the value of the threshold on different datasets