String reconciliation with unknown edit distance

Aryeh (Leonid) Kontorovich Email:karyeh@cs.bgu.ac.il
Department of Computer Science
Ben-Gurion University
Beer Sheva, Israel Ari Trachtenberg Email:trachten@bu.edu
Department of Electrical & Computer Engineering
Boston University
Boston, MA 02215

Abstract

We consider the problem of reconciling two remote strings of arbitrary and unknown similarity using minimum communication, which is at the core of some important problems in networking, cryptography, genetic engineering, and even linguistics. Though this problem is efficiently convertible into a set reconciliation instance, for which efficient solutions exist, this conversion may introduce ambiguity in the decoding process, which may require significant communication and computational resources to resolve. We leverage some recent advances in efficient unique decodability of strings to reduce decoding ambiguity, and thus pave the way for a practical implementation of this string reconciler. For certain random strings and in some ideal cases, our approach reconciles two length n strings that differ in α edits (with α not known a priori) using $O\left(\alpha \log^2(n)\right)$ communication.

A version of this article appeared as:

- A. Kontorovich and A. Trachtenberg, "String reconciliation with unknown edit distance", IEEE ISIT 2012, Boston, USA.

I. Introduction

The problem of efficiently reconciling similar strings is fundamental to a variety of problems. Within a biological context, this problem is related to sequencing of DNA from short reads [5] and reconstruction of protein sequences from K-peptides [22]. Communications protocols [1, 3] attempt to identify differences between related documents using string reconciliation, and fuzzy extractors [8] employ similar techniques to match noise-prone biometric data to baseline measurements in a cryptographically secure fashion. Even computational linguistics uses related approaches to learn transformations on varying-length sequences [21].

A. Approach

Our approach extends and, in many ways makes practical, the protocol in [1], which effectively translates the *edit distance* problem into a *set reconciliation* problem. The transformation is accomplished through the use of shingling [3, 4], wherein a string is divided into overlapping substrings (called *shingles*). Two remote strings are thus reduced to sets of shingles, which can be reconciled efficiently using existing approaches (e.g. [16, 17]).

Once a remote string's shingle set is known, the shingles can be pieced together to determine the remote string. The problem is that, in general, there may be a large number of possible string reconstructions from a given collection of shingles, equal to the number of Eulerian cycles in the corresponding de Bruijn graph. For example, the shingles

$$\{\mathsf{at}, \mathsf{an}, \mathsf{ka}, \mathsf{na}, \mathsf{ta}\} \tag{1}$$

can be combined into the string katana or kanata.

The work in [1] proposed two solutions to the reconstruction ambiguity. The first involves enumerating all (possibly exponentially many) Eulerian cycles in the graph, and transmitting the index of the cycle producing the desired string; this can require significant computational resources and may incur an additional communication cost superlinear in the number of shingles. An alternate solution in the same work shows that if all shingles are of a certain minimal length, roughly logarithmic in string length, then there is likely just one possible reconstruction; that solution views strings unrealistically as independent and identically distributed (iid) bits, and it also only gives a probabilistic assurance of unique reconstruction.

In this work, we propose a third solution based on the theory of uniquely decodable strings, wherein we efficiently tailor the shingles so as to guarantee exactly one possible reconstruction.

B. Outline

We begin in Section II with an overview of related work from the information theory and theoretical computer science communities, followed in Section III by a technical exposition of the existing approach to string reconciliation on which this work is based. Section IV summarizes relevant work in [13] for efficiently determining whether a string has a unique decoding. Our new string reconciliation protocol based upon an automaton for detecting unique decodability is described in Section V. We close with concluding remarks in Section VI.

II. Related work

1) Edit distance: The problem of determining the minimum number of edits (insertions or deletions) required to transform one string into another has a long history in the literature [6, 11]. Orlitsky [18] shows that amount of communication $C_{\hat{\alpha}}(x, y)$ *needed* to reconcile two strings x and y (of lengths $|x|$ and $|y|$ respectively) that are known to be at most $\hat{\alpha}$-edits apart is at most

$$C_{\hat{\alpha}}(x, y) \leq f(y) + 3\log f(y) + log\hat{\alpha} + 13,$$

for

$$\log\left(\binom{|y|+\hat{\alpha}}{\hat{\alpha}}\right) \leq \lceil f(y) \rceil \leq \log\left(\binom{|y|+\hat{\alpha}}{\hat{\alpha}}\right) + 3\log(\hat{\alpha}),$$

although he leaves an efficient one-way protocol as an open question.

The literature includes a variety of proposed protocols for this problem. Cormode et al. [7] propose a hash-based approach that requires a known bound $\hat{\alpha}$ on edits between x and y (assuming, without loss of generality, that y is the longer string) and communicates at most

$$4\alpha\log(\frac{2|y|}{\alpha})\log(2\hat{\alpha}) + O\left(\alpha\log n\log\frac{\log(n)}{\ln\frac{1}{1-\epsilon}}\right) \tag{2}$$

bits to reconcile the strings with probability of failure ϵ.

Orlitsky and Viswanthan [19] propose a interactive protocol that does not need to know the number of edits in advance and requires at most

$$2\alpha\log|y|\left(\log|y| + \log\log|y| + \log(1/\epsilon) + \log\alpha\right)$$

bits of communication.

Other approaches include those of Evfimievski [10] for small edit distances, Suel [23] based on delta-compression, and Tridgell [24] which presents the computationally efficient (but potentially communicationally inefficient) rsync protocol.

2) Unique Decoding: It was shown in [14] that the collection of strings having a unique reconstruction from the shingles representation is a regular language. An explicit construction of a deterministic finite-state automaton (DFA) recognizing this language was given in by Li and Xie [15]. Unfortunately, for an input alphabet Σ, this DFA has

$$2^{|\Sigma|}(|\Sigma|+1)(|\Sigma|+1)^{(|\Sigma|+1)} \in 2^{O(|\Sigma|\log|\Sigma|)}$$

states, and thus does not seem practical except for very small alphabets.

Our work in [13], briefly summarized in Section IV for sake of completeness, has demonstrated that there is no DFA of subexponential size for recognizing this language, and, instead, exhibited an equivalent NFA with $O(|\Sigma|^3)$ states, which we shall utilize in this paper.

There has also been work on the probability of a collection of shingles having a unique reconstruction. The authors in [1] show that we can expect a unique decoding for substrings of identically distributed, independent random bits as long as the substrings are roughly logarithmic in the size of the overall decoded string (a precise statement is provided in Section V).

The work in [9] also provides evidence of a high probability of unique decoding for logarithmically sized substrings, and includes generalizations to non-binary and even non-uniformly random characters for the strings. This is extended in [2] to categorize the number of decodings for a given collection of shingles, and [20] considers decoding from regularly gapped collections of substrings in a DNA sequencing framework.

III. BASE PROTOCOL

We next present the string reconciliation approach in [1], which is based on a transformation to an instance of the set reconciliation [17].

A. Set reconciliation

The problem of set reconciliation seeks to reconcile two remote sets S_A and S_B of b-bit integers using minimum communication. The approach in [17] involves translating set elements into an equivalent *characteristic polynomial*, so that the process of set reconciliation is translated into an equivalent problem of rational function interpolation, similar to Reed-Solomon decoding.

The resulting algorithm requires one message of roughly bm bits of communication and bm^3 computation to reconcile two sets that differ in m entries. As such, two sets of a billion 32-bit integer that differ in three integers can be reconciled with roughly 96 bits of communication. The approach can be improved to expected bm communication and computation through the use of interaction [16] and generalized to multisets (straightforwardly) and arbitrary error-correcting codes [12].

B. String reconciliation

A string σ can be transformed into a multiset S through shingling, or collecting all contiguous substrings of a given length. For example, shingling the string katana into size 2 shingles produces the multiset in (1). As such, in order to reconcile two strings σ_A and σ_B, the protocol STRING-RECON first shingles each string, then reconciles the resulting sets, and then puts the shingles back together into strings in order to complete the reconciliation. It is important to note that if two strings differ by α edits, then they will also differ in $O(\alpha)$ shingles, as long as shingle size is a constant.

The process of putting shingles of length l back into a string involves the construction of a modified de Bruijn graph of the shingles. In this graph, each shingle corresponds to an edge, with weight equal to the number times the shingle occurs in the multiset. The vertices of the graph are all length $l-1$ substrings over the shingling alphabet; in this manner, an edge $e(u, v)$ corresponds to a shingle s if u (resp. v) is a prefix (resp. suffix) of s. A special character $ used at the beginning and end of the string in order to mark the first and last shingle.

An Eulerian cycle in the modified de Bruijn graph, starting at the first shingle, necessarily corresponds to a string that is consistent with the set of shingles. Unfortunately, there may be a large number of such strings, requiring either the enumeration of a specific cycle of interest or another way to guarantee only one possible cycle.

IV. UNIQUE DECODING

Our string reconciliation approach hinges upon the ability to efficiently discern whether a string is uniquely decodable from its shingles. To this end, we next summarize the relevant work in [13] for the explicit construction of a deterministic finite-state automaton (DFA) recognizing exactly such strings, and an equivalent, but more efficient non-deterministic finite-state automaton (NFA).

A. Preliminaries

We assume a finite alphabet Σ along with a special delimiter character $\$ \notin \Sigma$, and define $\Sigma_\$ = \Sigma \cup \{\$\}$. For $k \geq 1$, the k-gram map Φ takes string $x \in \$\Sigma^*\$$ to a vector $\xi \in \mathbb{N}^{\Sigma_\$^k}$, where $\xi_{i_1,\ldots,i_k} \in \mathbb{N}$ is the number of times the string $i_1 \ldots i_k \in \Sigma^k$ occurred in x as a contiguous subsequence, counting overlaps. Note that, though we focus this section on the *bigram* case when $k = 2$, we will subsequently employ these results for the general case $k > 2$.

As we have seen in the introduction, the bigram map $\Phi : \$\Sigma^*\$ \to \mathbb{N}^{\Sigma_\$^2}$ is not injective; for example, $\Phi(\$\mathsf{katana}\$) = \Phi(\$\mathsf{kanata}\$)$.

We denote by $L_{\text{UNIQ}} \subseteq \Sigma^*$ the collection of all strings w for which

$$\Phi^{-1}(\Phi(\$w\$)) = \{\$w\$\}$$

and refer to these strings as *uniquely decodable*, meaning that there is exactly one way to reconstruct them from their bigrams. The induced *bigram graph* of a string $w \in \Sigma^*$ is a weighted directed graph $G = (V, E)$, with $V = \Sigma_\$$ and $E = \{e(a, b) : a, b \in \Sigma_\$\}$, where the edge weight $e(a, b) \geq 0$ records the number of times a occurs immediately before b in the string $\$w\$$. Finall, we will denote the omission of a symbol from the alphabet by $\Sigma_{\bar{x}} := \Sigma \setminus \{x\}$ for $x \in \Sigma$.

B. Construction and simulation of the NFA

For $x \in \Sigma$ and $a, b \in \Sigma_{\bar{x}}$, the languages

$$I_{x,a,b} = L\left(\Sigma^* a x \Sigma_{\bar{a}}^* b \Sigma^*\right)$$

and

$$J_{x,a,b} = L\left(\Sigma^* a \Sigma_{\bar{x}}^* b \Sigma^*\right)$$

form the obstruction language

$$K_{x,a,b} = I_{x,a,b} \cap J_{x,a,b},$$

whose elements are called *obstructions*. The language of all obstructions is thus

$$L_{\text{OBST}} = \bigcup_{x \in \Sigma} \bigcup_{a,b \in \Sigma_{\bar{x}}} K_{x,a,b}. \tag{3}$$

The work in [13] provides a canonical DFA that recognizes $K_{x,a,b}$ with 9 states, regardless of Σ. Over all $x \in \Sigma$ and $a, b \in \Sigma_{\bar{x}}$, there are

$$|\Sigma|\left(|\Sigma| - 1 + (|\Sigma| - 1)(|\Sigma| - 2)\right) \tag{4}$$

distinct obstruction languages, whose union can thus be accepted by an NFA of $O(|\Sigma|^3)$ states.

The main theorem is thus that the language of obstructions is precisely the complement of the language of uniquely decodable strings.

Theorem 1 ([13])**.**

$$L_{\text{OBST}} = \Sigma^* \setminus L_{\text{UNIQ}}.$$

The result of Theorem 1 is that the NFA accepting $K_{x,a,b}$'s can be used to efficiently test for unique decodability.

V. String reconciliation

We next propose the main protocol of our paper for string reconciliation, as an amalgam of Section III and the work in [13], presented as a high-level description in Protocol 1.

A. Definitions

Our protocol is fundamentally based on the concept of a *shingling*, as used in Section III. Recall that a *shingle* $s = s_1 s_2 \dots s_k$ is simply an element of $\Sigma_{\$}^*$. For two shingles $s = s_1 s_2 \dots s_k$ and $t = t_1 t_2 \dots t_\ell$, we write $s \overset{l}{\rightsquigarrow} t$ if there is some length $\geq l$ suffix u of s that is also a prefix of t, or, more precisely, if we can rewrite $s = s'u$ and $t = ut'$ for strings s', t' and $|u| \geq l$. We define the *non-overlapping concatenation* $s \odot t$ as the concatenation $s'ut'$, where $s = s'u$, $t = ut'$ and $|u| = l - 1$. For example, $\mathsf{kata} \overset{3}{\rightsquigarrow} \mathsf{tana}$ and $\mathsf{kata} \odot \mathsf{tana} = \mathsf{katana}$.

For a fixed l, the sequence of shingles $s^1 \overset{l}{\rightsquigarrow} s^2 \overset{l}{\rightsquigarrow} \dots \overset{l}{\rightsquigarrow} s^t$ is said to *represent* the word $w \in \Sigma^*$ if $w = s^1 \odot s^2 \odot \dots \odot s^t$ and $s^i \overset{l}{\rightsquigarrow} s^{i+1}$ for all i. If $S = \left\{s^1, \dots, s^t\right\}$ is a multiset of shingles, we will use $\Phi^{-1}(S) \subset \Sigma^*$ to denote the collection of all words represented by S. More formally, define $\Pi = \Pi(S)$ to be the set of all permutations on $t = |S|$ elements with the property that $s^{\pi(i)} \overset{l}{\rightsquigarrow} s^{\pi(i+1)}$ for all i. Then $\Phi^{-1}(S)$ is

$$\left\{w \in \Sigma^* : \$w\$ = s^{\pi(1)} \odot s^{\pi(2)} \odot \dots \odot s^{\pi(t)}, \pi \in \Pi\right\}.$$

We refer to the members of $\Phi^{-1}(S)$ as the *decodings* of S, and say that S is uniquely decodable if $|\Phi^{-1}(S)| = 1$.

An *shingling* I of a word $w = w_1 \dots w_t \in \Sigma^*$ is a set of substrings of w that represents w. We say that I is an uniquely decodable shingling of w if $|\Phi^{-1}(I(w))| = 1$.

As a simple example, consider the string $w = \mathsf{katana}$ with the shingling $I(w) = \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{ta}, \mathsf{an}, \mathsf{na}, \mathsf{n}\$\}$. As we saw in the introduction, for l=2, I can be alternately decoded into kanata and is thus not uniquely decodable. However, if the second and third shingles are merged into ata, that the shingling becomes $\{\$\mathsf{k}, \mathsf{ka}, \mathsf{ata}, \mathsf{an}, \mathsf{na}, \mathsf{n}\$\}$, and then there is exactly one decoding: katana.

B. Details

Protocol 1 transforms a string that is not uniquely decodable into one that is by merging shingles. Several important details of Protocol 1 require explanation and proof of correctness.

1) Steps 1 and 2: The first two steps of the protocol derive from the base protocol described in Section III. Note that l is an implementation parameter.

2) Step 3: The expression S_σ^i represents the multiset of shingles that have been seen so far. It is modified, by combining shingles as necessary in the subsequent steps, in order to ensure unique decodability. If full reconciliation is desired (i.e. both hosts know the other host's string, as opposed to just one host having this knowledge) then Steps 3 and 4 are similarly run on the remote host with set S_τ^i.

1. Split σ into a set S_σ of length l shingles, with the i-th shingle of the string denoted s_i. Similarly split τ into S_τ.
2. Reconcile sets S_σ and S_τ.
3. The first host sets $S_\sigma^0 \longleftarrow \{s_0\}$.
4. **For** i from 1 to $|\sigma| - l + 1$ **do**

   $S_\sigma^i \longleftarrow S_\sigma^{i-1} \cup \{s_i\}$

   While S_σ^i is not uniquely decodable

   Merge the last two shingles added to S_σ^i.
5. Exchange indices of merged shingles.
6. Uniquely decode S_σ^i and S_τ^i on the remote hosts.

Protocol 1: Reconciliation of remote strings σ and τ.

3) Step 4: In merging two shingles s_a and s_b, we are simply computing the non-overlapping concatenation $s_a := s_a \odot s_b$, as defined earlier. Since the shingles are contiguous and based on an initial length l shingling, we know necessarily that $s_a \overset{l}{\rightsquigarrow} s_b$. Furthermore, it is clear that such merging will always, eventually, lead to a decodable set of shingles because, at worst, the protocol results in just one shingle representing the entire string, which is necessarily uniquely decodable.

The main challenge of this step is in checking whether a given set of shingles is uniquely decodable. This can be done by considering the de Bruijn graph of the shingles. The vertices of this graph are the length $l-1$ prefixes and suffixes of the shingles, and the edges correspond directly to shingles, as described in Section III. Clearly a given set of shingles is uniquely decodable iff there is a unique Eulerian cycle through this graph. At the same point, if we were to relabel all vertices with distinct characters over a fictitious alphabet Σ' of cardinality S, then we can view the graph as a bigram graph over Σ'. In other words, determining the unique decodability of the shingle set is equivalent to determining the unique decodability of a string corresponding to an arbitrary Eulerian path in the graph, and this can be tested using the machinery described in Section IV.

4) Step 5: Each host needs to know which shingles were merged on the other host in order to produce a uniquely decodable multiset of shingles. Since each merge involves at least one shingle of length l, it suffices to exchange a list of indices of length l shingles that are involved in a merge. The index can be chosen canonically from an alphabetically ordered list of all shingles.

The success of the protocol relies upon having as few merge operations as possible, since, at worst, *every* shingle is merged in this step, requiring $n \log n$ bits of communication for a shingle set of size n. In the best case, no shingles are merged and the communication complexity of the protocol is directly related to the edit distance between reconciled strings.

Though it is hard to give precise bounds on the number of shingles that are merged in this step, the work in [1] provides some guidance for random strings. Specifically, for strings of n random bits, in which each bit is 0 with

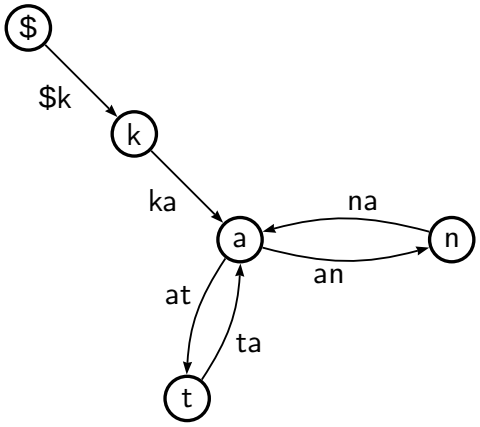

Fig. 1. A de Bruijn graph corresponding to the string $katan.

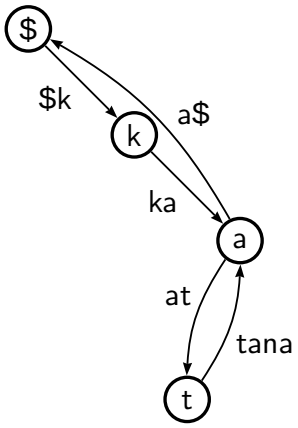

Fig. 2. A uniquely-decodable de Bruijn graph corresponding to the string $katana$.

probability $p > 0.5$, then we can expect each node in the de Bruijn graph of length l shingles to have only one outgoing edge (implying unique decodability) if

$$l \leq n + 1 + \frac{W\left(-\ln(p)p^{-n}\right)}{\ln p}, \tag{5}$$

where $W(\cdot)$ is the Lambert W function. When n goes to infinity, then (5) is $O(\log(n))$, meaning that logarithmically sized shingles should avoid communicationally expensive merges.

5) Step 6: The resulting collection of shingles can only be decoded in one way, which can be provided by any efficient algorithm for generating an Eulerian cycle through the graph.

C. Example

We next present an example of Protocol 1 in action.

Consider two remote hosts, one with the string $\sigma = \mathsf{katana}$ and the other with the string $\tau = \mathsf{kanatas}$. In Step 1 of the protocol, the two hosts produce the set of length $l = 2$ shingles

$$S_\sigma = \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{ta}, \mathsf{an}, \mathsf{na}, \mathsf{a}\$\}$$

and

$$S_\tau = \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{ta}, \mathsf{an}, \mathsf{na}, \mathsf{as}, \mathsf{s}\$\}\,.$$

In Step 2, both hosts utilize a set-reconciliation protocol to reconcile their shingle sets. At the end of this step, the first host knows S_τ and the second host knows S_σ.

In Step 3, the first host sets $S_\sigma^0 = \{\$\mathsf{k}\}$, corresponding to its first shingle. Then, in Step 4, the host extends this until it reaches a non-uniquely decodable multiset:

$$\begin{aligned} S_\sigma^1 &= \{\$\mathsf{k}, \mathsf{ka}\} \\ S_\sigma^2 &= \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}\} \\ S_\sigma^3 &= \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{ta}\} \\ S_\sigma^4 &= \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{ta}, \mathsf{an}\} \\ S_\sigma^5 &= \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{ta}, \mathsf{an}, \mathsf{na}\} \end{aligned}$$

To notice that S_σ^5 is not uniquely decodable, the host considers the de Bruijn graph in Figure 1 and runs the NFA described in Section IV on the corresponding alphabet $\{\$, \mathsf{k}, \mathsf{a}, \mathsf{t}, \mathsf{n}\}$ for the prefix $\$\mathsf{katana}$ formed from an arbitrary Eulerian path through the graph (starting at $\$$).

Correcting the non-unique decodability of S_σ^5 involves merging shingles an and na into ana, and then again with shingle ta into tana. The resulting shingles

$$S_\sigma^5 = \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{tana}\}$$

are uniquely decodable, and Step 4 continues without further merges to produce

$$S_\sigma^6 = \{\$\mathsf{k}, \mathsf{ka}, \mathsf{at}, \mathsf{tana}, \mathsf{a}\$\}.$$

In Step 5, the host must communicate the fact that it had merged an and na into the shingle ana, and then merged ta into ana to get tana. This involves transmitting the indices of an and ta in the alphabetical ordering of S_σ :

$$[\$\mathsf{k}, \mathsf{a}\$, \mathbf{an}, \mathsf{at}, \mathsf{ka}, \mathsf{na}, \mathbf{ta}];$$

as such, the host transmit the integers 2 and 6.

Finally, the remote host produces a de Bruijn graph of the shingle set S_σ^6, with vertices corresponding to a length $l - 1 = 1$ prefix and suffix of each shingle, as in Figure 2. There is only one decoding of this de Bruijn graph, namely the string $\$\mathsf{katana}\$$.

D. Communication Complexity

Only Steps 2 and 5 in Protocol 1 transmit data.

For two strings of length n differing in α edits, Step 2 will require $O(\alpha\ l^2)$ bits of communication for the implementation parameter l. Step 5 will require between 0 and $n\log(n-l+1)$ communication, depending on the decodability of the string.

When the two strings are composed of random iid bits, then, under the appropriate choice of l from (5), we can expect that no merging is needed giving an overall communication complexity that is $O\left(\alpha\log^2(n)\right)$, for large n.

VI. Conclusion

We have provided a novel algorithm for string reconciliation by combining an existing approach based on transformation to set reconciliation with an efficient means for testing the unique decodability of a string. In the best case and in certain random cases, our approach provides a computationally efficient and nearly communicationally optimal protocol for string reconciliation, although we leave open a precise categorization of when or how often this best case appears in practical situations.

Acknowledgment

This work was supported in part by NSF CCF-0916892.

References

[1] Sachin Agarwal, Vikas Chauhan, and Ari Trachtenberg. Bandwidth efficient string reconciliation using puzzles. *IEEE Trans. Parallel Distrib. Syst.*, 17(11):1217–1225, 2006.

[2] Richard Arratia, Béla Bollobás, Don Coppersmith, and Gregory B. Sorkin. Euler circuits and dna sequencing by hybridization. *Discrete Applied Mathematics*, 104(1 - 3):63 – 96, 2000.

[3] A. Broder. On the resemblance and containment of documents. *Compression and Complexity of Sequences*, June 1997.

[4] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[5] Mark Chaisson, Pavel A. Pevzner, and Haixu Tang. Fragment assembly with short reads. *Bioinformatics*, 20(13):2067–2074, 2004.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C.F. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[7] G. Cormode, M. Paterson, S.C. Sahinalp, and U. Vishkin. Communication complexity of document exchange. In *SODA*, pages 197–206, 2000.

[8] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[9] Martin Dyer, Alan Frieze, and Stephen Suen. The probability of unique solutions of sequencing by hybridization. *Journal of Computational Biology*, 1(2):105–110, Summer 1994.

[10] A.V. Evfimievski. A probabilistic algorithm for updating files over a communication link. *Theoretical Computer Science*, pages 191–199, 2000.

[11] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

[12] M. Karpovsky, L. Levitin, and A. Trachtenberg. Data verification and reconciliation with generalized error-control codes. *39th Annual Allerton Conference on Communication, Control, and Computing*, October 2001.

[13] Aryeh (Leonid) Kontorovich and Ari Trachtenberg. Unique decodability for string reconciliation. submitted.

[14] Leonid Kontorovich. Uniquely decodable n-gram embeddings. *Theor. Comput. Sci.*, 329(1-3):271–284, 2004.

[15] Qiang Li and Huimin Xie. Finite automata for testing composition-based reconstructibility of sequences. *J. Comput. Syst. Sci.*, 74(5):870–874, 2008.

[16] Y. Minsky and A. Trachtenberg. Scalable set reconciliation. In *Proc. 40-th Allerton Conference on Comm., Control, and Computing*, Monticello, IL., October 2002.

[17] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Trans. on Info. Theory*, September 2003.

[18] A. Orlitsky. Interactive communication: Balanced distributions, correlated files, and average-case complexity. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, pages 228–238, 1991.

[19] A. Orlitsky and K. Viswanathan. Practical protocols for interactive communication. In *IEEE International Symposium on Info. Theory*, June 2001.

[20] Franco P. Preparata and Eli Upfal. Sequencing-by-hybridization at the information-theory bound: An optimal algorithm. *Journal of Computational Biology*, 7(3-4):621–630, August 2000.

[21] D. E. Rumelhart and J. L. McClelland. On learning past tenses of english verbs. In *Parallel Distributed Processing: Vol 2: Psychological and Biological Models*, pages 216–271. MIT press, 1986.

[22] Xiaoli Shi, Huimin Xie, Shuyu Zhang, and Bailin Hao. Decomposition and reconstruction of protein sequences: The problem of uniqueness and factorizable language. *Journal of the Korean Physical Society*, 50(1I):118–123, 2007.

[23] Torsten Suel, Patrick Noel, and Dimitre Trendafilov. Improved file synchronization techniques for maintaining large replicated collections over slow networks. In *ICDE*, pages 153–164, 2004.

[24] A. Tridgell. *Efficient algorithms for sorting and synchronization*. PhD thesis, The Australian National University, 2000.