Random Projection and Geometrization of String Distance Metrics

Daniel Devatman Hromada
Université Paris 8 – Laboratoire Cognition Humaine et Artificielle
Slovak University of Technology – Faculty of Electrical Engineering and Information Technology
hromi@giver.eu

Abstract

Edit distance is not the only approach how distance between two character sequences can be calculated. Strings can be also compared in somewhat subtler geometric ways. A procedure inspired by Random Indexing can attribute an D-dimensional geometric coordinate to any character N-gram present in the corpus and can subsequently represent the word as a sum of N-gram fragments which the string contains. Thus, any word can be described as a point in a dense N-dimensional space and the calculation of their distance can be realized by applying traditional Euclidean measures. Strong correlation exists, within the Keats Hyperion corpus, between such cosine measure and Levenshtein distance. Overlaps between the centroid of Levenshtein distance matrix space and centroids of vectors spaces generated by Random Projection were also observed. Contrary to standard non-random "sparse" method of measuring cosine distances between two strings, the method based on Random Projection tends to naturally promote not the shortest but rather longer strings. The geometric approach yields finer output range than Levenshtein distance and the retrieval of the nearest neighbor of text's centroid could have, due to limited dimensionality of Randomly Projected space, smaller complexity than other vector methods.

Mèδεις ageôμετρèτος eisitô μου tèή stegèή

1 Introduction

Transformation of qualities into still finer and finer quantities belongs among principal hallmarks of the scientific method. In the world where even "deep" entities like "word-meanings" are quantified and co-measured by ever-growing number of researchers in computational linguistics (Kanerva et al., 2000; Sahlgren, 2005) and cognitive sciences (Gärdenfors, 2004), it is of no surprise that "surface" entities like "character strings" can be also compared one with another according to certain metric.

Traditionally, the distance between two strings is most often evaluated in terms of edit distance (ED) which is defined as the minimum number of operations like insertion, deletion or substitution required to change one string-word into the other. A prototypical example of such an edit distance approach is a so-called Levenshtein distance (Levenshtein, 1966). While many variants of Levenshtein distance (LD) exist, some extended with other operations like that of "metathese" (Damerau, 1964), some exploiting probabilist weights (Jaro, 1995), some introducing dynamic programming (Wagner & Fischer, 1974), all these ED algorithms take as granted that notions of insertion, deletion etc. are crucial in order to operationalize similarity between two strings.

Within this article we shall argue that one can successfully calculate similarity between two strings without taking recourse to any edit operation whatsoever. Instead of discrete insert&delete operations, we shall focus the attention of the reader upon a purely positive notion, that of "occurence of a part within the whole" (Harte, 2002). Any string-to-be-compared shall be understood as such a whole and any continuous N-gram fragment observable within it shall be interpreted as its part.

2 Advantages of Random Projection

Random Projection is a method for projecting high-dimensional data into representations with less dimensions. In theoretical terms, it is founded on a Johnson-Lindenstrauss (Johnson & Lindenstrauss, 1984) lemma stating that *a small set of points in a high-dimensional space can be* *embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved*. In practical terms, solutions based on Random Projection, or a closely related Random Indexing, tend to yield high performance when confronted with diverse NLP problems like synonym-finding (Sahlgren & Karlgren, 2002), text categorization (Sahlgren & Cöster, 2004), unsupervised bilingual lexicon extraction (Sahlgren & Karlgren, 2005), discovery of implicit inferential connections (Cohen et al., 2010) or automatic keyword attribution to scientific articles (El Ghali et al., 2012). RP distinguishes itself from other word space models in at least one of these aspects:

1. Incremental: RP allows to inject on-the-fly new data-points (words) or their ensembles (texts, corpora) into already constructed vector space. One is not obliged to execute heavy computations (like Singular Value Decomposition in case of Latent Semantic Analysis) every time new data is encountered.
2. Multifunctional: As other vector-space models, RP can be used in many diverse scenarios. In RI, for example, words are often considered to be the terms and sentences are understood as documents. In this article, words (or verses) shall be considered as documents and N-gram fragments which occur in them shall be treated like terms.
3. Generalizable: RP can be applied in any scenario where one needs to encode into vectorial form the set of relations between discrete entities observables at diverse levels of abstraction (words / documents, parts / wholes, features / objects, pixels/images etc.).
4. Absolute: N-grams and terms, words and sentences, sentences and documents – in RP all these entities are encoded in the **same** *randomly constructed yet absolute space* . Similarity measurements can be therefore realized even among entities which would be considered as incommensurable in more traditional approaches[1].

There is, of course, a price which is being paid for these advantages: Primo, RP involves stochastic aspects and its application thus does not guarantee replicability of results. Secundo, it involves two parameters D and S and choice of such parameters can significantly modify model's performance (in relation to corpus upon which it is applied). Tertio: since even the most minute "features" are initially encoded in the same way as more macroscopic units like words, documents or text, i.e. by a vector of length D "seeded" with D-S non-zero values, RP can be susceptible to certain limitations if ever applied on data discretisable into millions of distinct observable features.

3 Method

The method of geometrization of strings by means of Random Projection (RP) consists of four principal steps. Firstly, strings contained within corpus are "exploded" into fragments. Secondly, a random vector is assigned to every fragment according to RP's principles. Thirdly, the geometric representation of the string is obtained as a sum of fragment-vectors. Finally, the distance between two strings can be obtained by calculating the cosine of an angle between their respective geometric representations.

3.1 String Fragmentation

We define the fragment F of a word W having the length of N as any continuous[2] 1-, 2-, 3-...N-gram contained within W. Thus, a word of length 1 contains 1 fragment (the fragment is the word itself), words of length 2 contain 3 fragments, and, more generally, there exist N(N+1)/2 fragments for a word of length N. Pseudo-code of the fragmentation algorithm is as follows:

```
function fragmentator;
for  frag_length (1..word_length) {
    for offset (0..(word_length - frag_length)) {
      frags[]=substr (word,offset,frag_length);
    }
  }
```

where substr() is a function returning from the string *word* a fragment of length *frag_length* starting at specified *offset*.

[1] In traditional word space models, words are considered to be represented by the rows (vectors/points) of the word-document matrix and documents to be its columns (axes). In RP, both words (or word-fragments) and documents are represented by rows.

[2] Note that in this introductory article we exploit only continuous N-gram fragments. Interaction of RP with possibly other relevant patterns observable in the word – like N-grams with gaps or sequences of members of diverse equivalence classes [e.g. consonants/vowels] – shall be, we hope, addressed in our doctoral Thesis or other publications.

3.2 Stochastic fragment-vector generation

Once fragments are obtained, we transform them into geometric entities by following the fundamental precept of Random Projection:

To every fragment-feature F present in the corpus, let's assign a random vector of length containing D-S elements having zero values and S elements whose value is either -1 or 1.

The number of dimensions (D) and the seed (S) are the parameters of the model. It is recommended that S<<D. Table 1 illustrates how all fragments of the corpus containing only a word[3] "DOG" could be, given that S=2, randomly projected in a 5-dimensional space.

Fragment	Vector
D	0, 1, 0, 0, -1
O	1, 1, 0, 0, 0
G	0, 0, -1, 0, -1
DO	-1, 0, -1, 0, 0
OG	0, 1, 0, 1, 0
DOG	0, 0, 0, -1, -1

Table 1: Vectors possibly assigned to the fragments of the word "dog" by $RP_{5,2}$

3.3 String geometrization

Once random "init" vectors have been assigned to all word-fragments contained within the corpus, the geometrization of all word-strings is relatively straightforward by applying the following principle:

The vector representation of a word X can be calculated as a sum of vectors associated to fragments contained in the word X.

Thus, the vector representation of a word "dog" would be [0, 3, -2, 0, -3]. Note also that this vector for the word "dog" is different from randomly initialized fragment-vector referring to the fragment "dog". This is due to the fact that the vector space of "fragments" and "words" are two different spaces. One possible way how could one can collapse the fragment space with the string space is to convolute them by Reflected Random Indexing (Cohen et al., 2010) – such an approach, however, shall not be applied in a limited scope of this article.

3.4 String distance calculation

The string geometrization procedure calculates a vector for every string present in the corpus. Subsequently, the vectors can be compared with each other. While other measures like Jaccard index are sometimes also applied in relation to RI, the distance between words X and Y shall be calculated, in the following experiment, in the most traditional way. Id est, as a cosine of an angle between vectors V_X and $V_{Y.}$

4 Experiment(s)

Two sets of simulations were conducted to test our hypothesis. The first experiment looked for both correlations as well as divergences between three different word-couple similarity data-sets obtained by applying three different measures upon the content of the corpus. The second experiment focused more closely upon overlaps among the centroids of three diverse metric spaces under study.

4.1 Corpus and word extraction

ASCII-encoded version of the poem "The Fall of Hyperion" (Keats, 1819) was used as a corpus from which the list of words was extracted by

1. Splitting the poem into lines (verses).
2. Splitting every verse into words, considering the characters [:;,.?!()] as word separator tokens.
3. In order to mark the word boundaries, every word was prefixed with ^ sign and post-fixed with **$** sign.
4. All words were transformed into lowercase.

Corpus has size of 22812 bytes representing 529 lines which contain the total number of N_w=1545 distinct word types exploded into N_F=22340 distinct fragments.

4.2 "Word couple" experiment

Three datasets were created, all containing the list of all possible (i.e. N_w * N_w = (1545 * 1545) / 2 =1193512) distinct word-couples. For every dataset, a string distance was calculated for every word couple. Within the first dataset, the distance was determined according to traditional Levenshtein distance metrics. For second dataset, an RPD distance has been calculated by measuring word couple's cosine distance within the vector space constructed by Random Projection of words fragments set up with parameters D=1000,S=5. The third dataset contains values obtained by measuring the cosine measure between two sparse non-random vector representations of two different words , whereby the features were obtained by means of the same fragmentation algorithm as in the case of RPD,

[3] The role of fragment is analogical to the role of a "term" in Random Indexing. And the role of the "word" is identical to the role that "context" plays in RI.

but without Random Projection. In order to keep this scenario as pure as possible, no other processing (e.g. tf-idf etc.) was applied and the values which we shall label as „geometric distance“ (GD) denote simply the cosine between two vectors of a non-stochastically generated sparse fragment-word count matrix.

4.2.1 Results

Figure 1 shows relations between LD and RPD distances of all possible couples of all words contained in the Hyperion corpus. Both datasets seem to be strongly significantly corellated both according to Spearman‘s rho measure (p < 2.2e-16) as well as according to Pearson‘s product-moment correlation (p < 2.2e-16, cor = -0.2668235). While fifteen different LDs from the range of integers <0, 15> were observed among words of corpus, one could distinguish 252229 diverse real-numbered RPD values limited to interval <0, 1>.

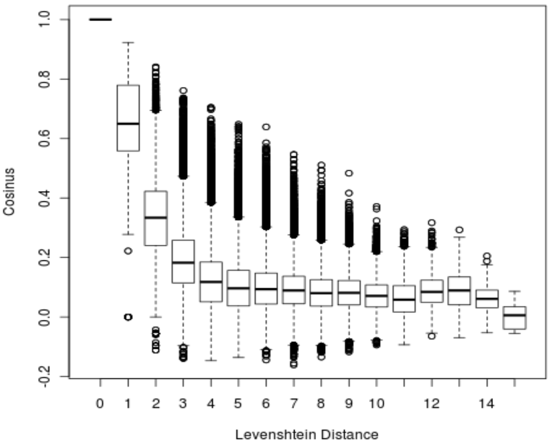

Figure 1: Scatter plot displaying relations between Levenshtein distances and cosine distances measured in the vector space constructed by $RI_{1000,5}$

String distance measured in the space constructed by **RP$_{1000,5}$** also strongly correlates (Pearson correlation coefficient = 0.992; Spearman rho = 0.679; minimal p < 2.2e-16 for both tests) with a GD cosine measure exploiting a non-deformed fragment-word matrix.

An important difference was observed, however, during a more „local“ & qualitative analysis of results produced by the two vectorial methods. More concretely: while non-stochastic „sparse“ cosine GD distance tends to promote as „closest“ the couples of **short** strings, *RPD yields the highest score for couples of* ***long*** *words*. This is indicated by the list of most similar word-couples generated by three methods present in Table 2.

GD	RPD
a ’	vessels vessel
it i	comfort comforts
i ’	sorrows sorrow
at a	’benign benign
o so	temples temple
o of	changing unchanging
as a	stream streams
o or	immortal’s immortal
’i i	breathe breath
an a	trance tranced

Table 2: Ten most similar world couples according to non-random “sparse” geometric distance (GD) and Randomly Projected Distance

4.3 The “centroid” experiment

Three types of concrete word-centroids were extracted from the corpus. A string having the smallest overall LD to all other strings in the corpus shall be labeled as the “Levenshtein centroid” (LC). A string having the maximal sum of cosines in relation to other words shall be labeled as the “Cosinal centroid” (CC). Contrary to LC and CC, for calculation of which one has to calculate distances with regard to all words in the corpus, the “Geometric Centroid” (GC) was determined as a word whose vector has the biggest cosine in regard to “Theoretical Centroid” (GC) obtained in a purely geometric way as a sum of all word-vectors. Stochastic CC_{RP} and GC_{RP} calculation simulations were repeated in 100 runs with D=1000, S=5.

4.3.1 Results

The word “**are**” was determined to be the LC of Hyperion corpus with average $LD_{ARE,X}$ = 4.764 to all words of the corpus. The same word are was ranked, by a non-stochastic “sparse” geometric distance algorithm, as 3rd most central CC and 36th most closest term to GC . Table 3 shows ten terms with least overall LD to all other words of the corpus (LC), ten terms with biggest cosine in relation to all other terms of the corpus (CC_{GD})

and ten terms with biggest cosine in regard to hypothetical Theoretical Centroid (GC$_{\mathrm{GD}}$) of a sparse non-projected space obtained from the Hyperion corpus.

Rank	**LC**	**CC$_{\mathrm{GD}}$**	**GC$_{\mathrm{GD}}$**
1	**are**	charm	**a**
2	**ore**	**red**	o
3	ate	arm	I
4	**ere**	**a**	‘
5	one	**me**	he
6	toes	hard	to
7	sole	had	at
8	ease	reed	an
9	lone	domed	**me**
10	here	**are**	as

Table 3: Ten nearest neighbor words of three types of non-stochastic centroids

Shortest possible strings seem to be GC$_{\mathrm{GD}}$’s nearest neighbors. This seems to be analogous to data presented on Table 2. In this sense does the GC$_{\mathrm{GD}}$ method seem to differ from the CC$_{\mathrm{GD}}$ approach which tends to promote longer strings.

Such a marked difference in behaviors between GC and CC approaches was not observed in case of spaces constructed by means of Random Projection. In 100 runs, both GC and CC centered approaches seemed to promote as central the strings of comparable content and length[4]. As is indicated by Table 4, the LC “are” turned out to be the closest (i.e. Rank 1, when comparing with Table 3) to all other terms in 6% of Random Projection runs. In 6% of runs the same term was labeled as the nearest neighbor to the geometrical centroid of the generated space. Other overlaps between all used methods are marked by bold writing in Tables 3 and 4.

Word	**CC$_{\mathrm{RPD}}$**	**GC$_{\mathrm{RPD}}$**
see	20	28
he	11	8
are	6	6
ore	5	6
ere	4	5
set	6	5
she	5	4
sea	4	4
a	9	4
red	1	3

Table 4: Central terms of Randomly Projected spaces and their frequency of occurence in 100 runs

Analogically to the observation described in the last paragraph of the section 4.2.1, it can be also observed that the strings characterized as “closest” to the Theoretical Centroid of vector spaces generated by Random Projection tend to be longer than “minimal” string nearest to GC$_{\mathrm{GD}}$ determined in the traditional non-stochastic feature-word vector space scenario.

5 Discussion

When it comes to CC$_{\mathrm{RP}}$-calculation run lasted, in average, CC$_{\mathrm{RPD\text{-}detection}}$ = 90 seconds, thus being almost twice as fast than the LC-calculation executed on the very same computer which lasted twice the time LC$_{\mathrm{detection}}$= 157 s for the same corpus, indicating that the computational complexity of our PDL (Glazebrook et al., 1997) implementation of CC$_{\mathrm{RP}}$-detection is lesser than the complexity of LC-detection based on PERL’s Text::Levenshtein implementation of LD.

When it comes to the computational complexity of the GC-calculation, it is evident that GC is determined faster and by less complex process than LCs or CCs . This is so because in order to determine the GC$_{\mathrm{RP}}$ of N words there is no need to construct an N * N distance matrix. On the contrary, since every word is attributed coordinates in a randomly-generated yet *absolute* space, the detection of a hypothetic Geometric Centroid of all words is a very straightforward and cheap process, as well as the detection of GC’s nearest word neighbor..

And since in RP, the length of GC-denoting vector is limited to a relatively reasonable low number of elements (i.e. D = 1000 in case of this paper), it is of no surprise that the string closest to GC shall be found more slowly by a traditional “sparse vector” scenario whenever the number of features (columns) > D. In our scenario with N$_{\mathrm{F}}$=22340 of distinct features, it was almost 4 times faster to construct the vector space + find a nearest word to GC of the Randomly Projected space han to use a “sparse” fragment-term matrix optimized by storing only non-zero values (GC$_{\mathrm{RPD\text{-}NN\text{-}detection}}$ ~ 6 sec ; GC$_{\mathrm{GD\text{-}NN\text{-}detection}}$ ~ 22 sec).

Other thing worthy of interest could be that contrary to a “sparse” method which seems to give higher score to shorter strings, somewhat longer strings seem to behave as if they were naturally “pushed towards the centroid” in a dense space generated by RP. If such is, verily, the case, then we believe that the method presented hereby could be useful, for example, in domains of gene sequence analysis or other scenarios where pattern-to-be-discovered is “spread out” rather than centralized.

[4] In fact only in 22 runs did GC$_{\mathrm{RPD}}$ differ from CC$_{\mathrm{RPD}}$

In practical terms, if ever the querying in RP space shall turn out to have lesser complexity than other vector models, our method could be useful within a hybrid system as a fast stochastic way to pre-select a limited set of “candidate” (possibly locally optimal) strings which could be subsequently confronted with more precise, yet costly, non-stochastic metrics ultimately leading to discovery of the global optimum.

Asides above-mentioned aspects, we believe that there exists at least one other theoretical reason for which the RP-based geometrization procedure could deem to be a worthy alternative to LD-like distance measures. That is: the cardinality of a real-valued <0, 1> range of a cosine function is much higher than a whole-numbered <0, max(length(word))> range possibly offered as an output of Levenshtein Distance. In other terms, outputs of string distance functions based on trigonometry of RP-based vector spaces are more subtler, more fine-grained, than those furnished by traditional LD. While this advantage does not hold for “weighted” LD measures we hope that this article could motivate future studies aiming to compare “weighted” LD and RPD metrics.

When it comes to the feature extracting “fragment explosion” approach, it could be possibly reproached to the method proposed hereby that 1) the fragmentation component which permutes blindly through all N-grams presented in the corpus yields too many “features”; that 2) that taking into account all of them during the calculation of the word’s final vector is not necessary and could even turn to be computationally counter-productive; or that 3) bi-grams and tri-grams alone give better results than larger N (Manning et al., 2008). A primary answer to such an ensemble of reproaches could be, that by the very act of projecting data upon limited set of same non-orthogonal dimensions, the *noise could simply cancel itself out*[5]. Other possible answer to the argument could be that while the bi&tri-gram argument holds well for natural language structures, the method we aim to propose here has ambitions to be used beyond NLP (e.g. bio-informatics) or pre-NLP (e.g. early stages of language acquisition where the very notion of N-gram does not make sense because the very criterion of sequence segmentation & discretization was not yet established). At last but not least we could counter-argue by stating that often do the algorithms based on a sort of initial blind “computational explosion of number of features” perform better than those who do not perform such explosion, especially when coupled with subsequent feature selection algorithms. Such is the case, for example, of an approach proposed by Viola & Jones in (Viola & Jones, 2001) which caused the revolution in the computer vision by proposing that in order to detect an object, one should look for combinations of pixels instead of pixels.

In this paper, such combinations of “letter-pixels” were, *mutatis mutandi*, called “fragments”. Our method departs from an idea that one can, and should, associate random vectors to such fragments. But the idea can go further. Instead of looking for occurrence of part in the whole, a more advanced RI-based approach shall replace the notion of “fragment occuring in the word” by a more general notion of “pattern which matches the sequence”. Thus even the vector associated to pattern /d.g/ could be taken into account during the construction of a vector representing the word “dog”.

Reminding that RP-based models perform very well when it comes to offering solutions to quite “deep” *signifiée*-oriented problems, we find it difficult to understand why could not be the same algorithmic machinery applied to the problems dealing with “surface”, *signifiant*-oriented problems, notably given the fact that some sort of dimensionality reduction has to occur whenever the mind tries to map >4D-experiences upon neural substrate of the brain embedded in 3D physical space.

Given that all observed correlations and centroid overlaps indicate that the string distance calculation based on Random Projection could turn out to be a useful substitute for LD measure or even other more fine-grained methods. And given that RP would not be possible if the Johnson-Lindenstrauss’s lemma was not valid, our results could be also interpreted as another empirical demonstration of the validity of the JL-lemma.

Acknowledgments

The author would like to thank Adil El-Ghali for introduction into Random Indexing as well as his comments concerning the present paper; to prof. Charles Tijus and doc. Ivan Sekaj for their support and to Aliancia Fair-Play for permission to execute some code on their servers.

[5] And this “noise canceling property” could be especially true for RP as defined in this paper where the rare non-zero values in the random “init” vectors can point in opposite directions (i.e. either -1 or 1).

References

Trevor Cohen, Roger Schvaneveldt & Dominic Widdows. 2010. Reflective Random Indexing and indirect inference: A scalable method for discovery of implicit connections. *Journal of Biomedical Informatics*, *43*(2), 240–256.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176.

Adil El Ghali, Daniel D. Hromada & Kaoutar El Ghali. 2012. Enrichir et raisonner sur des espaces sémantiques pour l'attribution de mots-clés. *JEP-TALN-RECITAL 2012*, 77.

Peter Gärdenfors. 2004. *Conceptual spaces: The geometry of thought*. MIT press.

Karl Glazebrook. Jarle Brinchmann, John Cerney, Craig DeForest, Doug Hunt, Tim Jenness & Tuomas Lukka. 1997. The Perl Data Language. *The Perl Journal*, *5*(5).

Verity Harte. 2002. *Plato on parts and wholes: The metaphysics of structure*. Clarendon Press Oxford.

Matthew A. Jaro. 1995. Probabilistic linkage of large public health data files. *Statistics in medicine*, *14*(5-7), 491–498.

William B. Johnson & Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, *26*(189-206), 1.

Pentti Kanerva, Jan Kristofersson & Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. *Proceedings of the 22nd annual conference of the cognitive science society* (Vol. 1036).

John Keats. 1819. The Fall of Hyperion. A Dream. *John Keats. complete poems and selected letters*, 381–395.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady* (Vol. 10, p. 707).

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini & Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, *2*, 419–444.

Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge University Press.

Magnus Sahlgren. 2005. An introduction to random indexing. *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE* (Vol. 5).

Magnus Sahlgren & Rickard Cöster. 2004. Using bag-of-concepts to improve the performance of support vector machines in text categorization. *Proceedings of the 20th international conference on Computational Linguistics* (p. 487).

Magnus Sahlgren & Jussi Karlgren. 2002. Vector-based semantic analysis using random indexing for cross-lingual query expansion. *Evaluation of Cross-Language Information Retrieval Systems* (p. 169–176).

Magnus Sahlgren & Jussi Karlgren. 2005. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Natural Language Engineering*, *11*(3), 327–341.

Alan M. Turing. 1936. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, *42*(2), 230–265.

Paul Viola & Michal Jones. 2001. Rapid Object Detection using a Boosted Cascade of Simple. *Proc. IEEE CVPR 2001*.

Robert A. Wagner & Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, *21*(1), 168–173.