Proceedings Paper DOI: 10.58190/icat.2023.51

PROCEEDINGS OF

INTERNATIONAL CONFERENCE ON ADVANCED TECHNOLOGIES

https://proceedings.icatsconf.org/

11th International Conference on Advanced Technologies (ICAT'23), Istanbul-Turkiye, August 17-19, 2023.

An Investigation of the Usage of Dynamic Time Warping in String Similarity Estimation

Mehmet Ali Özer [1], Emre Kaplan[1], Can Özbey[1], Meltem Çetiner[1], Ekin Can Erkuş[1]

[1] *Intelligent Application Department DC, Huawei Technologies Turkey R&D Center Istanbul, Turkey*

mehmet.ali.ozer@huawei.com, ORCID: 0000-0003-2254-0254
emre.kaplan@huawei.com, ORCID: 0009-0008-5445-3079
can.ozbey1@huawei.com, ORCID: 0009-0005-8432-9413
meltem.cetiner1@huawei.com, ORCID: 0000-0001-5026-0642
ekin.can.erkus2@huawei.com, ORCID: 0000-0002-2445-5929

***Abstract*— String similarity estimation is important in many fields, including natural language processing, information retrieval, and data mining. Dynamic Time Warping (DTW) has emerged as a widely used technique for measuring sequence similarity, effectively accommodating variations in length and temporal distortions. This paper presents an examination of the use of DTW in string similarity estimation. We delve into the adoption of DTW in string similarity estimation in various contexts, such as approximate string matching and spelling correction. We investigate DTW's strengths and limitations through empirical analysis, particularly in capturing complex patterns and variations within strings, while taking into account factors such as adaptability and robustness. Furthermore, we discuss the impact of various traditional similarity metrics in comparison with DTW based on their evaluation on two different experimental settings. The findings of this study provide important insights into the effectiveness and challenges of using DTW in string similarity estimation. This work may open up alternative ways for the development of more adaptive string similarity estimation techniques through the use of DTW.**

***Keywords*— string similarity, dynamic time warping, similarity estimation, distance measures, approximate string matching**

I. INTRODUCTION

String similarity estimation plays important role in many fields such as natural language processing, information retrieval, text classification, automatic question-answering, data mining, bioinformatics [1][2]. Text-based studies have at least two essential steps which are to represent the text and to measure the representations to be able to compare. Traditional techniques have focused on common words for measuring similarity between documents [1]. Cosine similarity applied on TF-IDF representations [3] as well as set intersection measures, e.g., Jaccard similarity, which can also be applied on keywords [4], are particularly effective when texts are more likely to have common words. However, in some cases, the number of shared words among documents happens to be limited, and thus a latent representation is needed to capture similarity between documents. For that matter, vector space models [5], e.g., Latent Semantic Analysis (LSA) [6][7], doc2vec [8], GloVe [9], BERT [10], etc., stand out since they are able to encode semantic information in a reduced vector space. In this respect, graph-based models are also used as they may capture latent relationships between documents [11], [12].

Similarity measures are not only applied on documents, but also on keywords especially in information retrieval [13], [14]. Depending on the purpose, there exist many different string similarity estimation algorithms [15]. In this paper, we aim to adapt Dynamic Time Warping (DTW) to the problem of measuring structural similarity between strings. Conventionally, DTW is mostly used for sequence alignment in specific domains such as speech recognition [16], bioinformatics [17], time series analysis [18], etc., and to our knowledge, it has been relatively unexplored as a string similarity measure for keywords to be compatible with popular metrics. In [19], a sentence-level DTW is applied to measure similarity between texts, which concerns with the parts of speech and word order. Likewise in [20], the authors considered the smallest unit in sequences to be words, and thus compute distance between documents. Our motivation is to adopt DTW such that it will be possible to calculate a degree of similarity between two keywords in a similar manner to common string similarity metrics. In the rest of the paper, we first explain how to transform DTW to a similarity measure by constructing a character-level binary cost matrix, then we briefly describe some common string similarity metrics to compare with our method, and lastly we conduct experiments on two different dataset to comparatively evaluate the proposed method.

II. METHODS

A. Dynamic Time Warping (DTW)

DTW aims to find an optimal alignment between two independent sequences $X = (x_1, x_2, ..., x_N)$ and $Y = (y_1, y_2, ..., y_M)$ of length $N \in N$ and $M \in N$ respectively, such that the sum of distances between the aligned points will be a global minimum [21]. More formally, its objective can be formulated as follows:

$$DTW(X,Y) = min\left(\sum_{(i,j)\in A} d(X_i, Y_i)\right) \quad (1)$$

Here, *A* refers to the set of all possible temporal paths that can be constructed between *X* and *Y*. The distance function *d(X, Y)* yields the distance between any given pair of elements from the two sequences. The optimal path with the minimum total distance is found by dynamic programming.

B. Binary Encoding

DTW requires that a cost matrix be constructed from a given pair of strings.Here, we adopt the most basic scheme where a binary encoding is used to create a pairwise distance matrix between the single characters in the two strings. If two characters are the same, the distance is 0; otherwise the distance simply becomes 1. A distance matrix constructed by the words "proof" and "profe" is given as an example in Table I.

TABLE I

BINARY DISTANCE MATRIX

	p	**r**	**o**	**o**	**f**
p	0	1	1	1	1
r	1	0	1	1	1
o	1	1	0	0	1
f	1	1	1	1	0
e	1	1	1	1	1

The pairwise distance matrix can be used to infer the optimal DTW path. This path is a sequence of points that represents the optimal alignment with the minimum total cost between the two strings. The green coloured cells in Table II corresponds to the optimal path and the minimum total distance associated with the path is calculated as 2.

TABLE II

OPTIMAL DTW PATH

	w	**i**	**t**	**h**	**o**	**u**	**t**
w	0	1	1	1	1	1	1
h	1	1	1	0	1	1	1
i	1	0	1	1	1	1	1
t	1	1	0	1	1	1	0
o	1	1	1	1	0	1	1
u	1	1	1	1	1	0	1
t	1	1	0	1	1	1	0

C. Distance Normalization

If the given two strings are of different lengths, the minimum cost may not come out as an accurate measure of dissimilarity. For this reason, we optionally apply normalization in a similar manner to the Longest Common Substring (LCS) algorithm in order to obtain a standardized measure of distance. In this respect, we apply two different normalization schemes:

MaxLen Normalization: In MaxLen normalization process, the minimum total distance is divided by the longest length of the given strings.

PathLen Normalization: In PathLen normalization, the minimum total distance is divided by the total number of steps in the DTW path.

STRING SIMILARITY MEASURES

In this section, we describe different types of string similarity algorithms that we used in our experiments for comparison with DTW. Some of these algorithms require tokens for the computation, so we used character-based tokenization with different q values.

Character-based tokenization is a simple and straightforward way of tokenizing strings. It simply splits the string into individual characters, or pairs of characters if q is greater than 1. For example, tokens of the word "similar" with $q = 1$ are $[s, i, m, i, l, a, r]$ and tokens with $q = 2$ are $[si, im, mi, il, la, ar]$. We applied at most up to $q = 3$ for the intersection-based algorithms.

A. Longest Common Substring

The Longest Common Substring (LCS) [22] is the longest consecutive sequence of two or more texts. LCS can be used for a variety of tasks, including plagiarism detection [23] and data deduplication [24]. The normalised similarity between two strings is calculated by dividing the length of the common string by the length of the longest among the texts. This gives a measure of how similar the texts are.

B. Eudex

Eudex [25] is a phonetic hashing algorithm that computes the Hamming distance between two texts based on their phonetic mapping. The algorithm uses four tables created according to the International Phonetic Alphabet (IPA) to extract encodings. Two of the tables are used for ASCII and C1 (Latin Supplement) characters in the first position of the word, and the other two tables are used for the rest of the characters [26]. Each character is represented by 8-bits, and those that share more phonetic resemblance will have lower Hamming distances. The final similarity is calculated

by measuring the Hamming distances of all the characters by the prefixes of a given pair of strings.

C. Levenshtein Distance

The Levenshtein distance (Edit distance) [27] is a measure of a structural similarity between two strings, commonly used in natural language processing [28] and information retrieval tasks [13]. It is defined as the minimum number of editing operations required to transform string a into string b. The original editing operations, which include substitution, insertion, and deletion, were later extended to incorporate transpositions as well.

D. Jaccard

The Jaccard similarity index is calculated as the ratio of the common elements between two sets to the total number of elements in the union set. A high Jaccard index indicates a high degree of similarity. Jaccard index is calculated according to the formula:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{2}$$

where x and y are two sets. $|X \cap Y|$ is the number of elements common to both sets, while $|X \cup Y|$ is the total number of elements in the union of the two sets.

E. Ample

Ample is the measure of text similarity that calculates the similarity between two strings using set intersection, similar to the Jaccard similarity coefficient. The Ample formula is as follows:

$$A(x,y) = \left|\frac{b}{a+b} - \frac{c}{c+d}\right| \tag{3}$$

Let us consider two sets x and y. We compute b as the size of the intersection of the two, i.e., b = |x ∩ y|, and subsequently, a = |x| − b, c = |y| − b, and d = p − |x ∪ y| where p is calculated by:

$$p = |\Sigma|^q \tag{4}$$

Here, |Σ| refers to the number of distinct symbols of the alphabet in use, and q denotes the tokenization parameter explained previously. For example, with q = 2, p is calculated as 282 with the start token ($) and end token (#) added for values of q greater than or equal to 2. However, for q = 1, these symbols are not required. In this case, the number of possible tokens is calculated as 261, which is the length of the standard English alphabet.

The first term in the equation stands for a degree of similarity, and the second ratio that of dissimilarity. As a result, a higher Ample score indicates a higher similarity.

F. Jaro-Winkler

Jaro-Winkler [29] is an algorithm that computes similarity over edit distance between two texts. It is a variant of Jaro [30] and uses prefix size p and prefix length l in addition to Jaro similarity. The formula of the Jaro-Winkler similarity is

$$sim_w = sim_j + l * p * (1 - sim_j) \tag{5}$$

where

- sim_j is the Jaro similarity
- p is a constant scaling factor
- l is the length of the common prefix at the beginning of the string.

In this paper we used the p value of 0.1 which is the default value. The p value should not exceed 0.25, otherwise the similarity may become greater than 1.0.

The Jaro similarity formula is calculated as follows:

$$sim_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) \tag{6}$$

where

- s1 and s2 is the two strings being compared
- m is the number of shared characters
- t is the number of transpositions
- |s1| and |s2| are the lengths of the strings s1 and s2, respectively.

III. EXPERIMENTAL RESULTS

A. Datasets

1) Titler: One of the datasets we used in this study is the Titler [31] dataset. Titler is a dataset that contains links of some websites and titles related to these websites annotated by human assessors. Some examples of link/title pairs are given in Table III.

TABLE III

TITLER DATASET

Link	Title
http://www.thefox.co.nz/	The Fox
http://fourstarpizza.ie/	Four Star Pizza
http://chinesecricketclub.com/	Chinese Cricket Club
http://www.themezegrill.com/	Meze Grill
http://www.brazenhead.com/	The Brazen Head
http://www.sodopizza.co.uk/	Sodo Pizza Caf

We created 4 different scenarios for the titles in the data. These cases are:

- *Case1 - Lowercased*
- *Case2 - Lowercased, Removed vowels*
- *Case3 - Lowercased, Removed vowels and whitespaces*
- *Case4 - Lowercased, Removed whitespaces*

TABLE IV

CASES FOR TITLER DATA

Case1	Case2	Case3	Case4
the apollo	th pll	thpll	theapollo
aqua dining	aq dnng	aqdnng	aquadining
manta	mnt	mnt	manta
manta	mnt rstrnt	mntrstrnt	mantarestauran
piato	pt	pt	piato
piato restaurant	pt rstrnt	ptrstrnt	piatorestaurant

We illustrate how titles are transformed for each case through some examples as shown in Table IV. Note that we exclude leading vowels in vowel removal process.

Since the title mostly happens to be relevant to the information contained in the root of the URL, we construct two distinct collections of ground-truth items, namely, URL and Root URL. As shown by the following examples, the protocol information and the subdomain *(www)* are removed in both cases, while the second case only retains the root.

- *URL = coffee-express.com.au*
- *Root URL = coffee-express*

Considering all of the cases mentioned above, we will conduct a total of 8 different experiments to compare string similarity measures.

2) TOEFL: We used the TOEFL-Spell Corpus dataset based on the ETS Corpus of Non-Native Written English [32]. It contains spelling errors made by more than 6000 non-native speakers, but in this study, we have deduplicated the same misspellings to ensure that the same misspelled word does not match more than one correct word. After singularizing the data, we obtained 4056 misspellings and correct word matches. Some examples are given in Table V.

TABLE V

TOEFL DATA

Misspelling	Correction
writi	writing
beacuse	because
enviroment	environment
suceesfull	successful
bussiness	business
succesful	successful
comfront	confront

B. Setup

In our experiments, we consider the given link/title pairs in the Titler dataset to be the ground-truth. Then, after extracting all possible link/title pairs in the data, we compute all pairwise similarity scores using different approaches and take the highest as a prediction for each title. We then compare these predictions with the true labels and calculate an accuracy score for evaluation.

The Titler data contained some titles that are linked to same URLs. To improve the quality of our experiments, we separated these titles into their individual components before extracting the possible link/title pairs as illustrated below.

Link	Title
http://www.theapollo.com.au/	The Apollo — The Apollo

Link	Title
http://www.theapollo.com.au/	The Apollo
http://www.theapollo.com.au/	The Apollo Restaurant

As for the TOEFL data, we first extract all possible misspelling/correction pairs and compute their pairwise similarities by each approach. Then, to predict correction for each misspelling, we take the highest among the similarity scores as shown in Table VI for the Jaro-Winkler metric, in the same way as performed for the Titler dataset.

TABLE VI

JARO-WINKLER TOEFL SCORES

Misspelling	Correction	Jaro-Winkler (q=1)
writi	**writing**	**0.942857**
writi	within	0.840000
writi	citizen	0.676190
writi	erratic	0.676190
...	...	...
beacuse	**because**	**0.961905**
beacuse	beauties	0.869444
beacuse	Because	0.849206
beacuse	became	0.826032
...	...	...
enviroment	**environment**	**0.961818**
enviroment	environmental	0.933846
enviroment	environmentally	0.913333
enviroment	enjoyment	0.860741

C. Results

Table VII summarizes the comparative accuracy results between all models for both the Titler and the TOEFL datasets. For Titler data, we observe that Jaro-Winkler outperforms the other models in all cases, regardless of the q value. The models whose performance were closest to that of Jaro-Winkler turned out to be other set intersection-based algorithms, i.e., Ample and Jaccard, particularly for Case1 and Case4. It is notable that Jaro-Winkler algorithm is much more resistant to vowel removal than the other two. Furthermore, the normalized DTW and Levenshtein metrics were also found to be more robust to character removals than Ample, Jaccard, Eudex and LCS.

TABLE VII

SUMMARY OF RESULTS

Model	Titler								Toefl
	Url				Root Url				
	Case1	Case2	Case3	Case4	Case1	Case2	Case3	Case4	
LCS	0.56961	0.10103	0.15752	0.79157	0.56802	0.09706	0.16309	0.77566	0.47584
Eudex	0.65473	0.10899	0.10899	0.65473	0.68417	0.10581	0.10581	0.68417	0.44625
Levenshtein	0.70247	0.57518	0.60700	0.71758	0.71201	0.58075	0.60382	0.73111	0.87303
Ample (q=1)	0.00159	0.00080	0.00000	0.00557	0.59029	0.19411	0.30788	0.64757	0.76849
Ample (q=2)	0.80748	0.41209	0.48687	0.81543	0.79475	0.28719	0.35640	0.80907	0.86021
Ample (q=3)	0.82657	0.19173	0.26730	0.84885	0.79554	0.13126	0.22673	0.83134	0.86243
Jaccard (q=1)	0.30469	0.20764	0.19014	0.29674	0.47494	0.34288	0.31981	0.46698	0.56164
Jaccard (q=2)	0.79952	0.35163	0.41925	0.81464	0.81225	0.35084	0.41607	0.82339	0.74482
Jaccard (q=3)	0.82339	0.13683	0.21639	0.83850	0.81941	0.13126	0.21957	0.83532	0.61243
Jaro-Winkler (q=1)	0.79395	**0.72713**	**0.72633**	0.79952	0.82657	0.82657	**0.72554**	0.82737	0.87327
Jaro-Winkler (q=2)	**0.85680**	0.54813	0.59666	**0.86158**	**0.84964**	**0.84964**	0.57916	**0.85998**	0.86760
Jaro-Winkler (q=3)	0.85123	0.25298	0.34924	**0.86158**	0.83134	0.83134	0.35879	0.85362	0.86119
DTW	0.54336	0.11138	0.10263	0.56006	0.70565	0.70565	0.39300	0.71281	**0.94412**
DTW (MaxLen Norm)	0.70883	0.52983	0.55290	0.71758	0.73111	0.73111	0.59029	0.73986	0.90829
DTW (PathLen Norm)	0.71042	0.52824	0.54574	0.70644	0.73270	0.73270	0.60064	0.73588	0.90312

For the URL case in the Titler dataset, the Ample model with $q = 1$ seems to perform poorly. In the URL case, unlike the Root URL case, some links are very long, and consequently, the token count of these links are quite high including numbers and some punctuation. This causes the value of d in the expression of $A(x, y)$ to be negative, because p is calculated as 26 for $q = 1$. If we set $d = 0$ and continue the process as such, the similarity score between two strings becomes much higher than it should be leading to incorrect predictions. It should be noted that this situation only occurs for Ample ($q = 1$) in the URL case.

For the TOEFL data, we can observe that DTW outperforms all the other approaches with or without normalization. In contrast to the Titler dataset, normalization seems to reduce the performance of DTW. This is possibly due to the fact that the lengths of misspellings and target correct forms do not as significantly differ from each other as in the Titler dataset. Overall, the DTW-based string similarity metric has been found much more effective in a spelling correction task rather than partial string matching.

IV. CONCLUSION

In this study, we have developed a method for measuring string similarity using DTW. In this respect, we have simply applied a binary cost function, which have been found effective particularly in the spelling correction task through a series of experiments with several traditional string similarity metrics. The proposed method can also be preferred for its robustness to non-systematic deletion/insertion of characters as well as for its parameter-free nature in contrast to set intersection algorithms. While our method uses the binary encoding between strings to compute the DTW distance, it can be easily extended such that distance matrix of the input strings are constructed via a weighted scheme depending on the task. This aspect makes the proposed method suitable for future improvements.

REFERENCES

[1] M. Yu, G. Li, D. Deng, and J. Feng, "String similarity search and join: a survey," *Frontiers of Computer Science*, vol. 10, pp. 399–417, 2016.

[2] V. Agoston, L. Kajan, O. Carugo, Z. Hegedus, K. Vlahovicek, and S. Pongor, "Concepts of similarity in bioinformatics," *nato science series sub series i life and behavioural sciences*, vol. 368, p. 11, 2005.

[3] P. Achananuparp, X. Hu, and X. Shen, "The evaluation of sentence similarity measures," in *Data Warehousing and Knowledge Discovery: 10th International Conference, DaWaK 2008 Turin, Italy, September 2-5, 2008 Proceedings 10*. Springer, 2008, pp. 305–316.

[4] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Us- ing of jaccard coefficient for keywords similarity," in *Proceedings of the international multiconference of engineers and computer scientists*, vol. 1, no. 6, 2013, pp. 380–384.

[5] O. Shahmirzadi, A. Lugowski, and K. Younge, "Text similarity in vector space models: a comparative study," in *2019 18th IEEE inter- national conference on machine learning and applications (ICMLA)*. IEEE, 2019, pp. 659–666.

[6] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge." *Psychological review*, vol. 104, no. 2, p. 211, 1997.

[7] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[8] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[11] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior research methods, in- struments, & computers*, vol. 28, no. 2, pp. 203–208, 1996.

[12] C. Paul, A. Rettinger, A. Mogadala, C. A. Knoblock, and P. Szekely, "Efficient graph-based document similarity," in *The Semantic Web. Latest Advances and New Domains: 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29–June 2, 2016, Proceedings 13*. Springer, 2016, pp. 334–349.

[13] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[14] W. H. Gomaa, A. A. Fahmy *et al.*, "A survey of text similarity approaches," *international journal of Computer Applications*, vol. 68, no. 13, pp. 13–18, 2013.

[15] J. Wang and Y. Dong, "Measurement of text similarity: a survey," *Information*, vol. 11, no. 9, p. 421, 2020.

[16] M. Yadav and M. A. Alam, "Dynamic time warping (dtw) algorithm in speech: a review," *International Journal of Research in Electronics and Computer Engineering*, vol. 6, no. 1, pp. 524–528, 2018.

[17] W. Hou, Q. Pan, Q. Peng, and M. He, "A new method to analyze protein sequence similarity using dynamic time warping," *Genomics*, vol. 109, no. 2, pp. 123–130, 2017.

[18] V. Froese, B. Jain, M. Rymar, and M. Weller, "Fast exact dynamic time warping on run-length encoded time series," *Algorithmica*, vol. 85, no. 2, pp. 492–508, 2023.

[19] X. Liu, Y. Zhou, and R. Zheng, "Sentence similarity based on dynamic time warping," in *International Conference on Semantic Computing (ICSC 2007)*. IEEE, 2007, pp. 250–256.

[20] M. Matuschek, T. Schlu¨ter, and S. Conrad, "Measuring text similarity with dynamic time warping," in *Proceedings of the 2008 international symposium on Database engineering & applications*, 2008, pp. 263–267.

[21] M. Muller, *Information Retrieval for Music and Motion*, 2007th ed. Berlin, Germany: Springer, sep 2007.

[22] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, 2000, pp. 39–48.

[23] M. E. B. Menai, "Detection of plagiarism in arabic documents," *International Journal of Information Technology and Computer Science*, vol. 10, no. 10, pp. 80–89, 2012.

[24] S. R. Alenazi, K. Ahmad, and A. Olowolayemo, "A review of similarity measurement for record duplication detection," in *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*. IEEE, 2017, pp. 1–6.

[25] Ticki, "Eudex: A blazingly fast phonetic reduction/hashing algorithm," https://github.com/ticki/eudex, 2016.

[26] Y. Doval, M. Vilares, and J. Vilares, "On the performance of phonetic algorithms in microtext normalization," *Expert Systems with Applications*, vol. 113, pp. 213–222, 2018.

[27] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[28] K. Kukich, "Techniques for automatically correcting words in text," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 377–439, 1992. [29] W. E. Winkler, "String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage." 1990.

[30] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989.

[31] N. Gali, R. Mariescu-Istodor, and P. Fra¨nti, "Similarity measures for title matching," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1548–1553.

[32] M. Flor, M. Fried, and A. Rozovskaya, "A benchmark corpus of english misspellings and a minimally-supervised model for spelling correction," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2019, pp. 76–86.