2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Modelling Text Similarity: A Survey

Wenchuan Mu and Kwan Hui Lim
Singapore University of Technology and Design
{wenchuan_mu, kwanhui_lim}@sutd.edu.sg

***Abstract*—Online social networking services such as Twitter and Instagram have become pervasive platforms for engaging in discussions on a wide array of topics. These platforms cater to both mainstream subjects, like music and movies, as well as more specialized areas, such as politics. With the growing volume of textual data generated on these platforms, the ability to define and identify similar texts becomes crucial for effective investigation and clustering. In this paper, we explore the challenges and significance of text similarity regression models in the context of online social networking services. We delve into the methods and techniques employed to define and find similarities among texts, enabling the extraction of meaningful patterns and insights. Specifically, we categorize text similarity regression models into four distinct types: set-theoretic, sequence-theoretic, real-vector, and end-to-end methods. This categorization is based on the mathematical formalisation of similarity used by each model. Ultimately, our survey aims to provide a comprehensive overview of the interlinkages between independently proposed methods for text similarity. By understanding the strengths and weaknesses of these methods, researchers can make informed decisions when designing novel approaches and algorithms. We hope this survey serves as a valuable resource for advancing the state-of-the-art in addressing the complex problem of text similarity.**

***Index Terms*—Modelling and simulation, Deep learning and embeddings, Algorithms and techniques**

I. Introduction

The widespread adoption of online social networking services, like Twitter and Instagram, has transformed the landscape of communication and information exchange. These platforms have established themselves as influential mediums, fostering discussions that encompass a diverse range of topics. They serve as dynamic spaces accommodating both popular and mainstream subjects. With the continual growth of these platforms, the sheer volume of textual data generated necessitates the development of robust mechanisms for defining and identifying similarities among texts. Such mechanisms play a pivotal role in enabling the extraction of meaningful patterns and insights.

In the past years, the computation of similarity has gained increasing importance as a fundamental tool in various social computing tasks and real-world applications. These applications include information retrieval, clustering, topic detection, social recommendation and even financial and legal domain.

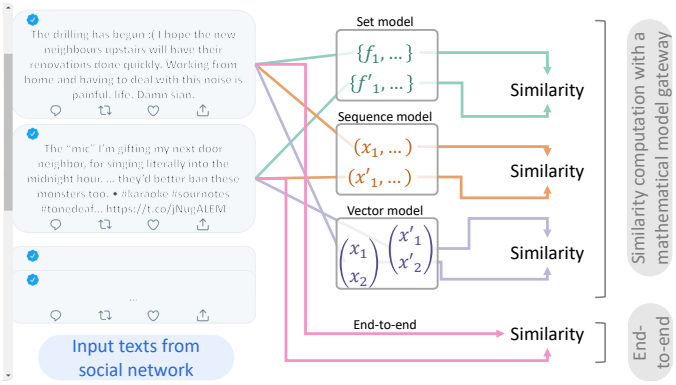

Fig. 1: Similarity can be computed using specific mathematical models such as sets, sequences, or vectors, or it can be computed in an end-to-end manner.

For instance, search engines rely on modelling document correlations to find relevant documents for a given query, while clustering methods need to determine if the texts given are similar in semantics. Social Recommendation service systems aim to comprehend human language queries or commands and provide the most appropriate recommendation. In legal and financial domains, assessing the similarity to existing contracts can help estimate the risk level associated with a new contract [1].

The field of computing similarity between texts has made significant achievements. In this regard, we present a taxonomy that centres on the final step of each similarity calculation. Analyzing the formal aspects of this last step is crucial for enhancing the interpretability of similarity calculations. It is worth noting that similarity computation extends beyond its application in social computing and can be regarded as a broader concept that is applicable to any domain dealing with text.

Our proposed taxonomy offers several advantages over existing methods. Existing taxonomies often tend to focus solely on technical intricacies like whether they use a transformer architecture or not, or specific applications [2]. Alternatively, some methods combine multiple orthogonal characteristics of similarity [3]. In contrast, our proposed scheme ensures that each similarity approach falls into one distinct and mutually exclusive subset. Moreover, this taxonomy can be extended to accommodate future advancements in similarity computing techniques. Importantly, the proposed scheme is not based

ASONAM '23, November 6–9, 2023, Kusadasi, Turkey

ACM ISBN 979-8-4007-0409-3/23/11…$15.00
https://doi.org/10.1145/3625007.3627305

on technical details but rather on mathematical formalisation, providing a robust framework for future developments. In taxonomizing and analyzing similarity approaches, we make the following contributions :

- **General Problem Definition**: We present a clear and encompassing definition of the similarity problem, along with a thorough examination of existing similarity schemes.
- **Taxonomy for Text Similarity**: We provide the first taxonomy for text similarity approaches based on formal data structure.
- **Comprehensive Analysis**: We conduct a comprehensive analysis of text similarity approaches, outlining their advantages and disadvantages, and providing insights into their application within specific categories.

II. General Form of Similarity

Similarity is widely discussed in social computing literature [4], [5]. The general formalisation for similarity calculation is as follows. Each text unit, denoted as variable $u \in \mathcal{U}$, can be compared using a similarity scheme represented by a multivariate function $s : \mathcal{U},\mathcal{U}\ldots\mathcal{U} \rightarrow \mathbb{R}^k$. The number of variables in the function corresponds to the number of units being compared, and similarity is measured across k dimensions. These dimensions include lexicon, syntax, style, semantics, and pragmatics.

Typically, a similarity scheme concentrates on a specific dimension of similarity when comparing two pieces of text, and the general form can be simplified as $s : \mathcal{U},\mathcal{U} \rightarrow \mathbb{R}$. Previous surveys often classify these schemes based on their intended dimension of similarity. However, the definitions for these dimensions remain ambiguous [3]. In light of this, our primary objective is not to elucidate the intent of each similarity scheme, but rather to present the mathematical formulation underlying them.

III. Taxonomy on Text Similarity Computation

This section provides a comprehensive taxonomy of similarity computing schemes, offering precise definitions for accurate interpretation and comparison. There are two main approaches: 1) modelling text units with mathematical models and computing similarity based on those models, or 2) computing similarity without relying on particular data structures. Currently, sets, sequences, and real vectors are the primary representations used for similarity computation in the literature. The illustration of the four is shown in Fig. 1.

A. Model Texts as Sets

The set model is a conventional approach for representing texts. Initially, we will explore how set operations can effectively express similarity. Subsequently, we will delve into text modelling using sets.

The journey from sets to similarity typically involves employing mathematics. Set operations such as union ($A \cup B$), intersection ($A \cap B$), complement ($\complement A$), and cardinality ($|A|$) are utilized to estimate the similarity between two sets (A and B). The Jaccard index, developed by Paul Jaccard in 1901, measures similarity by comparing the shared elements to the total distinct elements in sets. It is widely used for assessing set-based similarity. Additionally, the Jaccard index can be applied to bags instead of sets [7] within the context of set algebra [8]. This is particularly useful in scenarios where elements can occur multiple times (in bags) rather than just once (in sets). Various other indices, such as the Jaccard distance, Sørensen-Dice index, or Otsuka-Ochiai coefficient, are summarized in Table I.

TABLE I: From sets to similarity/dissimilarity. Set-based similarity measures quantify the intersection of sets, with larger intersections indicating higher similarity and unique elements reducing similarity.

Index	Formula	Relation to other indices
Jaccard index	$J(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$	$J = \frac{S}{2-S}$
Jaccard distance	$1 - J(A,B)$	$1 - J(A,B)$
Sørensen-Dice index	$S(A,B) = \frac{2\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert}$	$S = \frac{2J}{1+J}$
Otsuka-Ochiai coefficient [9]	$K(A,B) = \frac{\lvert A \cap B \rvert}{\sqrt{\lvert A \rvert \lvert B \rvert}}$	Equivalent to cosine similarity

There are multiple approaches to modelling text as sets. Typically, a text is converted into a set comprising different features. One of the most straightforward features is an n-gram, a contiguous sequence of n items from a given text, such as phonemes, syllables, letters, words, etc. The presence of shared features between texts enhances the similarity, while the absence of common features decreases the similarity. Formally, a set or bag is created like $\{\, x \mid x \text{ is an } n\text{-gram in } u \,\}$. When the Jaccard index is computed between such bags, we are actually computing ROUGE-N, the most used variant of ROUGE [10].

In addition to getting n-grams directly, features can also be functions of word attributes, such as gloss or neighbour concepts [2], [11]. For example, each word can be expanded using its gloss [12], [14] or synonyms [13] where more n-grams can be found. In this way, the set models are able to capture semantic similarity, although these features typically rely on ontologies that may lack extensive semantic features beyond taxonomic relationships [11]. There is still a desire for a comprehensive feature set encompassing diverse textual features [3].

B. Model Texts as Sequences

A sequence, also called a string, is an enumerated collection of symbolised objects where order matters. It contains members (also called elements, or terms) and repetitions are allowed. Usually, a sequence is defined as $s : \mathbb{N}_{+,\leq L} \rightarrow \mathcal{X}$, where $\mathcal{X}$ is the set of all possible members of the sequence, and L is the length of the sequence. This function maps from positive natural numbers, the positions of elements in the sequence, to the elements at each position.

TABLE II: From sequences to similarity/dissimilarity. Sequence-based similarity requires specialised algorithms from information/computer science. The time complexity of computing each value between two sequences is illustrated.

Index	Time complexity	Relation to other indices
Longest common subsequence	$O(\|a\| \times \|b\|)$	A substring is always a subsequence
Longest common substring	$\Theta(\|a\| + \|b\|)$	
Edit distance	$O(\|a\| \times \|b\|)$	-

There are various quantities to measure sequence similarity. One is to find the longest common subsequence (LCS) [16], where subsequences can be non-consecutive. Another method is to find the longest common substring, which requires consecutive token positions [17]. Edit distance is widely used and involves atomic operations like substitution, insertion, and deletion [18]. Table II lists the current best time complexities for computing these quantities using specific algorithms.

Current sequence representations of texts are simple because they are sequences by nature. Converting a text into a sequence often resorts to tokenization. Common tokens are characters, words (separated by spaces), or sub-words [19]. In some cases, rules are done to tokens, for example, "u.s." may be treated as the same token as "US".

The different similarity schemes for sequence models vary in how texts are modelled as sequences and which calculation (LCS, longest common substring, or edit distance) they employ. ROUGE-L disregards case and tokenizes a text by spaces to obtain a sequence of tokens, subsequently applying LCS [10]. Greedy string tiling [17] uses the same tokenization as ROUGE-L and applies the longest common substring approach. Word Error Rate (WER) also adopts the same tokenization and utilizes edit distance [21]. TER (Translation Edit Rate) is an evolution of WER that allows the consideration of a phrase as a single token to minimize the distance [23]. ITER represents an improved version of TER [25] where tokens are stemmed words rather than the original uncased words. CharacTER also utilizes edit distance but tokenizes the text into cased characters [26].

C. Model Texts as Vectors

The vector model is sometimes called feature, embedding etc. Similarity calculation between vectors can take advantage of numerous operations in real vector spaces. In the following, we first discuss how similarity between vectors can be calculated, and then discuss how to represent a unit of text using a vector.

Converting texts into a vector is not a secret skill in NLP, therefore computing the similarity between vectors is a more general problem with widespread applications in other fields such as computer vision. Some researchers [2], [3], [28] examined various formulas, including standard mathematical operations and others specifically designed similarity computing. The inner product is the most straightforward operation on two vectors. Various similarity calculations can be derived based on the inner product. For example, dividing the inner product by modulo of the vectors results in cosine, and changing the denominators can result in Jaccard, Sørensen-Dice, and Overlap formulation for real vectors, as shown in Table III.

TABLE III: From vectors to similarity/dissimilarity. Vector-based similarity measures quantify the inner product between vectors, with larger inner product indicating higher similarity.

Index	Formula	Relation to other indices
Inner product	$\langle \mathbf{a}, \mathbf{b} \rangle$	$\frac{\|\mathbf{a}-\mathbf{b}\|^2}{2} = 1 - \langle \mathbf{a}, \mathbf{b} \rangle$,[1]
Cosine Sim.	$\frac{\langle \mathbf{a},\mathbf{b} \rangle}{\|\mathbf{a}\|\|\mathbf{b}\|}$	Equivalent to Pearson correlation [29]
Jaccard index	$J(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a},\mathbf{b} \rangle}{\|\mathbf{a}\|^2+\|\mathbf{b}\|^2-\langle \mathbf{a},\mathbf{b} \rangle}$	$S = \frac{2J}{1+J}$
Sørensen-Dice index	$S(\mathbf{a}, \mathbf{b}) = \frac{2\langle \mathbf{a},\mathbf{b} \rangle}{\|\mathbf{a}\|^2+\|\mathbf{b}\|^2}$	$S = \frac{2J}{1+J}$
Overlap index	$\frac{\langle \mathbf{a},\mathbf{b} \rangle}{\min(\|\mathbf{a}\|^2,\|\mathbf{b}\|^2)}$	$S = \frac{2J}{1+J}$
L^2-norm	$\|\mathbf{a} - \mathbf{b}\|$	$\frac{\|\mathbf{a}-\mathbf{b}\|^2}{2} = 1 - \langle \mathbf{a}, \mathbf{b} \rangle$,[1]
Cosine distance	$1 - \frac{\langle \mathbf{a},\mathbf{b} \rangle}{\|\mathbf{a}\|\|\mathbf{b}\|}$	-

[1] for unit vectors

Besides inner products, cross-covariance between vectors can serve as a statistical measure of similarity. Typically, three types of correlations are computed: Pearson correlation (r), Kendall rank correlation (τ), and Spearman correlation (ρ). Pearson correlation is equivalent to the cosine of the angle between two vectors [29]. Kendall correlation and Spearman correlation, on the other hand, are nonparametric tests that assess the strength of dependence and the degree of association between two vectors, respectively [30]. The similarity between vectors can also be assessed using distance measures. One commonly used distance function is the Euclidean distance or L^2-norm, which is differentiable and convex but not scale invariant. However, differences in measurement units can introduce biases in distances, and vector normalization is often employed as a solution. Despite its widespread usage, the Euclidean distance is not well-suited for measuring vectors in high-dimensional spaces due to the concentration of norm or the curse of dimensionality [31], [32]. A popular distance variant is the L^1-norm which is preferred for outlier-rich data and provides sparser estimates. Other p-norms offer robustness to outliers. Different weights can be applied to dimensions, such as Canberra distance [33]. L^1-norm is equivalent to Hamming distance for bit vectors [34].

Next, we discuss how texts are modelled as vectors. Existing vector models are able to build a vector from either a word, phrase, sentence, paragraph or even larger units. Word vector models aim to capture information about each word, including its meaning, part of speech, and other attributes, using a single vector representation. The most widely used models

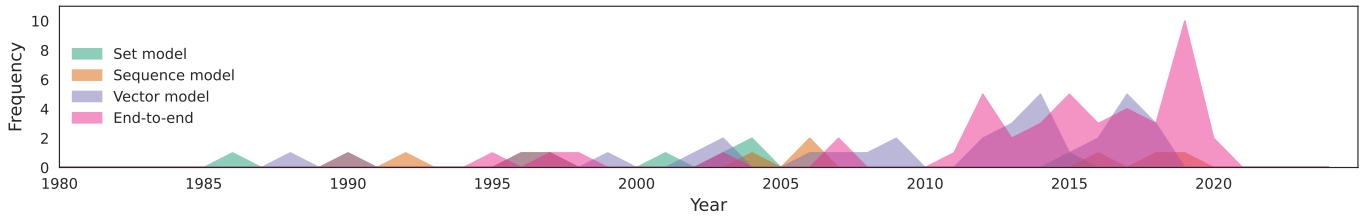

Fig. 2: Text similarity computing is an evolving field with changing popularity of different modeling approaches. Before 2010, set and vector modeling were popular, but in the past decade, vector modeling has gained widespread adoption. End-to-end computing schemes have also become increasingly popular in recent years.

for word representation are Word2Vec, GloVe, and FastText. Word2Vec takes the distributional semantics that words with similar meanings tend to appear in similar contexts. Thus, A shallow neural network is trained to capture information from the context of other words surrounding a given word to predict a vector, which is believed to preserve latent linguistic relationships [36] and semantic similarity [37] between words. While also taking distributional semantics, GloVe globally builds a global word co-occurrence matrix on a corpus [38]. FastText includes characters in the vocabulary set and employs a skip-gram approach [19] for training the character representation. This enables it to capture the morphological structure of words and effectively handle out-of-vocabulary words using their characters or subunits.

The vectors at sentence or document level can be obtained from word vectors [28], [39], including taking an average of the word vectors or the dimension-wise max/min (Vector Extrema, [40]). Simple averaging assigns equal weights to both important and unimportant words. However, considering extreme values along each dimension can help disregard common words and prioritize informative ones. To preserve more information, one can assign TF-IDF weights during the averaging process. TF-IDF (term frequency-inverse document frequency) is a widely used vectorisation method [46]–[48] that does not originate from distributional semantics. Instead, word frequencies in the text can be informative. In the space it defines, each term in the vocabulary is an orthonormal basis. The modulo length of an orthonormal basis is determined by the product of two values: how many times a word appears in a document, and the logarithm of inverse document frequency of the word across a set of documents. Some variations of TF-IDF are taking n-grams as terms [49], or using time to increase the IDF [50], where a term has a low IDF and high discrimination when it is first introduced, and the IDF decays as the term's usage increases. TF-IDF has some specific version names, like CIDEr [51] when used in image captioning comparison. Consider the frequency of function word only can indicate authorship attribution [52].

D. End-to-end Similarity Computing

Some similarity measures are not limited to specific representations like sets, bags, or vectors. Instead, they are only governed by a general formalisation of similarity computation.

In the following discussion, we describe representative end-to-end schemes in a rough chronological order to illustrate their evolution. Early works focused on analyzing specific knowledge sources of concepts [53]. Subsequently, methods began leveraging distributional semantics assumptions, more powerful computing resources, and pretrained contextualized embeddings [56]. More recently, end-to-end neural models have been utilized for modelling similarity as well [121].

Ontology measures are specifically useful for term (word) similarity. They effectively capture human-annotated information, encompassing term meanings and semantic relationships [11]. These measures determine similarity by leveraging knowledge sources and information content indices (IC), where higher IC values indicate greater specificity [53]. Existing similarity formulas in ontology and IC follow a general form: $s(A,B) = (1 + IC_1(A,B) \cdot k^{IC_2}(A,B))^{-1}$, where k is a hyperparameter tuned for different IC calculations [54].

When it comes to larger text units like sentences, composite methods are developed. These methods extract customized features, including bags of words, sentence length, and word embeddings, from the text. A clear composition rule is then used to compute similarity based on these features. Although initially developed as context-free metrics for sentence similarity evaluation [6], many composite methods are occasionally overlooked in the textual similarity community [2]. These methods typically rely on unambiguous features from texts and may avoid using word embeddings to avoid potential uninterpretable mistakes, as they are not definitive labels. However, when contextualised embedding starts to show great potential in various NLP tasks [55], many similarity schemes use contextualised embeddings as their main features. For example, BERTScore measures soft overlap between two token-aligned texts [56]. To compute BERTScore, contextualised word embeddings are generated from each text using BERT model [57], and the composition rule is to greedily maximize the cosine similarity between embeddings from two texts [58].

When a substantial amount of annotated similarity exists between two texts, one effective approach is to employ neural networks as regressors. CNN [60], LSTM [61]), transformers [64], [66], and other techniques are commonly utilized. Furthermore, the performance of these models can be enhanced by utilizing a larger corpus, which underscores the significance of constructing an ideal corpus. However, neural

models tend to be black-box models, making them challenging to interpret.

IV. Analysis

This section will evaluate the pros and cons of the four categories, followed by a discussion of their appropriate application scenarios.

A. Advantages and Disadvantages

Employing mathematical models as the foundation for computing text similarity is a double-edged sword, possessing both immense potential and inherent limitations. Mathematical models, including sets, sequences, and vectors, provide standardized approaches, efficient algorithms, and scalability advantages in the context of text similarity computation. They enable rapid computations and effective handling of large-scale similarity tasks. However, relying solely on a single mathematical model can restrict the ability to capture diverse types of text similarity, potentially resulting in suboptimal performance and a loss of specialized capabilities. For instance, vector models may struggle with nuances such as shifts in meaning or irony. The choice of data structure depends on specific task requirements, available resources, and the trade-offs between efficiency, robustness, and performance.

Set-based models efficiently apply set operations, but ignore word order and rely solely on word presence. Sequence-based models preserve word order and context, but small changes can significantly impact similarity calculations, requiring complex algorithms and being computationally expensive. Vector models, like word embeddings, capture semantic relationships and enable similarity measures, but fixed-length representations may hinder long-range dependencies and encounter dimensionality challenges.

End-to-end approaches offer the advantage of simplifying modelling by eliminating the need for explicit feature extraction and being flexible with data structures. They excel in specialized and contextualized tasks, particularly in categorizing fine-grained topics in short texts. However, they can be computationally resource-intensive, limiting their applicability in scenarios with constraints. While they may achieve higher accuracy, there is a risk of overfitting and limited generalizability in diverse scenarios. Moreover, end-to-end schemes often have high time complexity and computational expense, especially in tasks involving one-to-many mapping like topic identification. The choice between a mathematical model and an end-to-end approach for text similarity depends on task requirements, available computational resources, data availability, and the need for specialized flexibility. Please refer to Table IV for a comparison and Fig. 2 for the popularity shift over time.

B. Applications

In social computing, various tasks demand analysis of textual data to extract meaningful insights and drive decision-making. Text similarity computation plays a pivotal role in tasks such as online community management, social network analysis, recommender systems, crowdsourcing, and more. For each task in social computing, selecting the most appropriate text similarity computation is crucial. Factors such as the nature of the task and the characteristics of the textual data influence the choice between a set model, sequence model, vector model, or end-to-end computation. By examining each task's requirements, we discuss the best-fit approach for the following tasks in the following sections.

a) Community Detection: In social network analysis, community detection is a task that involves identifying cohesive and densely connected subgroups within a network. Text similarity plays a crucial role in this process by comparing textual attributes such as user profiles, descriptions, or communication patterns. Vector models tend to be effective. By representing these attributes as vectors using techniques like word embeddings or document embeddings, similarity can be computed using measures like cosine similarity. This approach allows for capturing semantic relationships and contextual meaning in the text. By applying text similarity analysis, researchers can uncover clusters of users with similar interests, identify influential users, and understand the community structure within a social network, as seen in Twitter network analyses. Overall, text similarity analysis enhances the study of social network structure, relationships, and dynamics, providing meaningful insights for decision-making.

b) Online Community Management: Moderating discussions, resolving conflicts, and promoting user engagement are essential for fostering online community growth and enhancing positive user experiences. Text similarity can aid in detecting duplicate or similar posts, comments, or discussions, enabling community managers to efficiently identify repetitive content or spam and merge similar discussions. Set models, treating each post or comment as a set of words or tokens, can effectively identify duplicates. Set similarity metrics like Jaccard similarity or cosine similarity measure word overlap or similarity between sets. For instance, in an online gaming community, text similarity techniques can help identify multiple instances of the same question posted across different threads. Community managers can swiftly recognize these duplicates and consolidate them into a single thread, reducing redundancy and enhancing community organization.

c) Recommender Systems: Recommender systems aim to provide personalized recommendations to users in various domains such as e-commerce, social media, and content streaming services. Text similarity analysis can compare textual representations of items or user preferences. By measuring the similarity between item descriptions or user profiles, recommender systems can identify similar items or users, leading to accurate recommendations. Vector models, again, are effective in this context. For instance, in music streaming platforms, text similarity can be employed to analyze song metadata and user listening histories. By calculating the similarity between songs represented as vectors based on genre, lyrics, or other textual attributes, the recommender system can suggest similar songs to users who have shown interest in a particular music style, according to vector distances like Euclidean distance.

TABLE IV: Advantages and Disadvantages of Text Similarity Modeling Approaches. Among the surveyed schemes, 55% convert linguistic units to some form of representation, including sets (8%), sequences (15%), or real vectors (32%). The remaining 45% take an end-to-end perspective to return real numbers. The unit of texts can be words (90%), sub-words, or characters ($< 10\%$). 70% of the schemes came out after 2010.

Approach	Advantages	Disadvantages	Schemes in Literature
Set Modelling	Simplicity, Robustness to minor changes, Efficiency	Loss of sequence information, Loss of frequency information	n-**gram** [10], [67], [68], **Lesk** [12], [13], **Wiki-feature** [14]
Sequence Modelling	Preserves sequence information, Contextual understanding	Increased complexity, Sensitivity to variations	**ROUGE-L** [10], **Greedy string tiling** [17], **Jaro-Winkler distance** [20], **WER** [21], **PER** [22], **TER** [23], **ITER** [25], **CDER** [24], **characTER** [26], **EED** [27]
Vector Modelling	Efficient computation, Rich semantic representation, Enables various similarity measures	Dimensionality, Fixed-length representation	**Function word frequency** [52], **TF-IDF** [46]–[48], **CIDEr** [51], **Word order** [69], **Word2vec** [35], **GloVe** [38], **FastText** [19], **LSI/LSA** [39], [70], **HAL** [71], **LDA** [72], **ESA** [73], **Dependency-based models** [74], [75], **Tree kernels** [76]–[82], **Word-Alignment models** [83]–[85], **Document embedding** [40]–[45], **InferSent** [86], **Quick-Thought** [87], **USE** [88]
End-to-end Modelling	Simplicity, Flexible representation learning, Task-specific optimization	Lack of interpretability, Data requirements, Limited control over similarity computation	**NGD** [89], **IC** [53], [54], [90]–[94], **Lexico-syntactic patterns** [95], **PORT** [96], **LEPOR** [97], [98], **GTM** [99], **MEANT** [100]–[103], **Greedy Emb. Matching** [104], **WMD** [105]–[107], **BERTScore** [56], **SIMILE** [108], **MoverScore** [109], **Q-Metrics** [110], **BEER** [111], [112], **ESIM** [58], [113], **Neural Network Methods in General** [59]–[66], [114]–[120], **RUSE** [121]

d) Crowdsourcing: Crowdsourcing involves outsourcing tasks or gathering information from a large crowd or online community. Text similarity can identify redundant or similar contributions from participants. By measuring text similarity between user-generated content, such as responses or solutions, organizers can group similar responses together, assess the consensus, and avoid duplication. Both the set models and ontology-based end-to-end approaches can be useful. For example, when participants are asked to categorize images, these models can cluster similar image descriptions provided by users. This helps in identifying patterns or themes within the collected data, facilitating the overall categorization process.

e) Online Collaboration: Online collaboration involves multiple individuals working together remotely on shared projects, documents, or tasks. It includes activities like co-authoring, content editing, and real-time communication. Text similarity can be helpful in detecting overlapping or duplicate content within collaborative documents. By measuring the text similarity between sections or paragraphs, collaborators can identify redundant information, merge similar contributions, and maintain document consistency. In a team working on a shared document, text similarity techniques can be applied to compare the content of different sections. Sequence models, such as sequence alignment or edit distance, are often suitable for this purpose. By representing sections or paragraphs of collaborative documents as sequences, the similarity between sections can be effectively measured and analyzed.

f) Influence Propagation: Influence propagation involves studying the spread of information, opinions, or behaviours through social networks. It aims to understand how influence or trends propagate and impact individuals within a network. Text similarity analysis provides insights into the spread of information and helps identify influential users by assessing the similarity between messages or posts. Vector models are typically well-suited. For example, when studying the diffusion of information on social media, each tweet or post can be first represented in a vector, and similarity between tweets pertaining to a specific event or topic can be obtained.

g) Topic Detection on Social Media: This task involves analyzing data from social media platforms to collect some particular topic-related posts. It encompasses tasks such as sentiment analysis, topic modelling, and trend detection. Text similarity can group similar posts or comments together. By measuring text similarity between social media content, analysts can identify conversations, discussions, or trends related to specific topics, enabling more targeted analysis and understanding of user sentiments. The vector models are commonly applied, where social media content are represented as vectors, e.g., sentence embeddings. However, when there is sufficient annotation available, the end-to-end approach can be more efficient in detecting the particular topic-related posts. It eliminates the need for explicit feature extraction and considers the overall characteristics of the posts, resulting in improved efficiency and effectiveness in analyzing topic-related content. For example, in a social media analytics project focused on a specific product launch, text similarity techniques can be used to group user-generated content related to the product.

V. Conclusion

In our systematic review of approximately 70 similarity computation schemes, we categorized them into four types: sets, sequences, real vectors, and end-to-end methods. We observed that vector models and end-to-end methods have been particularly popular in the past decade, with vector models demonstrating better efficiency and end-to-end methods showcasing higher performance in specialized tasks. However, these methods are more challenging to interpret compared to simpler

set and sequence models. Understanding the computation of text embedding and similarity in complex vector models and end-to-end methods remains a topic for future research.

Acknowledgment. This research is funded in part by the Singapore University of Technology and Design under grant RS-MEFAI-00005-R0201.

REFERENCES

[1] V. Rawte, A. Gupta, and M. J. Zaki, "A comparative analysis of temporal long text similarity: Application to financial documents," *MIDAS*, 2021.

[2] D. Chandrasekaran and V. Mago, "Evolution of semantic similarity—a survey," *ACM Comput. Surv.*, 2021.

[3] D. Bär, T. Zesch, and I. Gurevych, "Composing measures for computing text similarity," Tech. Rep., 2015.

[4] Y. Zhang, X. Wang, Y. Sakai, and T. Yamasaki, "Measuring similarity between brands using followers' post in social media," in *MMAsia*, 2019

[5] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *KDD*,2004

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," *ACL*, 2002.

[7] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Finding Similar Items*, 2022.

[8] L. E. Bertossi, G. Gottlob, and R. Pichler, "Datalog: Bag semantics via set semantics,", 2018.

[9] Y. Otsuka, "The faunal character of the japanese pleistocene marine mollusca, as evidence of climate having become colder during the pleistocene in japan," *Biogeogr. Soc. Japan*, 1936.

[10] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2004.

[11] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," *ESWA*, 2012.

[12] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone," in *SIGDOC*, 1986.

[13] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness," in *IJCAI*, 2003.

[14] Y. Jiang, X. Zhang, Y. Tang, and R. Nie, "Feature-based approaches to semantic similarity assessment of concepts using wikipedia," *IP&M*, 2015.

[15] P.-F. Marteau, "Sequence Covering Similarity for Symbolic Sequence Comparison," 2018.

[16] L. Allison and T. I. Dix, "A bit-string longest-common-subsequence algorithm," *IPRL*, 1986.

[17] M. J. Wise, "Yap3: Improved detection of similarities in computer program and other texts," *SIGCSE Bull.*, 1996.

[18] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Sov. Phys. Dokl.*, 1966.

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *TACL*, 2017.

[20] W. E. Winkler, "String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage," in *Proceedings of the Section on Survey Research*, 1990.

[21] K.-Y. Su, M.-W. Wu, and J.-S. Chang, "A new quantitative quality measure for machine translation systems," in *COLING*, 1992.

[22] C. Tillmann, S. Vogel, H. Ney, A. Zubiaga, and H. Sawaf, "Accelerated DP based search for statistical translation," in *EUROSPEECH*, 1997.

[23] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A study of translation edit rate with targeted human annotation," in *AMTA*, 2006.

[24] G. Leusch, N. Ueffing, and H. Ney, "CDER: Efficient MT evaluation using block movements," in *EACL*, 2006.

[25] J. Panja and S. K. Naskar, "ITER: Improving translation edit rate through optimizable edit costs," in *WMT*, 2018.

[26] W. Wang, J.-T. Peter, H. Rosendahl, and H. Ney, "CharacTer: Translation edit rate on character level," in *WMT*, 2016.

[27] P. Stanchev, W. Wang, and H. Ney, "EED: Extended edit distance measure for machine translation," in *WMT*, 2019.

[28] A. B. Sai, A. K. Mohankumar, and M. M. Khapra, "A survey of evaluation metrics used for nlg systems," *ACM Comput. Surv.*, 2022.

[29] R. J. Rummel, "The vector approach," in *Understanding correlation*, 1976, ch. 5.

[30] D. J. Colwell and J. R. Gillett, "Spearman versus kendall," *Math. Gaz.*, 1982.

[31] P. Demartines, "Analyse de données par réseaux de neurones auto-organisés," Ph.D. dissertation, 1994.

[32] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *ICDT*, 1999.

[33] G. Jurman, S. Riccadonna, R. Visintainer, and C. Furlanello, "Canberra distance on ranked lists," in *NIPS workshop*, 2009.

[34] J. M. Phillips, "L7-distances," 2013.

[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[36] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, "Evaluation methods for unsupervised word embeddings," in *EMNLP*, 2015.

[37] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *NAACL*, 2013.

[38] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *EMNLP*, 2014.

[39] 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." *Psychol. Rev.*, 1997.

[40] G. Forgues, J. Pineau, J.-M. Larchevêque, and R. Tremblay, "Bootstrapping dialog systems with word embeddings," in *Nips workshop*, 2014.

[41] O. Melamud, J. Goldberger, and I. Dagan, "context2vec: Learning generic context embedding with bidirectional LSTM," in *CoNLL*, 2016.

[42] J. Tissier, C. Gravier, and A. Habrard, "Dict2vec : Learning word embeddings using lexical dictionaries," in *EMNLP*, 2017.

[43] M. Pagliardini, P. Gupta, and M. Jaggi, "Unsupervised learning of sentence embeddings using compositional n-gram features," in *NAACL*, 2018.

[44] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014.

[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[46] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM J. Res. Dev.*, 1957.

[47] K. Sparck Jones, *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, 1988.

[48] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *ICML*, 2003.

[49] O. Shahmirzadi, A. Lugowski, and K. Younge, "Text similarity in vector space models: A comparative study," in *ICMLA*, 2019.

[50] B. Kelly, D. Papanikolaou, A. Seru, and M. Taddy, "Measuring technological innovation over the long run," *AER Insights*, 2021.

[51] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *CVPR*, 2015.

[52] L. P. Dinu and M. Popescu, "Ordinal measures in authorship identification," in *Proc. SEPLN*, 2009.

[53] G. Zhu and C. A. Iglesias, "Computing semantic similarity of concepts in knowledge graphs," *TKDE*, 2017.

[54] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "Yago2: A spatially and temporally enhanced knowledge base from wikipedia," *AI*, 2013.

[55] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," in *EMNLP-IJCNLP*, 2019.

[56] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," 2019.

[57] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[58] N. Mathur, T. Baldwin, and T. Cohn, "Putting evaluation in context: Contextual embeddings improve machine translation evaluation," in *ACL*, 2019.

[59] Z. Wang, H. Mi, and A. Ittycheriah, "Sentence similarity learning by lexical decomposition and composition," in *COLING*, 2016.

[60] Y. Shao, "HCTI at SemEval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity," in *SemEval*, 2017.

[61] N. H. Tien, N. M. Le, Y. Tomohiro, and I. Tatsuya, "Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity," *Ip& M*, 2019.

[62] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015.

[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[64] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," in *Findings EMNLP*, 2020.

[65] H. Shimanaka, T. Kajiwara, and M. Komachi, "Machine translation evaluation with BERT regressor," 2019.

[66] H. Kane, M. Y. Kocyigit, A. Abdalla, P. Ajanoh, and M. Coulibali, "NUBIA: NeUral based interchangeability assessor for text generation," in *EvalNLGEval*, 2020.

[67] C. Lyon, J. Malcolm, and B. Dickerson, "Detecting short passages of similar text in large document collections," in *EMNLP*, 2001.

[68] C. Lyon, R. Barrett, and J. Malcolm, "A theoretical basis to the automated detection of copying between texts, and its practical implementation in the ferret plagiarism and collusion detector," *Plagiarism: Prevention, Practice and Policies*, 2004.

[69] V. Hatzivassiloglou, J. L. Klavans, and E. Eskin, "Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning," in *EMNLP/VLC*, 1999.

[70] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *JASIS*, 1990.

[71] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behav. Res. Methods Instrum. Comput.*, 1996.

[72] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, 2003.

[73] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *IJCAI*, 2007.

[74] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa, "A study on similarity and relatedness using distributional and WordNet-based approaches," in *NAACL*, 2009.

[75] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *ACL*, 2014.

[76] A. Moschitti, D. Pighin, and R. Basili, "Tree kernels for semantic role labeling," *CL*, 2008.

[77] A. Severyn, M. Nicosia, and A. Moschitti, "Learning semantic textual similarity with structural representations," in *ACL*, 2013.

[78] A. Moschitti, "Efficient convolution kernels for dependency and constituent syntactic trees," in *ECML*, 2006.

[79] M. Collins and N. Duffy, "New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron," in *ACL*, 2002.

[80] S. Amir, A. Tanasescu, and D. A. Zighed, "Sentence similarity based on semantic kernels for intelligent text retrieval," *JIIS*, 2017.

[81] D. Bär, C. Biemann, I. Gurevych, and T. Zesch, "UKP: Computing semantic textual similarity by combining multiple content similarity measures," in *SemEval*, 2012.

[82] F. Šarić, G. Glavaš, M. Karan, J. Šnajder, and B. Dalbelo Bašić, "TakeLab: Systems for measuring semantic text similarity," in *SemEval*, 2012.

[83] M. A. Sultan, S. Bethard, and T. Sumner, "DLS@CU: Sentence similarity from word alignment," in *SemEval*, 2014.

[84] T. Kajiwara and M. Komachi, "Building a monolingual parallel corpus for text simplification using sentence similarity based on alignment between word embeddings," in *COLING*, 2016.

[85] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *SemEval*, 2017.

[86] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *EMNLP*, 2017.

[87] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," in *ICLR*, 2018.

[88] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, and R. Kurzweil, "Universal sentence encoder for English," in *EMNLP*, 2018.

[89] R. L. Cilibrasi and P. M. Vitanyi, "The google similarity distance," *TKDE*, 2007.

[90] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *IJCAI*, 1995.

[91] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *CICLing*, 1997.

[92] D. Lin, "An information-theoretic definition of similarity," in *ICML*, 1998.

[93] D. Sánchez and M. Batet, "A semantic similarity method based on information content exploiting multiple ontologies," *ESWA*, 2013.

[94] J.-B. Gao, B.-W. Zhang, and X.-H. Chen, "A wordnet-based semantic similarity measurement combining edge-counting and information content theory," *Eng. Appl. Artif. Intell.*, 2015.

[95] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring semantic similarity between words using web search engines," in *WWW*, 2007.

[96] B. Chen, R. Kuhn, and S. Larkin, "PORT: a precision-order-recall MT evaluation metric for tuning," in *ACL*, 2012.

[97] A. L. F. Han, D. F. Wong, and L. S. Chao, "LEPOR: A robust evaluation metric for machine translation with augmented factors," in *COLING*, 2012.

[98] A. L.-F. Han, D. F. Wong, L. S. Chao, L. He, and Y. Lu, "Unsupervised quality estimation model for english to german translation and its application in extensive supervised evaluation," *Sci. World J.*, 2014.

[99] J. P. Turian, L. Shen, and I. D. Melamed, "Evaluation of machine translation and its evaluation," in *MT Summit IX*, 2003.

[100] C.-k. Lo and D. Wu, "MEANT: An inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility based on semantic roles," in *ACL*, 2011.

[101] C.-k. Lo, A. K. Tumuluru, and D. Wu, "Fully automatic semantic MT evaluation," in *WMT*, 2012.

[102] C.-k. Lo and D. Wu, "Unsupervised vs. supervised weight estimation for semantic MT evaluation metrics," in *SSST-6*, 2012.

[103] C.-k. Lo, M. Beloucif, M. Saers, and D. Wu, "XMEANT: Better semantic MT evaluation without reference translations," in *ACL*, 2014.

[104] V. Rus and M. Lintean, "A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics," in *BEA*, 2012.

[105] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *ICML*, 2015.

[106] J. Chow, L. Specia, and P. Madhyastha, "WMDO: Fluency-based word mover's distance for machine translation evaluation," in *WMT*, 2019.

[107] H. Echizen'ya, K. Araki, and E. Hovy, "Word embedding-based automatic MT evaluation metric using word position information," in *NAACL*, 2019.

[108] J. Wieting, T. Berg-Kirkpatrick, K. Gimpel, and G. Neubig, "Beyond BLEU:training neural machine translation with semantic similarity," in *ACL*, 2019.

[109] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, "MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance," in *EMNLP-IJCNLP*, 2019.

[110] P. Nema and M. M. Khapra, "Towards a better metric for evaluating question generation systems," in *EMNLP*, 2018.

[111] M. Stanojević and K. Sima'an, "BEER: BEtter evaluation as ranking," in *WMT*, 2014.

[112] ——, "Fitting sentence level translation evaluation with many dense features," in *EMNLP*, 2014.

[113] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced LSTM for natural language inference," in *ACL*, 2017.

[114] N. Sharif, L. White, M. Bennamoun, and S. A. Ali Shah, "Learning-based composite metrics for improved caption evaluation," in *ACL*, 2018.

[115] N. Sharif, L. White, M. Bennamoun, and S. A. A. Shah, "Nneval: Neural network based evaluation metric for image captioning," in *ECCV*, 2018.

[116] H. He and J. Lin, "Pairwise word interaction modeling with deep neural networks for semantic similarity measurement," in *NAACL*, 2016.

[117] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[118] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *EMNLP*, 2015.

[119] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *EMNLP*, 2016.

[120] I. Lopez-Gazpio, M. Maritxalar, M. Lapata, and E. Agirre, "Word n-gram attention models for sentence similarity and inference," *ESWA*, 2019.

[121] H. Shimanaka, T. Kajiwara, and M. Komachi, "RUSE: Regressor using sentence embeddings for automatic machine translation evaluation," in *WMT*, 2018.