information

MDPI

Review

A Survey of Text-Matching Techniques

Peng Jiang * and Xiaodong Cai

School of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China; caixiaodong@guet.edu.cn
* Correspondence: jiangpengguet@gmail.com

Abstract: Text matching, as a core technology of natural language processing, plays a key role in tasks such as question-and-answer systems and information retrieval. In recent years, the development of neural networks, attention mechanisms, and large-scale language models has significantly contributed to the advancement of text-matching technology. However, the rapid development of the field also poses challenges in fully understanding the overall impact of these technological improvements. This paper aims to provide a concise, yet in-depth, overview of the field of text matching, sorting out the main ideas, problems, and solutions for text-matching methods based on statistical methods and neural networks, as well as delving into matching methods based on large-scale language models, and discussing the related configurations, API applications, datasets, and evaluation methods. In addition, this paper outlines the applications and classifications of text matching in specific domains and discusses the current open problems that are being faced and future research directions, to provide useful references for further developments in the field.

Keywords: text matching; natural language processing; applications

Citation: Jiang, P.; Cai, X. A Survey of Text-Matching Techniques. *Information* **2024**, *15*, 332. https://doi.org/10.3390/info15060332

Academic Editor: Katsuhide Fujita

Received: 1 May 2024
Revised: 26 May 2024
Accepted: 30 May 2024
Published: 5 June 2024

1. Introduction

Text matching aims at determining whether two sentences are semantically identical or not, and it plays a key role in domains such as question and answer, information retrieval, and summarization. Early text-matching techniques relied heavily on statistical methods such as the longest common substring (LCS) [1], edit distance [2], Jaro similarity [3], dice coefficient [4], and Jaccard method [5]. These methods have the advantage of quantifiable similarity and have been widely used in several scenarios such as text comparison, spell-checking, gene sequence analysis, and recommender systems. They have a solid theoretical foundation that ensures the accuracy and reliability of similarity computation. However, there are some limitations to these methods, such as the high computational complexity of some of them, their sensitivity to input data, and limited semantic understanding. These methods include the bag-of-words model [6], distributed representation [7], matrix decomposition [8], etc., and their common advantage is that they can transform textual data into numerical representations, which is easy for computer processing. At the same time, they are flexible and scalable and can adapt to different needs and data characteristics. However, these methods still have limitations in deep semantic understanding and may not be able to accurately capture complex meanings and contextual information. In addition, the computational complexity is high when dealing with large-scale corpora, and the parameters need to be adjusted and optimized to obtain the best results. In recent years, with the rise of neural network technology, recursive neural network-based matching methods such as LSTM (Long Short-Term Memory) [9], BiLSTM(Bidirectional Long Short-Term Memory) [10], and BiGRU (Bidirectional Gated Recurrent Unit) [11] have gradually emerged. These methods share common features in terms of sequence modeling, gating mechanism, state transfer, bi-directional processing, and optimization and training. They can fully utilize the information in sequences, capture long-term dependencies, and show excellent performance in handling complex tasks.

However, these methods often ignore the interaction information between the statements to be matched during the matching process, thus limiting the further improvement of the matching effect. To solve this problem, researchers have proposed sentence semantic interaction matching methods, such as DSSTM [12], DRCN [13], DRr-Net [14], and BiMPM [15]. The core of these methods lies in capturing the semantic interaction features between sentences using deep learning models and optimizing the matching task through an end-to-end training approach. These methods usually have a high matching performance and are flexible enough to be applied to different types of matching tasks. However, these methods are usually sensitive to noise and ambiguity and are data-dependent, with limited generalization capability. With the continuous development of technology, knowledge graph and graph neural network matching methods are gradually emerging. These methods show unique advantages in dealing with graph-structured data, can effectively extract features and perform representation learning, and are suitable for dealing with complex relationships. At the same time, they also show a strong flexibility and representation capability and can be adapted to different types of matching tasks. However, these methods also face some challenges, such as high computational complexity and sensitivity to data quality and structure. In recent years, advances in deep learning methods, the availability of large amounts of computational resources, and the accumulation of large amounts of training data have driven the development of techniques such as neural networks, attention mechanisms, and Transformer [16] language models. These research results have further advanced the development of big language models. Big language models utilize neural networks containing billions of parameters, trained on massive amounts of unlabeled text data through self-supervised learning methods. Often pre-trained on large corpora from the web, these models can learn complex patterns, linguistic subtleties, and connections. This technique has also significantly advanced the development of sentence matching, with models such as BERT [17], Span-BERT [18], Roberta [19], XLNet [20], the GPT family [21–24], and PANGU-Σ [25] demonstrating superior performance in sentence-matching tasks.

The large language model described above lays the cornerstone for sentence semantic matching and focuses on exploring generic strategies for encoding text sequences. In text matching, previous approaches typically add a simple categorization target through fine-tuning and perform direct text comparisons by processing each word. However, this approach neglects the utilization of the semantic information of tags, as well as the differences in the granularity of multilevel matching in the sentence content. To address this problem, researchers have proposed a matching method based on a large language model that combines different features, which is capable of digging deeper into the semantic-matching information at different levels of granularity in tags [26] and sentences [27–29]. These studies have made important contributions to the development of the text-matching field. Therefore, a comprehensive and reliable evaluation of text-matching methods is particularly important. Although a few studies [30–32] have demonstrated the potential and advantages of some of the matching methods, there is still a relative paucity of studies that comprehensively review the recent advances, possibilities, and limitations of these methods.

In addition, researchers have extensively explored multiple aspects of text matching in several studies [30–32], but these studies often lack in-depth analyses of the core ideas, specific implementations, problems, and their solutions for matching methods. Also, there are a few exhaustive discussions on hardware configurations, training details, performance evaluation metrics, datasets, and evaluation methods, as well as applications in specific domains of matching methods. Therefore, there is an urgent need to delve into these key aspects to understand and advance the development of text-matching techniques more comprehensively. Given this, this paper aims to take a comprehensive look at the existing review studies, reveal their shortcomings, and explore, understand, and evaluate text-matching methods in depth. Our research will cover core concepts, implementation strategies, challenges encountered, and solutions for matching methods, while focusing on their required hardware configurations and training processes. In addition, we will compare

the performance of different metrics for different tasks, analyze the choice of datasets and evaluation methods, and explore the practical applications of matching methods in specific domains. Finally, this paper will also examine the current open problems and challenges faced by text-matching methods, to provide directions and suggestions for future research and promote further breakthroughs in text-matching technology.

The main contributions of this paper are as follows: (1) The core ideas, implementation methods, problems, and solutions for text-matching methods are thoroughly outlined and analyzed. (2) The hardware configuration requirements and training process of the matching method are analyzed in depth. (3) A detailed comparison of the performance of text-matching models with different metrics for different tasks is presented. (4) The datasets used for training and evaluating the matching model are comprehensively sorted out and analyzed. (5) The interface of the text-matching model in practical applications is summarized. (6) Application cases of matching models in specific domains are analyzed in detail. (7) Various current unsolved problems, challenges, and future research directions faced by matching methods are surveyed, aiming to provide useful references and insights to promote the development of the field.

The chapter structure of this study includes an overview of some leading research comparisons in Section 2. Then, the literature collection and review strategy is presented in Section 3. Then, in Section 4 we check, review, and analyze 121 top conference and journal papers published from 1912 to 2024. In Section 5, we focus on the core of the matching approach, exploring in detail its main ideas, implementation methods required hardware configurations, and training process, and comparing the performance of text-matching models with different metrics for different tasks. Through this section, readers can gain a deeper understanding of the operation mechanism and the advantages and disadvantages of the matching method. Section 6 then focuses on analyzing and summarizing the datasets used for training and evaluating the matching models and their evaluation methods, providing guidance and a reference for researchers in practice. Section 7 further explores the application of matching models in specific domains, demonstrating their practical application value and effectiveness through case studies. Section 8 reveals the various outstanding issues and challenges that the matching method is currently facing, and it stimulates the reader to think about future research directions. In Section 9, we delve into the future research directions of matching methods, providing useful insights and suggestions for researchers. Finally, in Section 10, we summarize the research in this paper.

2. Literature Review

Text matching, as an important branch in the field of artificial intelligence, has made a remarkable development in recent years. To deeply explore and evaluate its capabilities, numerous researchers have carried out extensive research work. Researchers from different disciplinary backgrounds have contributed many innovative ideas to text-matching methods, demonstrating their remarkable progress and diverse application prospects. Overall, these studies have collectively contributed to the continuous advancement in text-matching techniques. As shown in Table 1, Asha S et al. [30] provided a comprehensive review of string-matching methods based on artificial intelligence techniques, focusing on the application of key techniques such as neural networks, graph models, attention mechanisms, reinforcement learning, and generative models. The study not only demonstrates the advantages and potential of these techniques in practical applications but also deeply analyzes the challenges faced and indicates future research directions. In addition, Hu W et al. [31] provided an exhaustive review of neural network-based short text matching algorithms in recent years. They introduced models based on representation and word interaction in detail and analyzed the applicable scenarios of each algorithm in depth, providing a quick-start guide to short text matching for beginners. Meanwhile, Wang J et al. [32] systematically sorted out the current research status of text similarity measurement, deeply analyzed the advantages and disadvantages of existing methods, and constructed a more comprehensive classification and description system. The study provides an in-depth

discussion of the two dimensions of text distance and text representation, summarizes the development trend of text similarity measurement, and provides valuable references for related research and applications.

Table 1 compares in detail the research content and focus of different review papers in the field of text matching. Among them, the application of large language models in text matching, the core ideas of matching methods and their implementation, problems and solutions, and logical relationships have been explored to different degrees. However, the studies by Asha S [30] and Hu W et al. [31] are deficient in some key areas, such as the lack of a detailed overview and analysis of the matching methodology, an in-depth comparison of hardware configurations and training details, a comprehensive evaluation of the performance of the text-matching model under different tasks, and a comprehensive overview of the dataset, evaluation metrics, and interfaces of the model application. Similarly, the study by Wang J et al. [32] fails to cover many of these aspects. In contrast, our paper remedies these shortcomings by comprehensively exploring the core idea of the text-matching approach and its methodology, problems and solutions, hardware configurations, and training details, and comparing the performance of text-matching models for different tasks using different metrics. In addition, this study provides an overview of the datasets, evaluation metrics, and application programming interfaces for text-matching models, and analyzes in detail the application of the models in specific domains. Through these comprehensive studies and analyses, this thesis provides richer and deeper insights into the research and applications in the field of text matching.

As can be seen from the above analysis, most of the past studies have focused on limited areas of semantic matching, such as technology overviews, matching distance calculations, and evaluation tools. However, this study is committed to overcoming these limitations by providing an in-depth and comprehensive analysis of the above aspects, thus better revealing the strengths and weaknesses of the matching methods based on large language models. The core of this study is to deeply analyze the core idea of the matching method, compare the impact of different hardware configurations and training details on performance, evaluate the effectiveness of the text-matching model in various tasks with different metrics, outline the application program interface of the text-matching model, and explore its application in specific domains. By providing exhaustive information, this study aims to be an important reference resource for sentence-matching researchers. In addition, this study addresses open issues in sentence-matching methods, such as bottlenecks in the application of large models to sentence-matching tasks, model-training efficiency issues, interference between claim comprehension and non-matching information, the handling of key detail facts, and the challenges of dataset diversity and balance. The study highlights these key challenges and indicates directions and valuable resources for future research for sentence-matching researchers, to provide new ideas for solving these open problems.

Table 1. Comparison between leading studies.

Author or Comparative Content	Large Language Modeling for Text Matching	The Main Ideas and Methods of the Matching Approach Are Outlined and Analyzed in Detail	Detailed Overview and Analysis of the Problems and Solutions of the Matching Methodology	Detailed Overview and Analysis of the Logical Trajectories of Typical Matching Methods	Comparison of Hardware Configurations and Training Details of Matching Methods	Performance Comparison of Text-Matching Models for Different Tasks Using Different Metrics	Overview and Analysis of Text-Matching Model Datasets and Evaluation Metrics	Text Matching Model Application Programming Interface Overview	Application of Text-Matching Models to Specific Domains	Main Research Work
Asha S et al. [30], (2024)	√	×	×	×	×	×	×	×	×	This study reviews string-matching methods based on artificial intelligence techniques, focusing on the application of neural networks, graph models, attention mechanisms, reinforcement learning, and generative models. The study demonstrates their advantages and challenges in practical applications and points out future research directions.
Hu W et al. [31], (2019)	√	×	×	×	×	×	×	×	×	This paper reviews short text matching algorithms based on neural networks in recent years, introduces models based on representation and word interaction, analyzes the applicable scenarios of each algorithm, and provides a quick-start guide to short text matching for beginners.
Wang Jet al. [32], (2020)	√	×	×	×	×	×	√	×	×	This paper systematically combs through the current research status of text similarity measurement, analyzes the advantages and disadvantages of the existing methods, constructs a more comprehensive classification and description system, and carries out an in-depth discussion of the aspects of text distance and text representation. Meanwhile, the development trend of text similarity measurement is summarized, to provide a reference for related research and applications
Ours	√	√	√	√	√	√	√	√	√	A detailed overview of the core ideas, methods, problems, and solutions for each phase of the matching model, comparing hardware and software configurations, performance per task, and datasets and evaluation methods for large models.

(√ indicates the presence of this element, × indicates the absence of this element).

3. Data Collection

The literature review strategy consisted of three steps. First, the scope of the study was identified as text matching in NLP. The second step was to select widely recognized and high-quality literature search databases, such as Web of Science, Engineering Village, IEEE, SpringerLink, etc., and use Google Scholar for the literature search. Keywords include text matching, large language modeling, question and answer and retrieval, named entity recognition, named entity disambiguation, author names disambiguation, and historical documentation. In the third step, 121 articles related to text matching were collected for a full-text review and used for text-matching research and summarization. (See Figure 1).

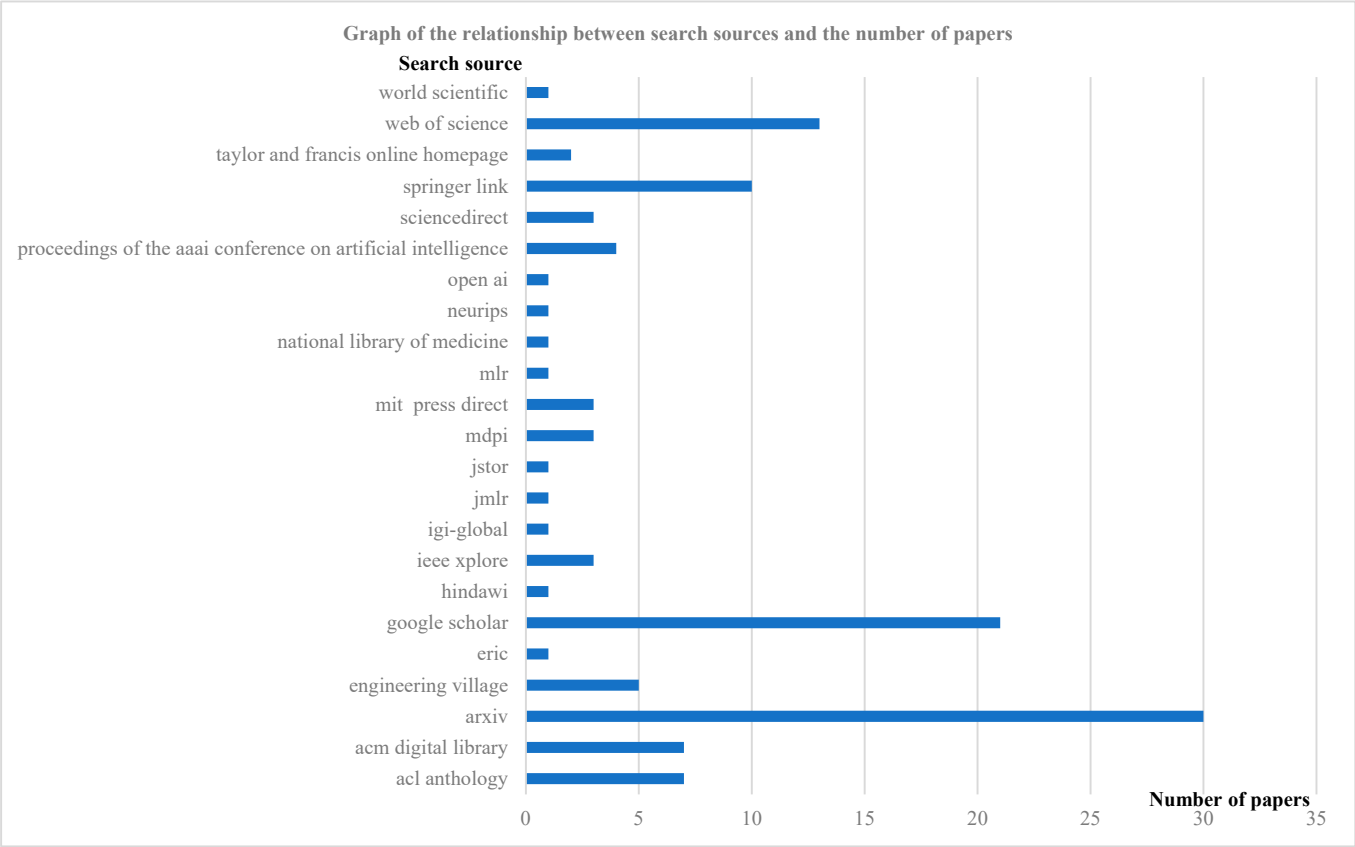

Figure 1. Graph of search sources versus number of papers.

4. Summary of Results

Figures 2–4 illustrate the distribution of articles related to text-matching survey research by year, region, and journal. As can be seen in Figure 2, this type of research has shown a significant growth trend since around 2015. Previous studies have also shown that the field of machine vision is facing a similar situation, which is closely related to the increased demand for AI technology in various industries, the improved performance of computer hardware, and the rapid development and dissemination of AI. From Figure 3, it can be observed that the US has the highest number of papers published in text-matching-related research, accounting for 49.59% of the total. This is followed by China, the UK, India, Germany, and Canada. Other countries such as the United Arab Emirates, South Korea, Israel, and South Africa have published corresponding articles. Finally, Figure 4 shows the 65 top conferences and journals that were sources of the literature, mainly focusing on the field of natural language processing, where 18 papers from 2017 to 2024 were selected in the Arxiv preprint. Eight papers from 2014 to 2022 were selected for ACL. 6 papers from 2016 to 2022 were selected for AAAI.

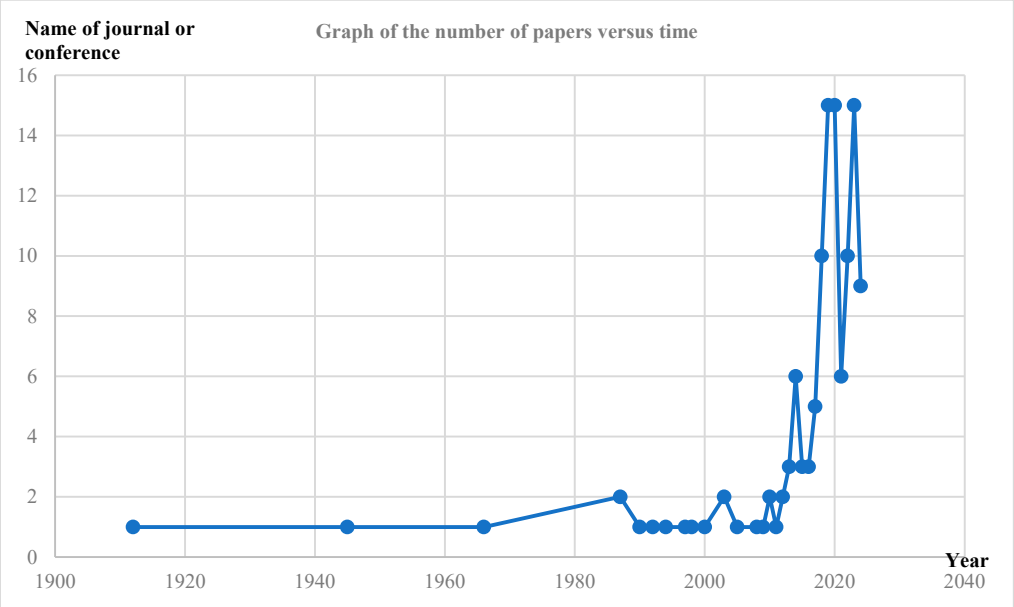

Figure 2. Graph of number of papers vs. time.

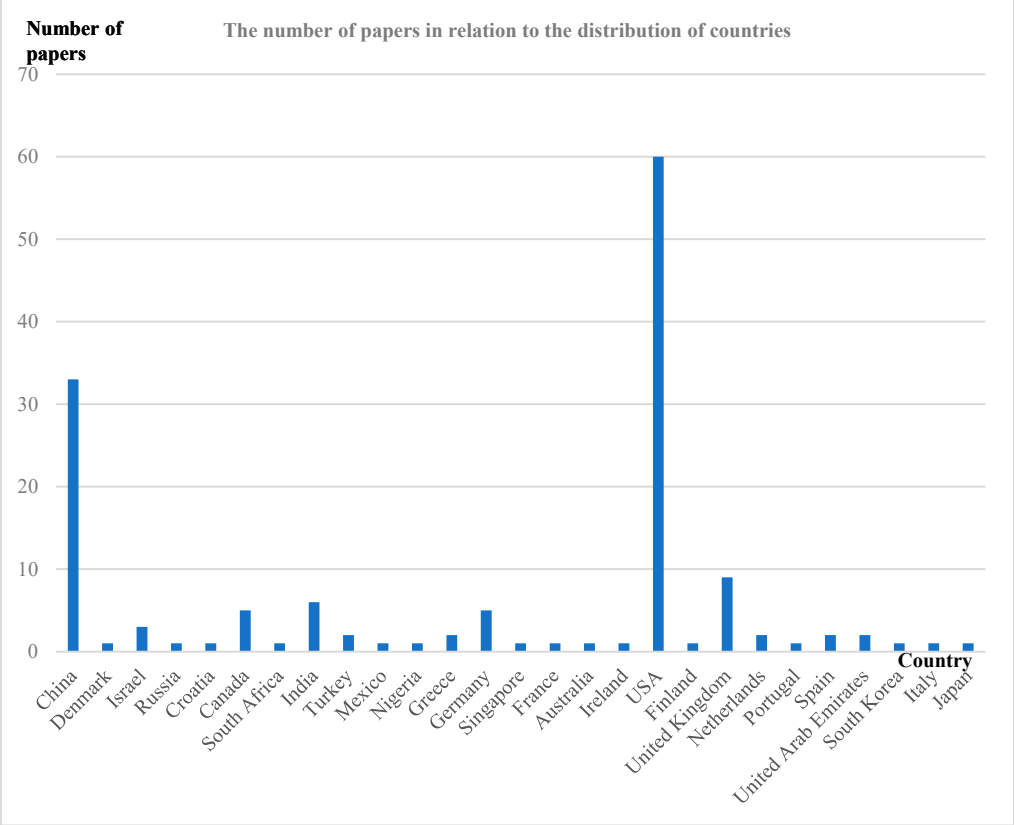

Figure 3. Graph of the number of papers in relation to the distribution of countries.

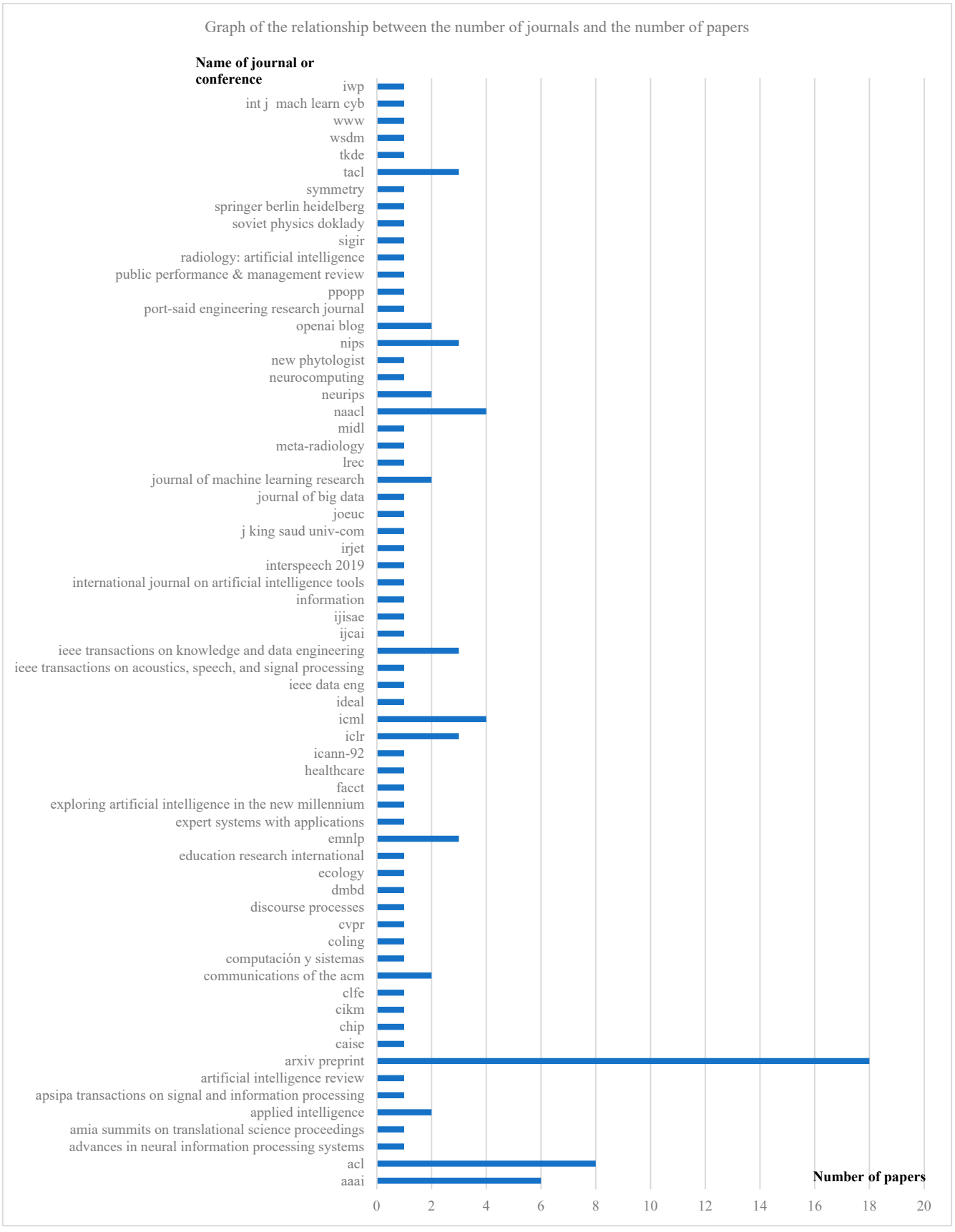

Figure 4. Graph of journals versus number of conferences and papers.

5. Text-Matching Methods

Text-matching methods cover several different techniques and strategies, each with its unique application scenarios and advantages. These methods can be broadly classified into the following categories: string-based methods, corpus-based methods, recursive neural network matching methods, sentence semantic interaction matching, matching methods based on graph structures, matching methods based on pre-trained language models, and matching methods based on large language models utilizing different features, as shown in Figure 5. String-based methods usually match by comparing the similarity of characters or strings in text and are suitable for simple and straightforward text comparison tasks. Corpus-based methods, on the other hand, rely on large-scale corpora to extract similarities and relationships between texts and achieve matching through statistical and probabilistic models. Recursive neural network matching methods utilize neural networks to deeply understand and represent the text, capturing the hierarchical information of the text through recursive structures to achieve a complex semantic matching. Sentence semantic interaction matching, on the other hand, emphasizes the semantic interaction and understanding between sentences, and improves the matching effect by modeling the interaction between sentences. Graph structure-based matching methods utilize graph theory knowledge to represent and compare texts, and match by capturing structural information in the text. In contrast, sentence vector-matching methods based on large language models improve the matching performance by capturing the deep semantic features of the text with the help of pre-trained language models from large-scale corpora. Among the large language model-based matching methods utilizing different features, the matching methods utilizing tagged semantic information consider the effect of tags or metadata on text matching and enhance the accuracy of matching by introducing additional semantic information. The fine-grained matching method pays more attention to the detailed information in the text and realizes fine-grained matching by accurately capturing the subtle differences between texts.

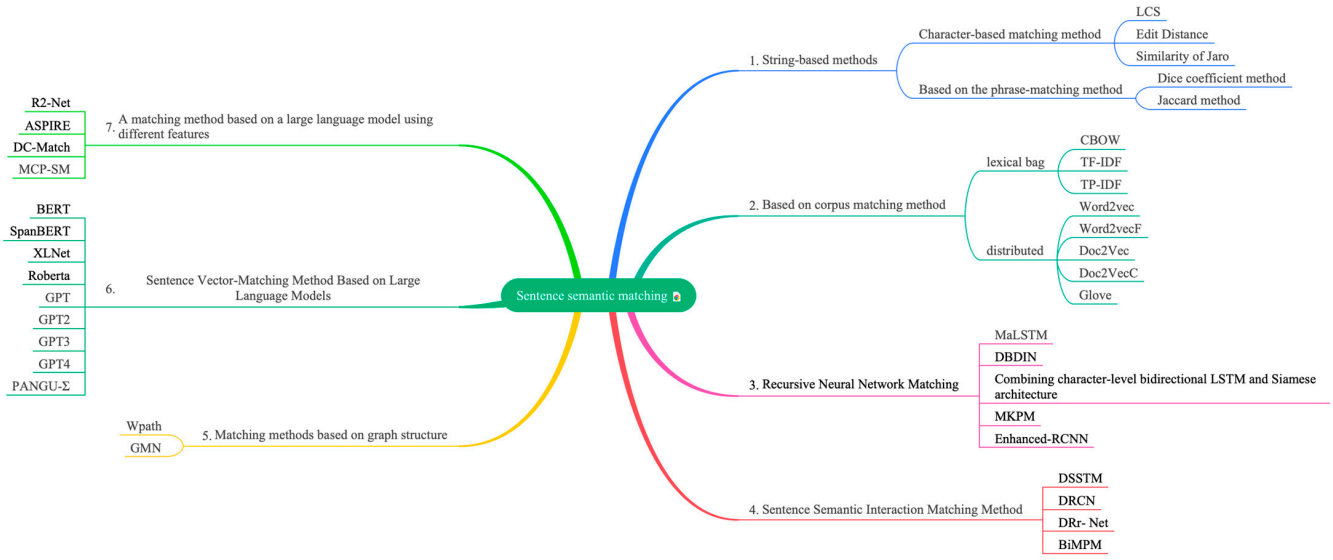

Figure 5. Matching method overview diagram.

5.1. Text-Matching Methods

String similarity metrics are an important text analysis tool that can be used to measure the degree of similarity between two text strings, which in turn supports string matching and comparison. Among these metrics, the cosine similarity measure is popular for its simplicity and effectiveness. The cosine similarity metric calculates the similarity between two texts based on the cosine of the angle of the vectors [33], which consists of calculating the dot product of the two vectors, calculating the modulus (i.e., length) of the respective vectors, and dividing the dot product by the product of the moduli; the result is the cosine of the angle, i.e., the similarity score. However, the cosine similarity is not sensitive to the

magnitude of the value, does not apply to non-positive spaces, and is affected by the feature scale. To improve its performance, methods such as adjusting the cosine similarity (e.g., centering the values), using other similarity measures (e.g., Pearson's correlation coefficient), or feature normalization can be adopted to improve the accuracy and applicability of the similarity calculation.

Several variants or improved forms of cosine similarity have also been proposed by researchers to address specific needs in different scenarios. These variants include weighted cosine similarity [34] and soft cosine similarity [35], among others. Weighted cosine similarity allows different weights to be assigned to each feature dimension to reflect the importance of different features; soft cosine similarity reduces the weight of high-frequency words in text similarity computation by introducing a nonlinear transformation. In addition, depending on the basic unit of division, string-matching-based methods can be divided into two main categories: character-based matching and phrase-based matching. These methods are illustrated in Figure 6, and their key ideas, methods, key issues, and corresponding solutions are summarized in detail in Table 2. These string-matching-based methods provide a rich set of tools and options for text analysis and processing to meet the needs of different application scenarios.

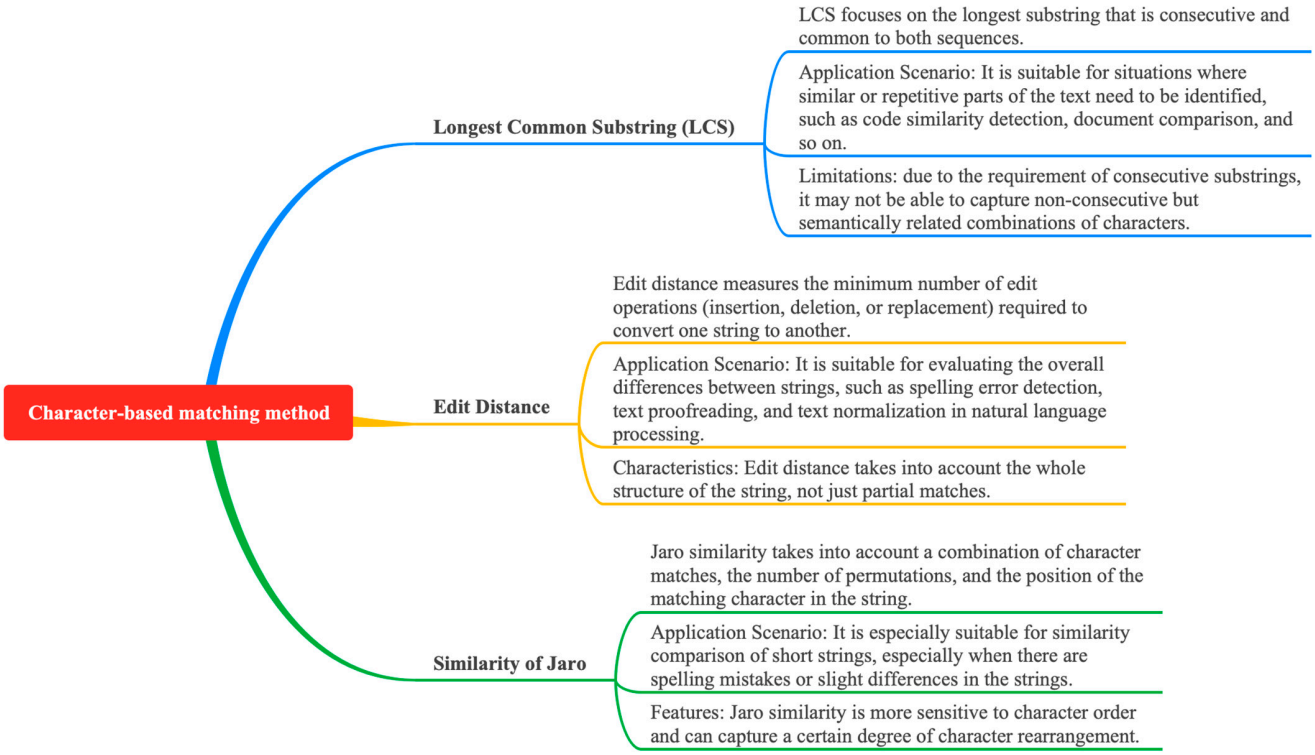

Figure 6. Overview of logical relationships based on character-matching methods.

Table 2. Overview of key ideas, methods, key issues, and solutions for string-matching-based approaches.

Models	Key Ideas	Approaches	Key Problems	Solutions
Longest common substring matching [1]	Computing an array of thresholds for the two sequences, iteratively calculating each threshold, and then step-by-step determining the length of their longest common subsequence and ultimately finding the longest common subsequence.	Inserting, deleting, and testing the affiliation of elements in a set, where all elements are restricted to the first n integers of the key operations, which in turn reduces the running time of the algorithm.	Scalability of the algorithm.	Parallelization [36] and distributed processing.

Table 2. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
Editorial distance [2]	The analogous construction enables the correction of wounding, and the insertion and inversion of optimal codes.	Correcting, inserting, and reversing binary code.	(1) Error correction may be limited. (2) Insertion errors are more difficult to recognize and handle. (3) Detection and correction of inversion errors. (4) Computational complexity and resource overhead may be high.	(1) More powerful error-correcting codes, such as Hamming or BCH codes, can be used. At the same time, a combination of error-correcting and error-detecting codes can be used. (2) Special synchronization marks or delimiters can be designed to separate blocks of data. In addition, coding schemes can be developed that can tolerate insertion errors. (3) Errors can be detected using methods such as parity, checksum, or cyclic redundancy check (CRC), and corrected by complex error-correcting codes for multi-bit inversions. (4) It is possible to optimize the implementation of error correction algorithms, to select an error correction method that suits the needs of the system, and to consider the use of specialized error correction circuits or gas pedals in the hardware design.
Jaro similarity [3]	Improving the record linkage method in the FellegiSunter model through string comparator metrics and enhanced decision rules.	(1) Uses the Jaro string comparator method, which is able to partially take into account differences between strings. (2) Proposed a method for adjusting the matching weights to trade off between perfect agreement and disagreement. (3) Use of the FellegiSunter model to deal with record linkage.	(1) Quality of empirical databases in record-matching applications. (2) Parameterization of generic string comparators.	(1) Enhance data cleaning and preprocessing, increase sample diversity [37], regularly update and maintain database. (2) Development of adaptive string comparators, introduction of partial and fuzzy matching mechanisms, optimization of parameters using domain knowledge, optimization and testing through iterations.
Dice coefficient method [4]	Lies in the use of the dice coefficient to measure the degree of overlap between two text collections (e.g., a collection of words or word embedding vectors) and thus assess their semantic similarity. The higher the dice coefficient, the closer the two texts are semantically.	The method usually includes text preprocessing, word embedding transformation, constructing a collection of word embedding vectors for the text, and calculating the intersection and concatenation of these collections, and finally using the dice coefficient formula to derive the similarity score.	(1) Inability to deal effectively with synonyms or semantically similar words. (2) Ignoring contextual information, leading to misinterpretation of semantics. (3) Lower computational efficiency when dealing with large-scale text.	(1) Using word embedding models that can capture synonyms and semantically similar words, such as context-based word embedding (e.g., BERT) [17]. (2) Using models or methods that can take contextual information into account, such as semantic similarity models based on deep learning [9]. (3) For large-scale text data, approximation algorithms [38] or optimized data structures can be used to reduce the computation.
Jaccard method [5]	The semantic similarity of two text collections is measured by calculating the ratio of their intersection to their concatenation, emphasizing the proportion of shared elements in the whole.	The Jaccard method first preprocesses the text to obtain the set of lexical items, then calculates the intersection and concatenation of these sets, and finally obtains the similarity score of the text by a ratio calculation.	(1) Insensitive to sparse text. (2) Ignores word weights. (3) Insensitive to word order.	(1) The word items are converted into word embedding vectors [7,39], and the similarity between the vectors is utilized to measure the semantic similarity of the text, so as to capture the semantic information in the sparse text. (2) When calculating the Jaccard similarity, the weight of the word items can be added, e.g., using methods such as TF-IDF [40] to weight the word items to reflect their importance in the text. (3) The Jaccard method can be used in conjunction with other semantic similarity measures, such as cosine similarity and edit distance, in order to comprehensively assess the similarity of the text and to reduce the bias that may be introduced by a single method.

Longest common substring (LCS) [1], edit distance [2], and Jaro similarity [3] are three methods based on character matching, which have their own focuses and logically complement each other. LCS focuses on consecutive character matching, edit distance measures the overall conversion cost, and Jaro similarity considers character matching, permutation, and positional information. These methods constitute a diversified toolbox

for character-matching analysis, which can be flexibly selected and combined according to the application scenarios. With the development of natural language processing technology, more new methods may emerge in the future to meet more complex text-processing needs.

The longest common substring matching algorithm iteratively determines the length of the longest common substring by calculating a threshold array to find the optimal solution, and it reduces the running time by optimizing the set operation [1]. For large-scale data and complex scenarios, parallelization and distributed processing schemes are proposed to improve the scalability and processing speed of the algorithm. Edit distance constructs binary codes by analogy to correct optimal code problems such as wounding, insertion, and inversion [2]. However, it faces problems such as a limited error correction capability, difficulty in recognizing insertion errors, and high computational complexity in inversion error detection and computation. The solutions include adopting more powerful error correction codes, designing synchronization markers, utilizing checksum methods, and optimizing the algorithm implementation. Jaro similarity is measured by a string comparator and optimized by the record-joining method of the FellegiSunter model, which uses the Jaro method to handle string discrepancies and adjust the matching weights [3]. However, the database quality and string comparator parameter settings affect the matching accuracy. We propose strategies such as data cleaning, increasing the sample diversity, and an adaptive string comparator to optimize the performance and accuracy of the matching algorithm.

Phrase-based matching methods include the dice coefficient method [4] and the Jaccard method [5], as shown in Figure 7. The difference between phrase-based matching methods and character-based matching methods is that the basic unit of processing is the phrase. The dice coefficient method in the phrase-based matching method is mainly used to measure the similarity of two samples. However, this method has limitations in dealing with synonyms, contextual information, and large-scale text. For this reason, context-based word embedding models, deep learning methods, and the optimization of data structures, in combination with approximation algorithms, are used to improve the accuracy and efficiency of the similarity assessment. In the field of natural language processing, it is often used to calculate the similarity of two text phrases. The principle of the dice coefficient is based on the similarity measure of sets [4], which is calculated as follows:

$$\text{dice coefficient} = (2 \times |X \cap Y|)/(|X| + |Y|) \quad (1)$$

where $|X \cap Y|$ denotes the size of the intersection of X and Y, and $|X|$ and $|Y|$ denotes the size (i.e., number of elements) of X and Y, respectively. The value domain of the dice coefficient is [0, 1]. When two samples are identical, the dice coefficient is 1; when two samples have no intersection, the dice coefficient is 0. Therefore, the larger the dice coefficient, the more similar the two samples are.

The Jaccard method in the phrase-based matching method is an algorithm used to calculate the similarity of two sets [5]. Its principle is based on the Jaccard coefficient in set theory, which measures the similarity of two sets by calculating the ratio of their intersection and concatenation. Specifically, given two sets A and B, the Jaccard coefficient is defined as the number of elements in the intersection of A and B divided by the number of elements in the concatenation of A and B. However, the method suffers from the problems of being insensitive to sparse text and ignoring word weights and word order. To improve these problems, it is possible to convert word items into word embedding vectors, add word item weights, and for these to be used in combination with other similarity measures to evaluate text semantic similarity more accurately. Its calculation formula is as follows:

$$\text{Jaccard}(A, B) = |A \cap B| / |A \cup B| \quad (2)$$

where $|A \cap B|$ denotes the number of intersection elements of sets A and B, and $|A \cup B|$ denotes the number of concatenation elements of sets A and B. The Jaccard coefficient has a value range of [0, 1], with larger values indicating that the two sets are more similar.

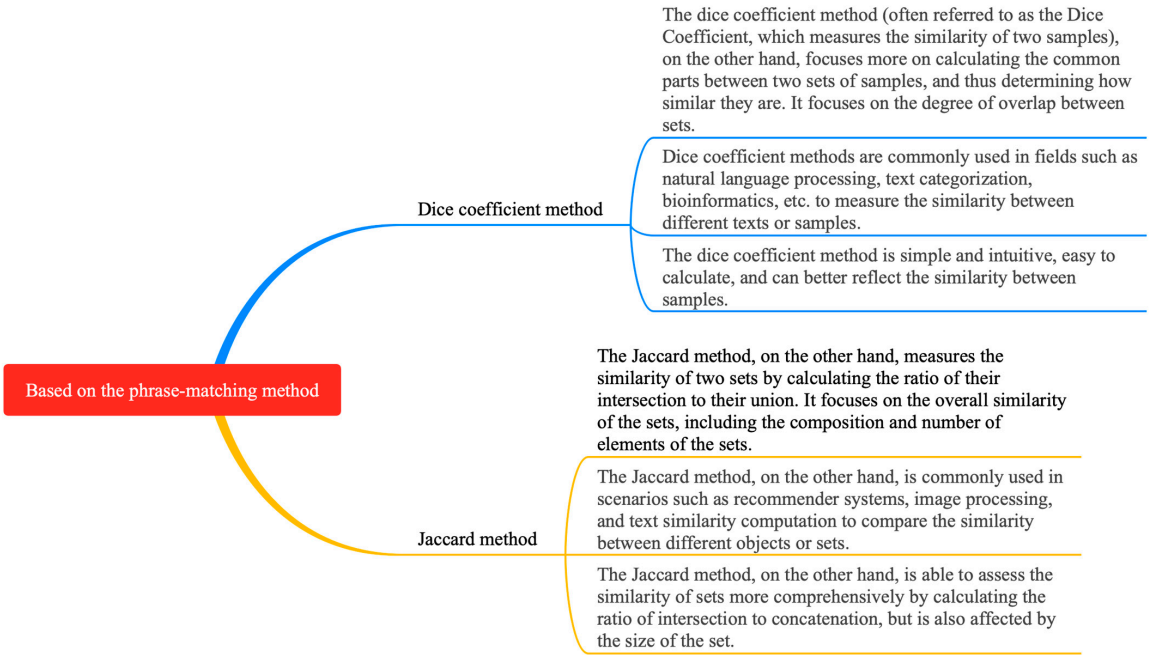

Figure 7. Overview of logical relationships based on phrase-matching methods.

5.2. Based on the Corpus-Matching Method

Typical methods for corpus-based matching include bag-of-words modeling such as BOW (bag of words), TF-IDF (term frequency-inverse document frequency), distributed representation, and matrix decomposition methods such as LSA (Latent Semantic Analysis), as shown in Figure 8. The key ideas, methods, key problems, and solutions for typical methods of corpus-based matching are summarized in Table 3.

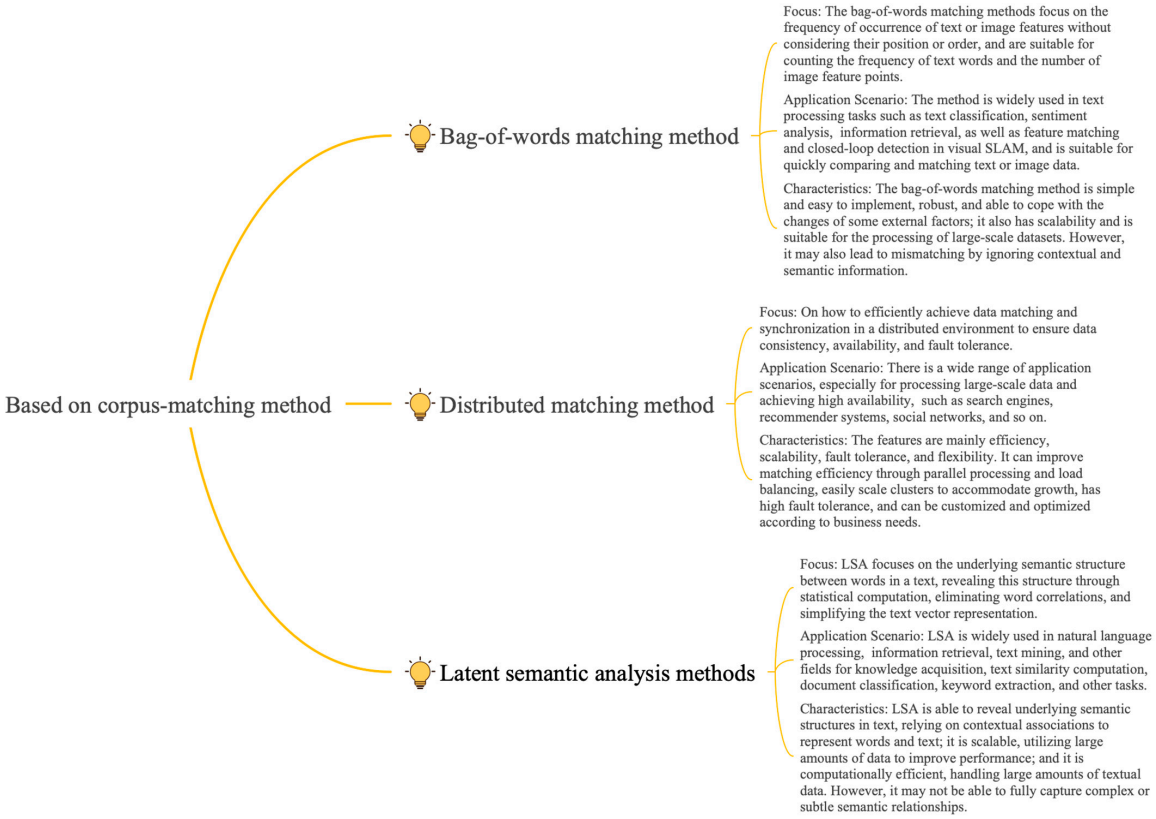

Figure 8. Corpus-based matching method.

Table 3. Overview of key ideas, methods, key problems, and solutions of corpus-based matching methods.

Models	Key Ideas	Approaches	Key Problems	Solutions
BOW [6]	Treating text (or images) as a collection of words (or features) without regard to the order or grammatical relationships of those words (or features).	(1) Extract features or vocabulary of the text or image. (2) Count the frequency of occurrence of these features or words in the text or image. (3) Combine these frequency values into a vector as a representation of the text or image.	Inability to relate to lexical order and semantic similarity.	The n-gram model [41] and word embedding techniques [7,39] can be used as an improvement scheme.
TF-IDF [40]	Evaluating the importance of words in a document by combining word frequency (TF) and inverse document frequency (IDF),	The TF-IDF algorithm evaluates the weight of a word in a document by calculating its frequency in the document and its prevalence in the whole corpus to achieve an efficient processing of the text.	Simplified processing of IDF may lead to loss of information and underestimation of keywords in similar texts.	TF-IWF improvement methods can be used or combined with other text representation techniques, such as word embedding models [7,39].
TP-IDF [42]	Improvement of the traditional TF-IDF method by combining the positional information of lexical items in the document and the inverse document frequency, especially when dealing with short texts.	By combining the position information of word items in documents with the TF-IDF model, the accuracy of assessing the importance of word items in documents is improved. This method is especially suitable for short text analysis, which can effectively use the word item position information to optimize the calculation of word item weights.	How to rationally define positional factors to accurately reflect lexical item importance, the lack of stability of lexical item frequency and positional information in short texts, the interference of deactivated and noisy lexical items, and the computational complexity that may be associated with dealing with large document collections.	The efficiency and accuracy of tp-idf in processing textual data is improved by optimizing the definition of positional factors through experiments or domain knowledge, enhancing the accuracy of word weights by combining them with other textual features, deactivating words on documents to reduce noise interference, and optimizing algorithms and data structures to reduce computational complexity.
Word2vec [7]	Quantitative representation of semantic information is achieved by mapping words to continuous vectors through neural network models.	Two training methods, CBOW (continuous bag of words) and Skip-gram(word-skipping model), are used to learn word vectors by context-predicting words and word-predicting context, respectively.	(1) Existence of ignored word order. (2) Inability to handle multiple meanings of words. (3) OOV words.	(1) Models such as RNN [43] or Transformer [16] can be combined to capture word order information. Dynamic word vector models such as ELMo [44] and BERT [17] are used to handle multiple meanings of a word. (2) Introducing character embedding [7,39] or pre-trained language models to solve the OOV word problem.
Word2VecF [45]	Word2VecF is a flexible and controllable implementation of Word2Vec that allows users to customize word embeddings by providing word–context tuples to train models using global contextual information.	The method includes data preparation, preprocessing, model training, and word embedding generation. The model is directly trained to generate custom word embeddings by preprocessing user-provided tuple data, combined with vocabulary lists and subsampling.	The impact of data quality on model performance, the complexity of parameter tuning, and the consumption of computational resources when dealing with large-scale datasets.	Solutions include data cleansing and filtering to improve data quality, parameter optimization to find the best configurations, use of distributed computing to reduce computational resource consumption, and model integration to improve the accuracy and robustness of word embeddings.
Doc2Vec [46]	Doc2Vec's Distributed Memory model achieves the capture of semantic information at the paragraph level through a combination of paragraph vectors (representing paragraph topics) and local context words, thus improving the accuracy of text representation and prediction.	The method trains the model by predicting the next word in the text based on the input of paragraph vectors and local context words. During the training process, the paragraph vectors and word vectors are updated to capture their respective semantic information, and the parameters are optimized iteratively to improve the predictive performance of the model.	(1) High computational cost: when Doc2Vec handles a large number of documents, the computational cost increases significantly with the number of documents. (2) Difficulty in scaling: For new documents, Doc2Vec usually needs to retrain the whole model, which limits its ability in dealing with large-scale or dynamically changing datasets.	(1) Optimization algorithms: reduce training cost of Doc2Vec and improve computational efficiency by introducing optimization algorithms such as negative sampling and hierarchical softmax. (2) Incremental learning: adopting an incremental learning approach allows Doc2Vec to update only some of the parameters when processing new documents, avoiding global retraining and enhancing scalability.
Doc2VecC [47]	A new approach based on Doc2Vec is introduced to capture the global context of a document. It does this by randomly sampling words in a document and computing the average of these word vectors to represent the semantics of the entire document, and then using this document vector to predict the current word along with local context words.	First, a portion of the words are randomly extracted from the document in each training iteration; then, the average of the vectors of these extracted words is computed to form a document vector; and finally, this document vector and the local context words are used to predict the currently processed word.	(1) The sparsity of the training data may lead to limited model-training effectiveness. (2) The generated document vectors are usually high-dimensional and dense, which makes the vectors difficult to understand and visualize intuitively. (3) Processing all the words in the whole document may lead to computational inefficiency.	Increase the amount of training data by collecting more relevant documents or using data augmentation techniques; secondly, explore interpretable vector representations such as topic modelling to improve the interpretability and visualization of the document vectors; and lastly, construct document vectors using only some of the words in documents to improve the computational efficiency of the training process.

Table 3. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
glove [39]	Constructing word vector representations to capture the global statistical information of words in the corpus through co-occurrence matrices.	Construction of co-occurrence matrices and iterative optimization of word vectors	Insufficient processing of rare words, missing local contextual information, and high computational complexity.	External knowledge bases [48] can be introduced to strengthen the rare word representation, combined with local context methods to enhance semantic understanding, optimized algorithms and model structures to reduce the computational burden, and distributed computing frameworks to accelerate the processing of large-scale corpora.
LSA [8] (Latent Semantic Analysis)	Revealing implicit semantics in text by mapping high-dimensional document space to low-dimensional latent semantic space.	By performing a singular value decomposition or a non-negative matrix decomposition of this matrix, LSA is able to obtain a low-dimensional topic vector space and map words and text to that space.	(1) Dimensionality catastrophe. (2) Data sparsity. (3) Lack of interpretability. (4) Inability to handle word order and syntactic information. (5) Sensitivity to parameters and decomposition methods.	(1) Feature selection [49], dimensionality reduction techniques. (2) Data preprocessing [50], sparse matrix processing techniques. (3) Topic labeling, visualization techniques. (4) Combination of other text representation methods: e.g., word embedding techniques such as word2vec [7], BERT [17], use of more complex models: e.g., recurrent neural network (RNN) [43] or Transformer [16], etc. (5) Parameter tuning, comparing different decomposition methods, e.g., evaluating the performance of different decomposition methods, such as SVD, NMF, etc., on a specific task and selecting the most suitable method

Typical approaches to bag-of-words models are BOW [6], TF-IDF [40], and its variant TP-IDF [42]. The bag-of-words model reduces the text to a collection of lexical frequencies, ignores the order and semantics, and is not precise enough for complex text processing. To improve this model, the n-gram model can be used to consider word combination and order, or word embedding technology can be used to capture semantic relationships, so as to enhance the accuracy and performance of text processing.TF-IDF is an important method in the bag-of-words model, which evaluates the importance of words by combining the word frequency and the inverse document frequency, and extracts key information effectively. However, IDF simplified processing may lead to information loss and the underestimation of similar text keywords. For improvement, text representation techniques such as TF-IWF, or ones combined with word embedding models, can be used to represent text content more comprehensively.TP-IDF is an improvement to the traditional TF-IDF method, which combines the positional information of the word items in the document to assess the word importance more accurately, and it is especially suitable for short text analysis. However, it also faces the challenges of defining location factors, stability, noise interference, and computational complexity in its application. By optimizing the position factor definition, combining other text features, deactivating word processing, and algorithm optimization, TP-IDF can improve the efficiency and accuracy of text data processing.

Distributed approaches cover a range of techniques based on dense vector representations [51], such as Word2vec [7], Word2VecF [45], Doc2Vec [46], Doc2VecC [47], and glove [39]. These techniques represent vectors using fixed-size arrays where each element contains a specific value, thus embedding words, documents, or other entities into a high-dimensional space to capture their rich semantic information. Dense vector representations have been widely used in natural language processing tasks such as text categorization, sentiment analysis, recommender systems, etc., because of their simplicity, intuitiveness, and ease of processing. Word2Vec, as a representative of this, achieves a quantitative representation of semantics by mapping words into continuous vectors through neural networks. However, CBOW and Skip-gram, as their main training methods, have problems such as ignoring word order, multiple meanings of one word, and OOV words. To overcome these limitations, researchers have introduced dynamic word vectors and character embeddings or pre-trained language models, in conjunction with models such as RNN, Transformer,

etc., to significantly improve the performance and applicability of Word2Vec. Word2VecF is a customized version of Word2Vec that allows users to customize word embeddings by supplying word–context tuples, to be able to exploit global context information. However, it also faces challenges such as data quality, parameter tuning, and computational resources. To overcome these challenges, researchers have adopted technical approaches such as data cleaning, parameter optimization, distributed computing, and model integration to improve the accuracy and efficiency of word embeddings.

The glove model, on the other hand, captures global statistics by constructing co-occurrence matrices to represent the word vectors, but it suffers from the challenges of dealing with rare words, missing local contexts, and high computational complexity. To improve the performance and utility of the glove, researchers introduced an external knowledge base, incorporated a local context approach, and used optimized algorithms and model structures, as well as a distributed computing framework for training. These improvements make the glove model more efficient and accurate in natural language processing tasks.

The core idea of LSA (Latent Semantic Analysis) is to reveal the implicit semantics in the text by mapping the high-dimensional document space to the low-dimensional latent semantic space [8]. It uses methods such as singular value decomposition to realize this process. However, LSA suffers from semantic representation limitations, sensitivity to noise, and computational complexity. To improve the performance, it can be combined with other semantic representation methods, data preprocessing, and cleaning, as well as the optimization of the algorithm and computational efficiency. Through these measures, LSA can be utilized more effectively for text processing and semantic analysis.

5.3. Recursive Neural Network-Based Matching Method

Neural network architectures are at the heart of deep learning models and determine how the model processes input data and generate output. These architectures consist of multiple layers and neurons that learn patterns and features in the data by adjusting the weights of connections between neurons. Common neural network architectures include feed-forward neural networks, recurrent neural networks, and convolutional neural networks, each of which is suitable for different data-processing needs. When dealing with sentence-matching tasks, specific architectures such as twin networks and interactive networks can efficiently compute the similarity between sentences. In addition, the attention mechanism [52] improves the efficiency and accuracy of processing complex data by simulating the allocation of human visual attention, enabling the model to selectively focus on specific parts of the input data. Recursive neural network matching methods, such as those based on LSTM, BiLSTM, and BiGRU, are shown in Figure 9. They further combine neural network architectures and attention mechanisms to provide powerful tools for processing sequential data. By properly designing and adapting these architectures, deep learning models can be constructed to adapt to various tasks. The key ideas, methods, key issues, and solutions for the recursive neural network matching approach are summarized in Table 4.

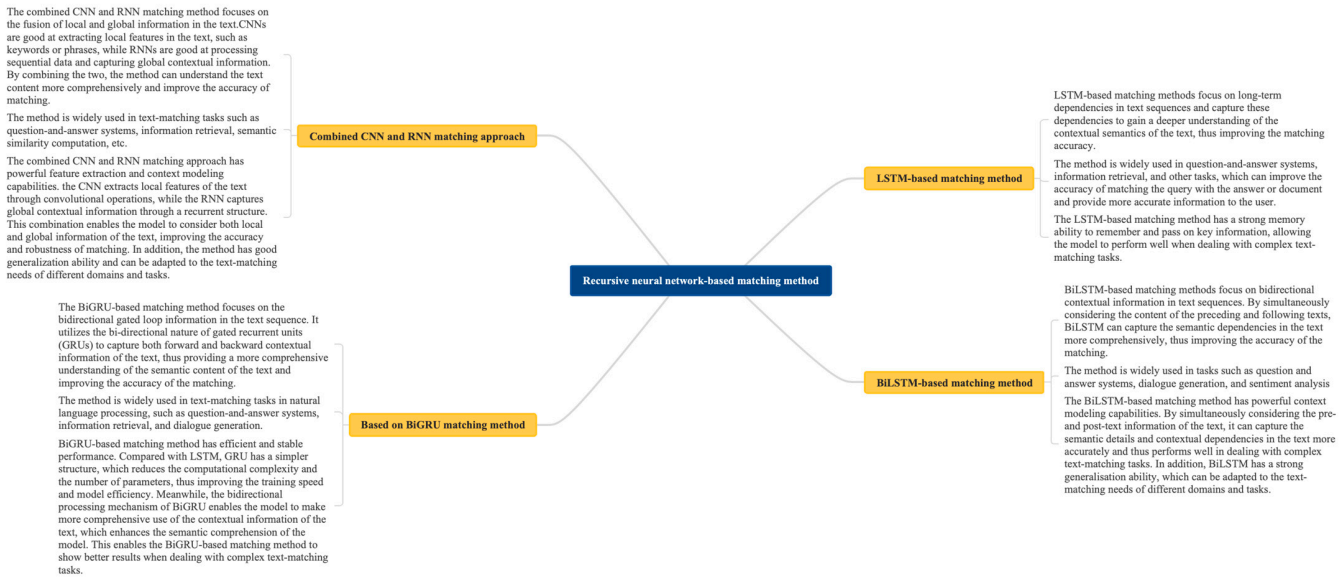

Figure 9. Overview of logical relationships of recursive neural network-based matching methods.

Table 4. Overview of key ideas, methods, key problems, and solutions for recursive neural matching-based approaches.

Models	Key Ideas	Approaches	Key Problems	Solutions
MaLSTM [9]	The model captures the core meaning of a sentence through fixed-size vectors and learns to structure the representation space to reflect semantic relationships.	The model utilizes LSTM to construct sentence representations and predicts semantic similarity by computing the Manhattan metric between vectors.	(1) The effect of data sparsity and noise on the model's ability to capture low-frequency lexical and semantic details. (2) Limitations of models in understanding complex semantic relationships. (3) Computational efficiency issues limit application of models on large-scale datasets.	(1) Data preprocessing [50] and enhancement techniques [53]. (2) Introduction of attention mechanisms [52]. (3) Optimization of model structure and parameters, migration learning using pre-trained models [54].
DBDIN [55]	Lies in capturing semantic relevance from both directions and deep interaction through multiple attention-based interaction units for more accurate semantic matching.	Precise semantic matching between sentences is achieved through two-way interaction and attention mechanisms, and interaction effects are enhanced through deep fusion and self-attention mechanisms.	(1) Higher model complexity. (2) Models may have difficulty accurately capturing certain complex or ambiguous semantic relationships. (3) Affected by noisy data or uneven distribution of training data.	(1) Adopt more efficient attention mechanisms or reduce number of network layers. (2) Introduce more contextual information or utilize pre-trained models (3) Data enhancement [56] and noise filtering.
Combining character-level bi-directional LSTM and Siamese architectures [10]	Deep earning models using bi-directional LSTM and Siamese architecture to learn character sequence similarity, capture semantic differences, and ignore non-semantic differences.	String embeddings are compared via bi-directional LSTM and Siamese architectures; models are trained using similarity information, and datasets are augmented to enhance generalization.	(1) High model complexity. (2) Excessive attention to detail. (3) Sensitivity to noisy data.	(1) Reduce number of network layers or use efficient algorithms to reduce complexity and reduce computational resource consumption. (2) Combine word-level or sentence-level embedding to compensate for the information loss of character-level processing. (3) Remove noisy data and outliers to improve model robustness. Reduce the risk of overfitting and improve model stability and generalization ability.

Table 4. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
MKPM [11]	Proposes a sentence matching method based on keyword pair matching that combines attention and two-way task architecture to improve matching accuracy.	The method utilizes an attention mechanism to filter keyword pairs and models semantic relationships through a two-way task that combines word-level and sentence-level information to improve matching.	(1) Keyword pair selection problem. (2) Complex semantic relationship-processing limitations. (3) Model complexity and computational cost.	(1) Improve keyword pair extraction and selection. (2) Expand or dynamically adjust the number of keyword pairs. (3) Reduce computational cost and improve model efficiency and scalability by optimizing model structure and algorithms.
Enhanced-RCNN [57]	Enhanced-RCNN combines the advantages of CNN and attention-based RNN for sentence similarity learning, aiming to capture similarities and differences between sentences more accurately, while reducing computational complexity.	Siamese multilayer CNN is utilized to extract sentence key information, attention-based RNN is employed to capture the interaction effects between sentences, and a fusion layer is designed to integrate the outputs of the CNN and RNN in order to calculate the similarity score.	(1) Limited ability to process semantic relations. (2) Dataset limitations. (3) As model complexity increases, consumption of computational resources and time may become a challenge	(1) Enhance the model's understanding of complex semantic relationships by introducing advanced techniques such as context embedding or pre-trained language models. (2) Improve the generalization ability of the model to adapt to more scenarios and domains by increasing the size and diversity of the dataset. (3) Optimizing model structure and algorithms, exploring integrated learning [58] and migration learning [54].

The MaLSTM model predicts semantic similarity by capturing the core meaning of sentences through LSTM and fixed-size vectors [9]. However, there are problems with low-frequency vocabulary processing, capturing complex semantic relations, and computational efficiency. This study proposes strategies such as data preprocessing, introducing an attention mechanism, optimizing the model structure, and utilizing pre-trained models to improve the model's effectiveness on the sentence similarity learning task. The DBDIN model [55] achieves precise semantic matching between sentences through two-way interaction and an attention mechanism but faces the problems of a high consumption of computational resources, difficulty in capturing complex semantics, and limited generalization ability. Optimizing the model structure, introducing contextual information and pre-training models, and combining multiple loss functions and regularization techniques can improve the performance and stability of DBDIN on the sentence similarity learning task. The deep learning model proposed by Neculoiu P et al. combines a character-level bi-directional LSTM and Siamese architecture [10] to learn character sequence similarity efficiently but suffers from high complexity, and it may ignore overall semantic information and show a lack of robustness. Optimizing the performance can be accomplished by simplifying the model structure, introducing high-level semantic information, cleaning the data, and incorporating regularization techniques.

Lu X et al. [11] proposed a sentence-matching method based on multiple keyword pair matching, which uses the sentence pair attention mechanism to filter keyword pairs and model their semantic information, to accurately express semantics and avoid redundancy and noise. However, the method still has limitations in keyword pair selection and complex semantic processing. The accuracy and application scope of sentence matching can be improved by introducing advanced keyword extraction techniques, extending the processing power of the model, and optimizing the algorithm. Peng S et al. proposed the Enhanced-RCNN model [57], which fuses a multilayer CNN and attention-based RNN to improve the accuracy and efficiency of sentence similarity learning. However, the model still has challenges in handling complex semantics, generalization ability, and computational resources. The model performance can be improved by introducing new techniques, optimizing models and algorithms, and expanding the dataset.

5.4. Semantic Interaction Matching Methods

Typical models of semantic interactive matching methods include DSSTM [12], DRCN [13], DRr- Net [14], and BiMPM [15], as shown in Figure 10. As shown in Table 5, this study outlines the key ideas, methods, key issues, and solutions for interaction matching methods.

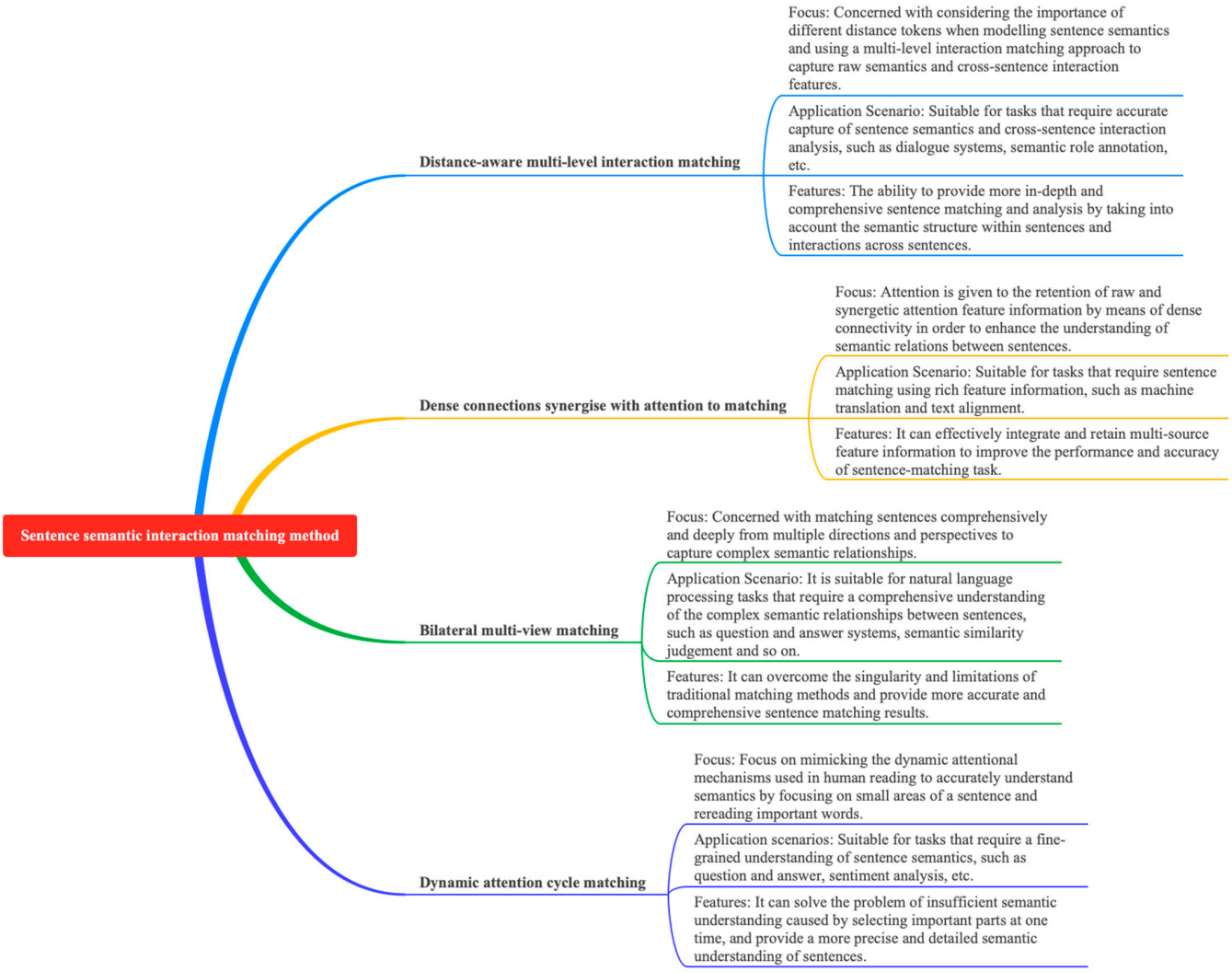

Figure 10. Overview of logical relations of sentence semantic interaction matching methods.

Table 5. Overview of key ideas, methods, key problems, and solutions for interaction matching methods.

Models	Key Ideas	Approaches	Key Problems	Solutions
DSSTM [12]	This model considers the importance of distance markers and combines self-attention and co-attention mechanisms to enhance sentence semantic modeling and extract cross-sentence interaction features for sentence semantic matching tasks.	The sentences are first encoded and embedded, then the distance-aware self-attention mechanism is used to enhance the semantic modeling, followed by the application of a co-attention layer to extract cross-sentence interaction features, and the fusing of these features into a multilevel matching function, and finally the model is validated by experiments in terms of effectiveness and performance.	(1) Computational efficiency issues. (2) Inadequate capture of deep semantic structure. (3) The design of the multilevel matching function may need to be further optimized to better capture the subtle differences and diversity between sentence pairs.	(1) Optimize model structures and algorithms. (2) Utilize more complex semantic representations such as pre-trained language models or knowledge graphs. (3) Optimizing multilevel matching function, designing more reasonable loss function and training strategy.

Table 5. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
DRCN [13]	The architecture effectively preserves the original and synergetic attention feature information by means of dense connectivity, which enhances the understanding of semantic relations between sentences and improves the performance of the sentence-matching task.	A sentence-matching architecture is proposed that combines a collaborative attention mechanism and dense connectivity to capture and preserve semantic information and feature representations between sentences, while an autoencoder is used to compress features and maintain model efficiency.	(1) Increased computational complexity and feature dimensions. (2) Information loss due to autoencoder.	(1) Reduce computational complexity by utilizing advanced feature selection algorithms to reduce redundant features and introducing lightweight network structures to reduce the number of parameters. (2) Employ finer reconstruction loss functions to ensure that key information is retained as much as possible when compressing features. Regularization terms are introduced to constrain the coding process and reduce information loss.
DRr- Net [14]	This method solves the problem of insufficient semantic comprehension that may result from selecting important parts at once by simulating the dynamic attention mechanism of human reading, focusing on a small part of the sentence area and rereading important words in each step, so as to understand the sentence semantics more accurately.	(1) Introducing Attention Stack Gated Loop Unit (ASG). (2) Design of dynamic rereading unit (DRr).	(1) Insufficient richness of training data. (2) Impact of data preprocessing quality.	(1) Optimizing training strategies and increasing training data. (2) Improving data preprocessing techniques [50], incorporating linguistic knowledge and rules.
BiMPM [15]	A bilateral multi-view matching model is used to achieve the comprehensive and in-depth matching of sentences and improve processing performance.	BiLSTM is utilized to encode sentences, obtain matching information through bilateral matching and multi-view matching mechanisms, aggregate these into matching vectors, and make decisions to improve matching accuracy.	(1) Higher model complexity. (2) Limited ability to handle long sentences. (3) Reduced performance in cross-domain or cross-language tasks.	(1) Reduce complexity by optimizing the model structure, reducing the number of parameters, or using a lightweight network structure to reduce computation and storage requirements. (2) Segmentation processing strategy to ensure that the model can adequately capture key information when processing long sentences. (3) Utilizing domain-adaptive techniques or cross-language pre-training models, incorporating external knowledge [48] or resources to enhance the context-awareness of the models in cross-domain or cross-language tasks.

Deng Y et al. proposed constructing a DSSTM model [12], which utilizes distance-aware self-attention and multilevel matching to enhance sentence semantic matching accuracy but suffers from computational inefficiency and insufficient deep semantic capture, and it is suggested to optimize the model structure and introduce complex semantic representations to improve the performance. Kim S et al. [13] proposed a novel recurrent neural network architecture, which enhances inter-sentence semantic comprehension through dense connectivity and collaborative attention but suffers from rising feature dimensions' computational complexity and information loss. It is suggested to optimize the feature selection and compression techniques to improve the model performance. Zhang K et al. [14] proposed a sentence semantic matching method that simulates the dynamic attention mechanism of human reading, which improves comprehension comprehensiveness and accuracy

but is limited by the quality of training data and preprocessing. It is suggested to optimize the training strategy, increase the diversity of data, improve the preprocessing techniques, and combine linguistic knowledge to enhance the model performance. The BiMPM model proposed by Wang Z et al. [15] improves the performance of NLP tasks through multi-view matching but suffers from complexity and cross-domain or cross-language adaptability problems. Optimizing the model structure and introducing the attention mechanism are schemes that help to improve the model's accuracy and adaptability.

5.5. Matching Methods Based on Graph Structure

Typical approaches for matching based on graph structures include knowledge graphs and graph neural networks, as shown in Figure 11. This study also outlines the key ideas, methods, key problems, and solutions for graph-based matching methods, as shown in Table 6.

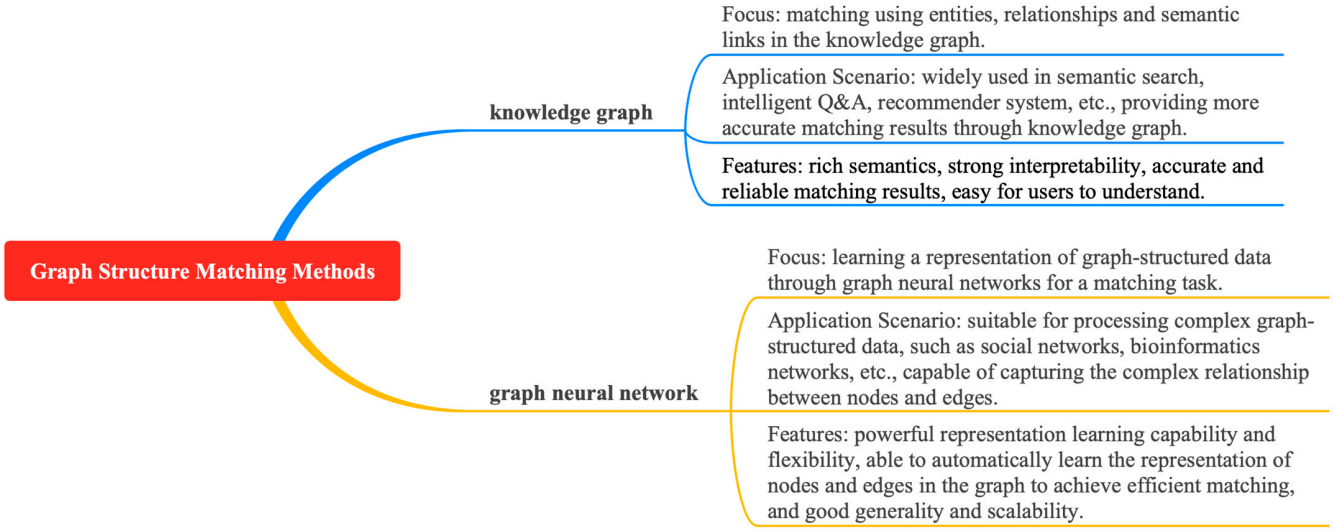

Figure 11. Overview of logical relationships of graph structure matching methods.

Table 6. Overview of key ideas, methods, key problems, and solutions based on graph matching methods.

Models	Key Ideas	Approaches	Key Problems	Solutions
Wpath [59]	A new semantic similarity measure, wpath, is proposed, which combines structural information and the information content of concepts in a knowledge graph to assess semantic similarity more comprehensively and accurately.	The final semantic similarity is derived by calculating the information content and the shortest path length of the concepts, and weighting the path length using the information content. This approach integrates information of multiple dimensions of concepts in the knowledge graph and improves the accuracy of the similarity calculation.	(1) Sparse knowledge graph applicability. (2) The computation of wpath methods may become complex and time-consuming in large knowledge graphs. (3) Information content, while reflecting conceptual universals, is not sufficient to fully capture semantic relationships.	(1) Increase path diversity to improve reliability of similarity measures in sparse graphs. (2) Reduce computational complexity in large-scale graphs by optimizing graph structure or using efficient algorithms. (3) Evaluate semantic similarity more comprehensively by combining contextual information, co-occurrence relations, etc. (4) Adjust the weights according to the knowledge graph characteristics to obtain a more accurate similarity measure.

Table 6. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
GMN [60]	A neural graph matching network framework (GMN) is proposed to solve the performance problem in Chinese short text matching by utilizing multi-granularity input information and an attention graph matching mechanism.	The GMN method uses word lattice graphs as inputs, updates node representations through the graph matching attention mechanism, fuses multi-granularity information, and can be combined with pre-trained language models to improve matching performance.	(1) Higher computational complexity. (2) Insufficient interpretability. (3) Dependence on pre-trained models. (4) Data quality and quantity challenges.	(1) Improve the graph matching attention mechanism with distributed [61] or parallel processing techniques [62]. (2) Enhance the study of internal mechanisms to provide visualization or explanatory methods. (3) Explore structures that do not rely on external models or utilize transfer learning methods [54]. (4) Expand training data by utilizing techniques such as data augmentation [63] and semi-supervised learning.

Zhu G et al. proposed the wpath method [59] to measure semantic similarity by combining path length and information content in a knowledge graph. However, there are problems of graph sparsity, computational complexity, and information content limitations. It is suggested to introduce multi-source paths, optimize the graph structure and algorithm, integrate other semantic features, and adjust the weights to improve the applicability and performance of the wpath method. Chen L et al. proposed a GMN framework to improve the accuracy of Chinese short text matching through multi-granularity input and attention graph matching [60], but there are problems such as high complexity and poor interpretability. It is suggested to optimize the graph matching mechanism, enhance interpretability, reduce model dependency, and use data enhancement and automation tools to improve performance.

5.6. Sentence Vector Matching Method Based on Large Language Models

The embedding technique is an important means to transform textual data, such as words, phrases, or sentences, into fixed dimensional vector representations. These vectors not only contain the basic features of the text but are also rich in semantic information, enabling similar words or sentences to have similar positions in the vector space. In the sentence-matching task, the embedding technique is particularly important because it can transform sentences into computable vector representations, thus facilitating our subsequent similarity calculations and comparisons. Among the many embedding methods, models such as Word2Vec, Glove, and BERT have shown excellent performance. They can capture the contextual information of words and generate semantically rich vector representations, providing strong support for NLP (Natural Language Processing) tasks such as sentence matching. In recent years, significant progress has been made in sentence vector matching methods based on large language models. Models such as BERT [17], Span BERT [18], XLNet [19], Roberta [20], GPT family, and PANGU-Σ [25] are among the leaders. As shown in Figure 12, these models have their features and advantages in sentence vector representation. This study delves into the key ideas and methods of these models, as well as the key problems and solutions encountered in practical applications. As shown in Table 7, we provide an overview of the performance of these models, aiming to help readers better understand their principles and applicable scenarios. Through this study, we can see that sentence vector matching methods based on large language models have become one of the important research directions in the field of NLP. These models not only improve the accuracy and efficiency of sentence matching but also provide rich feature representations for subsequent NLP tasks. With the continuous development of technology, we have reason to believe that these models will play an even more important role in future NLP research.

Figure 12. Overview of logical relations of sentence vector matching method based on large language model.

Table 7. Overview of key ideas, methods, key issues, and solutions based on large language model.

Models	Key Ideas	Approaches	Key Problems	Solutions
BERT [17]	It lies in the use of a bi-directional encoder and self-attention mechanism to capture the deep semantic and contextual information of the text, and it realizes the efficient application of the model through two phases: pre-training and fine-tuning.	BERT uses the Transformer model as an infrastructure and is pre-trained with large-scale unlabeled text data.	(1) High consumption of computing resources. (2) Insufficient learning. (3) Difficulty in processing long text.	(1) Model compression [64,65] and optimization, distributed training. (2) Pre-training data enhancement [63], task-specific fine-tuning. (3) Text truncation and splicing, designing model variants for long texts [18].

Table 7. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
SpanBERT [18]	SpanBERT optimizes the pre-training process by masking successive text segments and predicting their contents to better capture contextual and semantic information in the text, and it is particularly suitable for tasks that require the recognition and exploitation of text segments.	(1) Masking continuous text segments. (2) Prediction using boundary representation.	(1) Over-reliance on contextual information. (2) Randomness of masking strategies.	(1) Balanced learning of contextual information and segment internal details is achieved by designing adaptive masking method that combines multiple masking strategies. (2) Adopting means such as increasing the number of training rounds, expanding the dataset, or using advanced optimization algorithms, assisted by unsupervised pre-training tasks, to improve the stability and generalization ability of the model.
XLNet [19]	Combined modeling of mask language models and alignment language models,	(1) Randomized masking. (2) Target position sampling. (3) Self-attention and bi-directional contexts. (4) Pre-training and fine-tuning.	(1) Computational complexity problems. (2) Long sequence-modeling problem. (3) Large-scale pre-training problem. (4) Explosive growth of permutations problem. (5) Limitations of autoregressive models.	(1) Approximate attentional mechanisms or sparse attention [66]. (2) Chunked processing and intermediate layer delivery [67]. (3) Transfer learning [68]. (4) Approximate sampling and masking techniques [69]. (5) Localized windows and window relative position coding [70].
Roberta [20]	The key idea of Roberta's model is to further improve the model's performance on natural language processing tasks by improving and optimizing the structure and training strategy of the BERT model.	This study optimizes BERT's performance by replicating the BERT pre-training process, tuning hyperparameters, and evaluating different training data volumes, and it demonstrates that its performance is comparable to other state-of-the-art models.	(1) Hyperparameter tuning may not be comprehensive, leaving out some of the more optimal model settings. (2) Training a language model requires significant computational resources and time, limiting in-depth exploration. (3) The study mainly focuses on the BERT model itself and does not consider combining it with other state-of-the-art models, which may limit performance improvement.	(1) Expand hyperparameter searches to incorporate automated optimization techniques, such as Bayesian optimization or grid searches, to fully explore model configurations. (2) Utilize distributed computing or cloud computing resources to accelerate the training process and improve research efficiency. (3) Combine BERT with other advanced models or techniques, such as integrated learning [58] and knowledge distillation [71], to enhance model performance. Also draw on other domain optimization methods, such as transfer learning [54], meta-learning, etc. to improve the pre-training process.
GPT [21]	Understanding natural language through unsupervised learning, generating coherent text sequences in a unidirectional decoder structure, and applying it to a variety of NLP tasks through a pre-training–fine-tuning framework.	Pre-training and fine-tuning with a one-way Transformer decoder, learning language representations from large-scale data through self-supervised learning, and applying it on specific NLP tasks to improve performance.	(1) Problem of bias in training data. (2) Problem of dialog consistency. (3) Problem of processing long text.	(1) Diverse training data [72]. (2) Introduction of dialog history [73]. (3) Segmented processing [74].

Table 7. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
GPT2 [22]	Pre-training using the Transformer network structure to generate text and model the contextual context by autoregression to learn a generic language representation.	Pre-training, Transformer network architecture, Contextual Context Modeling, Multi-Layer Representation and Multi-Headed Attention, and fine-tuning.	(1) Can only realize the one-way text generation problem. (2) The problem of more parameters and high training costs. (3) Lack of common sense and world knowledge problem. (4) The problem of being susceptible to input bias. (5) Lack of consistency and controllability problem. (6) Difficulty in handling rare events problem. (7) Safety and ethical issues.	(1) Bi-directional encoders [72]. (2) Model compression techniques [75,76]. (3) Introduction of external knowledge [48]. (4) Dealing with input bias [77]. (5) Improving consistency and controllability [78]. (6) Data augmentation and domain-specific training [20]. (7) Safety and ethical measures [79].
GPT3 [23]	GPT-3 is based on a large-scale pre-trained language model, pre-trained by the Transformer architecture and massive text data. It is capable of generating coherent text with broad adaptability and flexibility, and it can demonstrate powerful language-processing capabilities on a variety of tasks.	Pre-training utilizes self-supervised learning to gain linguistic knowledge and patterns. Fine-tuning optimizes the model for specific tasks with labeled data.	(1) The problem of the high cost of training and reasoning. (2) Problem of lack of common sense and reasoning skills. (3) The problem of sensitivity to input bias. (4) Difficulty in ensuring consistency of generated content.	(1) Model size and efficiency optimization [80]. (2) External knowledge introduction [81–83], data augmentation [63], and domain-specific training. (3) Improved interpretability [84] and control, social and ethical norms [85]. (4) Providing more contextual information, introducing additional constraints, tuning model parameters, incorporating human review or post-editing, and using larger training datasets.
GPT4 [24]	A Transformer-based multimodal pre-trained language model is used to predict the next token in a document.	GPT-4 is a generative pre-trained model that uses an autoregressive Transformer architecture and uses more parameters and hierarchies in training to enhance its representation. It also utilizes larger datasets for pre-training to improve its understanding of different domains and contexts.	(1) Problems of illusory facts and faulty reasoning. (2) Problem of lack of understanding of updated data. (3) Problems of confident error and bias presence.	(1) Increasing the amount and diversity of training data [86], manual intervention for correction [87], continuous updating of the model e.g., reference to the BARD model to incorporate new knowledge and update the structure [88] and enhancement of the model's interpretability [89]. (2) Adding the latest training data, optimizing the design of the GPT-4 algorithm e.g., ChatABL method [90], enhancing the quality control of the input data [91], and developing better evaluation metrics [92]. (3) Checking the quality of input data [93], increasing the dataset [94] and the number of training sessions, and using integrated learning techniques [58].

Table 7. *Cont.*

Models	Key Ideas	Approaches	Key Problems	Solutions
PANGU-Σ [25]	Improving the Performance of Natural Language Processing Tasks Using Large-Scale Autoregressive Language Models Obtained by Pre-Training on Massive Text Corpora.	An autoregressive approach is used, containing stacked Transformer decoding and query layers. The bottom M layers are shared across all domains, while the top N layers are sparsely activated based on the domain of the input data. Each RRE layer consists of K experts in a G-group to provide a mixed, dense, and sparse mode selection.	(1) The problem of constructing a user-friendly sparse architecture system. (2) The problem of developing a language model for obtaining accurate feedback from open environments. (3) The problem of effectively combining language models with multimodal perceptual inputs. (4) Overcoming the cost of deploying large language models. (5) The problem of effectively storing and updating the knowledge required for large language models online.	(1) Designing optimized algorithms and system architectures. (2) Improve data-labeling methods and self-supervised learning techniques. (3) Design multimodal fusion network architecture. (4) Using model compression and optimization techniques. (5) Designing efficient storage systems and incremental training algorithms.

BERT [17], proposed by Devlin J et al., uses a bi-directional encoder and self-attention mechanism to capture deep semantic and contextual information of a text, which are pre-trained and fine-tuned to achieve efficient applications. However, there are problems such as a high consumption of computational resources and insufficient learning. Optimizing the algorithm and model structure, increasing the amount of data, and exploring new text-processing techniques can improve the performance and generalization of BERT. SpanBERT [18], proposed by Joshi M et al., improves the text representation by masking successive text segments and predicting the content, but it suffers from the problems of context-dependence and the stochasticity of the masking strategy. It is suggested to introduce an adaptive masking mechanism and balance multiple strategies, combining stable training and unsupervised pre-training tasks to improve the model's stability and generalization ability. XLNet [19], proposed by Yang Z et al., improves text quality and accuracy by introducing causal and bi-directional contextual modeling and aligned language modeling, and although there are problems such as computational complexity, these can be solved by techniques such as an approximate attention mechanism to improve the model efficiency and performance. Liu Y et al. presented the study of Roberta [20] and pointed out that the hyperparameters and the amount of training data in BERT's pre-training are crucial to the performance, and by replicating and adjusting the pre-training process, it was found that the performance of BERT could rival that of the subsequent models. However, there are problems such as an incomplete hyperparameter search and computational resource limitation. It is suggested to expand the search scope, use distributed computing, and combine advanced models and techniques to optimize its performance.

The GPT [21] model overcomes BERT's limitations through an autoregressive structure and expanded vocabulary to generate coherent text sequences, which are applicable to a variety of NLP tasks. It faces the problems of data bias, dialog consistency, and long text processing, but these can be solved by diversifying the data, introducing a dialog history, and segmentation. GPT [21] possesses the advantages of text generation and still faces the challenges of long text processing and training complexity.

The GPT [21] model proposed by Radford A et al. is widely used and provides high-quality and adaptable text generation, but the results are inaccurate and inconsistent. It performs poorly on rare data. GPT2 [22] improves the text quality and diversity by increasing the model size and training data but still faces challenges. It can be improved by a bi-directional encoder, model compression, etc., and addressing concerns about security

and ethical issues. GPT2 [22], proposed by Radford A et al., enhances the quality and diversity of text generation but suffers from a lack of accuracy and consistency. GPT3 [23], proposed by Brown T et al., further enhances its generation capability by increasing the size and training data, and it supports multi-tasking. However, it faces challenges such as a high cost and a lack of common sense and reasoning ability. An optimization of the model and attention to ethical norms are needed. GPT4 [24], proposed by Achiam J et al., demonstrates high intelligence, handles complex problems, and accepts multimodal inputs, but suffers from problems such as hallucinations and inference errors. It can be improved by adding more data, human intervention, and model updating. Meanwhile, optimizing its algorithms and quality control inputs can improve its accuracy and reliability; checking the data quality, increasing the dataset and number of training sessions, and using integrated learning techniques can also reduce the error rate and bias. The PANGU-Σ [25] model proposed by Ren X et al. adopts a scalable architecture and a distributed training system, and utilizes massive corpus in pre-training to improve the performance of NLP, but it faces challenges such as sparse architecture design, accurate feedback acquisition, and other challenges. These can be solved by optimizing its algorithms, improving data annotation and self-supervised learning, designing multimodal fusion networks, model compression, and optimization.

5.7. A Matching Method Based on a Large Language Model Utilizing Different Features

Typical models of matching methods based on large language models utilizing different features include the R2-Net [26], ASPIRE [27], DC-Match [28], and MCP-SM [29] models, as shown in Figure 13. This study outlines the key ideas, methods, key issues, and solutions for their fine-grained matching approaches, as shown in Table 8.

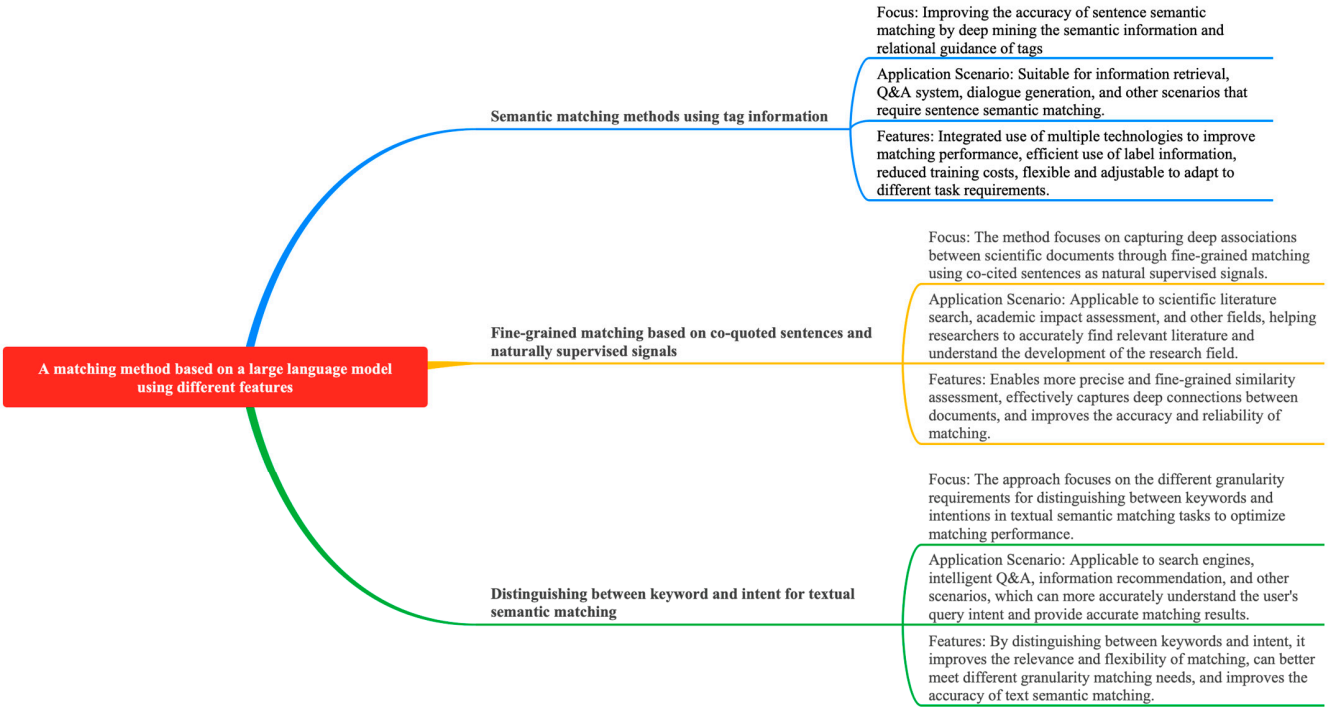

Figure 13. Overview of the logical relationships of fine-grained matching methods.

Table 8. Summary of key ideas, methods, key problems, and solutions for fine-grained matching methods.

Models	Key Ideas	Approaches	Key Problems	Solutions
R2-Net [26]	The accuracy of sentence semantic matching is improved by deep mining the semantic information and relational guidance of tags.	First, BERT and CNN encoders are combined to capture the semantic information of sentences from the global and local perspectives, respectively; second, a self-supervised relation classification task is introduced to guide the matching process using the semantic relations between labels; and finally, a ternary loss function is used to refine relation distinctions and improve the model performance.	The consumption of computational resources is high, and the speed is limited when dealing with large-scale datasets; the quality of labels affects the performance of the model, and the performance may be degraded in the presence of noisy or inaccurate labels; and the hyperparameter tuning and optimization is also a major challenge, as it needs to be carefully adjusted for specific tasks.	For the model complexity problem, model compression and distributed computing can be used to improve efficiency; for the label quality problem, semi-supervised learning or migration learning can be explored to make up for deficiencies; and the manual workload can be reduced by utilizing automatic hyperparameter-tuning tools. The combined application of these solutions can improve the model's performance and practicality.
ASPIRE [27]	This study proposes a method to accurately assess the similarity between scientific documents by matching fine-grained aspects using co-cited sentences in papers as natural supervised signals. This approach aims to capture the deeper associations between documents for a more precise and nuanced similarity assessment.	The ASPIRE method is based on sentence context representation and aggregation techniques to efficiently assess inter-document similarity by capturing fine-grained matches at the sentence level through multi-vector modeling.	(1) Data sparsity and noise. (2) Computational complexity.	(1) Data enhancement [63] and filtering. (2) Model Optimization and Acceleration Simplifying structure, dimensionality reduction, and combining parallel and distributed computing [61] to improve the efficiency of processing large-scale corpora.
DC-Match [28]	This study addresses the need for matching at different granularities in the text semantic matching task, and it proposes to optimize the matching performance and improve the matching accuracy by distinguishing between keywords and intent and adopting a partitioning strategy.	The proposed DC-Match training strategy contains three objectives: global matching model classification, remote supervised classification to distinguish between keywords and intent, and the special training of partitioning ideas to ensure that the distribution of global matches is similar to the distribution of combinatorial solutions distinguishing between keywords and intent, to achieve more accurate matches.	(1) The accuracy of the remotely supervised classification loss is affected by the quality of external resources, which may lead to misleading keyword extraction. (2) The special training objective of the partitioning idea may increase the computational resources and model complexity, and prolong the training time. (3) DC-Match methods may not be flexible enough and may need to be adapted for specific scenarios.	(1) Optimize the selection of external resources and quality control, and use advanced remote supervision technology to improve the accuracy of supervision information. (2) Optimize the algorithm and model structure to reduce the amount of computation, and use distributed or parallel computing to accelerate the training process [25]. (3) Introduce domain adaptive or migration learning techniques [68] to improve the flexibility and adaptability of DC-Match methods in different domains and types of text-matching tasks.
MCP-SM [29]	Proposes a multi-concept parsing approach to optimize multilingual semantic matching and reduce the dependence on traditional named entity recognition.	The study proposes an MCP-SM framework based on a pre-trained language model that parses text into multiple concepts and incorporates categorical tokens to enhance the model's semantic representation capability and achieve flexible multilingual semantic matching.	(1) Uncertainty in concept extraction. (2) Differences in language and domain. (3) Limitations of computational resources.	(1) Optimize the concept extraction algorithm. (2) Introducing multi-language and domain adaptation mechanisms. (3) Optimize the model structure and computation process.

Zhang K et al. [26] proposed to improve sentence semantic matching accuracy by deep mining label semantic information and relational guidance, combining global and local coding and self-supervised relational classification tasks to achieve significant results, but there are problems of model complexity, label quality, and hyperparameter tuning. Model

compression, distributed computing, semi-supervised learning or migration learning, and automatic hyperparameter-tuning tools are suggested to solve these problems and improve the model's performance and utility. Mysore S et al. [27] studied to achieve a fine-grained similarity assessment among scientific documents by utilizing paper co-citation sentences as a naturally supervised signal. The ASPIRE [27] method constructs multi-vector models to capture fine-grained matching at the sentence level but faces data sparsity, noise, and computational complexity issues. Data enhancement and filtering techniques can be used to improve its generalization ability, optimize the model structure, and reduce the complexity by using parallel computation to deal with a large-scale scientific corpus efficiently. Zou Y et al. study proposed a DC-Match [28] training strategy for different granularity requirements of query statements in text semantic matching tasks, which optimizes the matching performance by distinguishing between keywords and intent. However, DC-Match may face remote supervision information accuracy, computational complexity, and flexibility issues. It is suggested to optimize the resource selection and quality, adopt advanced remote supervision techniques, optimize the algorithm and model structure, utilize distributed computing techniques, and introduce domain-adaptive techniques to improve performance and adaptability. Yao D et al. propose a semantic matching method based on multi-concept parsing to reduce the dependence on traditional named entity recognition and improve the flexibility and accuracy of multilingual semantic matching. The method constructs a multi-concept parsing semantic matching framework (MCP-SM [29]), but faces concept extraction uncertainty, language and domain differences, and computational resource limitations. It is suggested to optimize the concept extraction algorithm, introduce multi-language and domain adaptation mechanisms, and optimize the model structure and computational process to improve the performance and applicability.

This study also compares the number of parameters, hardware equipment, training time, and context learning of recursive neural network matching methods, sentence vector matching methods based on large language models, and large language model-based matching methods utilizing different features, as shown in Table 9.

Table 9. Comparison of hardware configurations and training details for matching models.

Models	Parameters	Hardware Specifications	Training Duration	Context Learning
DBDIN [55]	7.8 M	NVIDIA GTX1080 GPU card (NVIDIA, Santa Clara, CA, USA)	-	Yes
MKPM [11]	1.8 M	-	-	Yes
Enhanced-RCNN [57]	7.7 M	Nvidia P100 GPU (NVIDIA, Santa Clara, CA, USA)	-	Yes
DRCN [13]	6.7 M	-	-	Yes
DC-Match [28]	-	RTX 3090 GPU	-	Yes
MCP-SM [29]	-	A100 GPUs	-	Yes
R2-Net [26]	-	Nvidia Tesla V100-SXM2-32 GB GPUs (NVIDIA, Santa Clara, CA, USA)	-	Yes
BERT [17]	340 M	Nvidia A100, V100 (NVIDIA, Santa Clara, CA, USA)	Depends on model parameter scale	Yes
SpanBERT [18]	-	V100 GPU	15 days	Yes
GPT [21]	125 M	-	-	Yes
XLNet [19]	340 M	512 TPU v3	5.5 days	Yes
Roberta [20]	355 M	V100 GPU	about 1 day	Yes
GPT2 [22]	1.5 B	-	-	Yes
GPT3 [23]	175 B	V100 GPU	-	Yes
GPT4 [24]	1.8 Trillion	-	-	Yes
PANGU-Σ [25]	1.085 T	Ascend 910 NPU (Huawei, Shenzhen, China)	about 100 days	Yes

In terms of the number of model parameters, there is an overall increasing trend from recursive neural network-based matching methods to interactive matching methods, to sentence vector matching methods based on large language models. Specifically, the MKPM model based on the recurrent neural network has the least number of parameters, which is

1.8 M, while the number of parameters of GPT4 is the largest, reaching an amazing 1.8 T. In terms of hardware requirements, as the performance of the matching model gradually increases, the requirements for the performance of the graphics card show a corresponding trend of enhancement. This suggests that to effectively train these high-performance matching models, we need to equip more advanced graphics card devices. Meanwhile, in terms of training time, the training time required for sentence vector matching methods based on large language models is also increasing. Notably, these matching models are designed based on contextual learning, which helps them to better understand and process complex semantic information.

This paper delves into the performance comparison of different matching models on different datasets, aiming to provide researchers with a comprehensive and detailed analysis perspective. Through comparative experiments and data analysis, different matching models do have significant differences in performance, as shown in Table 10.

Table 10. Performance comparison of matching models in different tasks.

Models	Accuracy/MAP/MRR/r/ρ/MSE/F1/GLUE	Task
MaLSTM [9]	Pearson correlation (r), Spearman's ρ and MSE are 0.8822, 0.8345, and 0.2286, respectively. The accuracy of MaLSTM features + SVM is 84.2%.	The Pearson correlation (r), Spearman's ρ, and MSE of MaLSTM on SICK are 0.8822, 0.8345, and 0.2286, respectively. The accuracy of MaLSTM features combined with SVM is 84.2%.
DBDIN [55]	The accuracy rates are 88.8%, 86.8%, and 89.03% respectively.	The accuracy rates of DBDIN on SNLI, SciTail, and QQP are 88.8%, 86.8%, and 89.03%, respectively.
MKPM [11]	The accuracy rates are 86.71%, 84.11%, 89.66%, and 88.2% respectively.	The accuracy rates of MKPM on LCQMC, BQ, QQP, and SNLI are 86.71%, 84.11%, 89.66%, and 88.2%, respectively.
Enhanced-RCNN [57]	The accuracies (Acc) are 89.3%, 75.51%, 90.28%, and 90.35%, respectively, while the F1 scores are 89.47%, 73.39%, 76.85%, and 74.20%, respectively.	The accuracy (Acc) of Enhanced-RCNN on QQP and Ant Financial are 89.3% and 75.51%, respectively, with F1 scores of 89.47% and 73.39% respectively. The accuracy (Acc) of Enhanced-RCNN (ensemble) on QQP and Ant Financial are 90.28% and 90.35%, respectively, with F1 scores of 76.85% and 74.20%, respectively.
DRCN [13]	The accuracy rates are 86.5%, 90.1%, 80.6%, 79.5%, 82.3%, 81.4%, and 91.3%, respectively.	DRCN (-Attn, -Flag) achieved an accuracy of 86.5% on SNLI, while DRCN achieved 90.1%. On MultiNLI, DRCN and DRCN + ELMo* achieved accuracies of 80.6% and 79.5%, and 82.3% and 81.4%, on matched and mismatched datasets, respectively. DRCN achieved an accuracy of 91.3% on QQP.
	The MAP values are 0.804, 0.830, and 0.925, while the MRR values are 0.862, 0.908, and 0.930, respectively.	DRCN achieved MAP and MRR of 0.804 and 0.830, respectively, on TrecQA: raw, and 0.862 and 0.908, respectively, on TrecQA: clean. On SelQA, DRCN achieved MAP and MRR of 0.925 and 0.930. respectively.
BiMPM [15]	The accuracy is 88.17% and 88.8% respectively. The MAP values are 0.802 and 0.875, while the MRR values are 0.718, and 0.731, respectively.	The accuracy of BiMPM on QQP and SNLI is 88.17% and 88.8%, respectively. On the TREC-QA and WikiQA datasets, the MAP values are 0.802 and 0.875, while the MRR values are 0.718 and 0.731, respectively.
BERT [17]	The GLUE scores are 84.6/83.4, 71.2, 90.5, 93.5, 52.1, 85.8, 88.9, 66.4, 86.7/85.9, 72.1, 92.7, 94.9, 60.5, 86.5, 89.3, 70.1, and 82.1, respectively.	The GLUE scores of BERT-base on MNLI (-m/mm), QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE are 84.6/83.4, 71.2, 90.5, 93.5, 52.1, 85.8, 88.9, and 66.4 respectively. While the GLUE scores of BERT-large on MNLI (-m/mm), QQP, QNLI, SST-2, CoLA, STS-B, MRPC, and RTE are 86.7/85.9, 72.1, 92.7, 94.9, 60.5, 86.5, 89.3, and 70.1, respectively.
SpanBERT [18]	The GLUE scores are 64.3, 94.8, 90.9/87.9, 89.9/89.1, 71.9/89.5, 88.1/87.7, 94.3, and 79.0, respectively.	The GLUE scores of SpanBERT on CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, and RTE are 64.3, 94.8, 90.9/87.9, 89.9/89.1, 71.9/89.5, 88.1/87.7, 94.3, and 79.0 respectively.
XLNet [19]	The GLUE scores are 77.4, 88.4, 93.9, 91.8, 81.2, 94.4, 90.0, 65.2, and 91.1, respectively.	The GLUE scores of XLNet-Large wikibooks on RACE, MNLI, QNLI, QQP, RTE, SST-2, MRPC, CoLA, and STS-B are 77.4, 88.4, 93.9, 91.8, 81.2, 94.4, 90.0, 65.2, and 91.1, respectively.
Roberta [20]	The GLUE scores are 90.8/90.2, 98.9, 90.2, 88.2, 92.3, and 67.8, respectively.	RoBERTa's GLUE scores on MNLI, QNLI, QQP, RTE, MRPC, and CoLA are 90.8/90.2, 98.9, 90.2, 88.2, 92.3, and 67.8, respectively.

Table 10. *Cont.*

Models	Accuracy/MAP/MRR/r/ρ/MSE/F1/GLUE	Task
GPT [21]	The accuracy rates are 82.1, 81.4, 89.9, 88.3, 88.1, and 56.0, respectively. The MC, ACC, F1, PC, and F1 scores are 45.4, 91.3, 82.3, 70.3, and 82.0, respectively.	GPT achieves accuracy rates of 82.1/81.4, 89.9, 88.3, 88.1, and 56.0 on MNLI (-m/mm), SNLI, QNLI, SciTail, QNLI, and RTE, respectively. On CoLA, SST2, MRPC, QQP, and STSB, GPT's scores are 45.4 for MC, 91.3 for ACC, 82.3 for F1, 70.3 for PC, and 82.0 for F1.
GPT3 [23]	The accuracy rate is 69%.	The accuracy rate of GPT-3 Few-Shot on RTE is 69%.
PANGU-Σ [25]	The accuracy rates are 51.14%, 45.97%, 68.49%, and 56.93%, respectively.	The accuracy rates of PANGU-Σ on CMNLI, OCNLI, AFQMC, and CSL are 51.14%, 45.97%, 68.49%, and 56.93%, respectively.
GMN [60]	The accuracy rates are 84.2% and 84.6%, and the F1 scores are 84.1% and 86%, respectively.	The GMN model achieves accuracy rates of 84.2% and 84.6% on LCQMC and BQ, respectively, with F1 scores of 84.1% and 86%.
DRr- Net [14]	The accuracy rates are 87.7%, 71.4%, 76.5%, 88.3%, and 89.75%, respectively.	The accuracy rates of DRr-Net on different SNLI datasets are 87.7%, 71.4%, and 76.5%, respectively, while the accuracy rates on the SICK and QQP datasets are 88.3% and 89.75%, respectively.
DC-Match [28]	The accuracy rates are 91.7%, 92.2%, 88.1%, 88.9%, 73.73%, and 74.22%, respectively, while the Macro-F1 scores are 72.96% and 73.67%.	DC-Match (RoBERTa-base) and DC-Match (RoBERTa-large) achieve accuracy rates of 91.7% and 92.2% on QQP, 88.1% and 88.9% on MRPC, and 73.73% and 74.22% on Medical-SM, respectively. The Macro-F1 scores for DC-Match (RoBERTa-base) and DC-Match (RoBERTa-large) are 72.96% and 73.67%, respectively.
MCP-SM [29]	The accuracy rates are 91.8%, 92.4%, 88.9%, 89.7%, 73.90%, 74.82%, 76.59%, 76.05%, 71.93%, and 72.49%, respectively. The Macro-F1 scores are 73.31%, 73.90%, 76.59%, 76.05%, 71.98%, and 72.42%, respectively.	MCP-SM (DEBERTa-base) and MCP-SM (DEBERTa-large) achieve accuracy rates of 91.8% and 92.4% on QQP, and 88.9% and 89.7% on MRPC, respectively. MCP-SM (MacBERT-base) and MCP-SM (MacBERT-large) achieve accuracy rates of 73.90% and 74.82% on Medical-SM, with Macro-F1 scores of 73.31% and 73.90%, respectively. MCP-SM (ARBERT) achieves an accuracy rate of 76.59% and a Macro-F1 score of 76.05% on the MQ2Q dataset. MCP-SM (AraBERT-base) and MCP-SM (AraBERT-large) achieve accuracy rates of 71.93% and 72.49% on XNLI, with Macro-F1 scores of 71.98% and 72.42%, respectively.
R2-Net [26]	The accuracy rates are 90.3%, 81%, 89.2%, 92.9%, 91.6%, and 84.3%, respectively.	The accuracy rates of R2-Net on the Full test and Hard test of the SNLI dataset are 90.3% and 81%, respectively, 89.2% on SICK, 92.9% on SciTail, and 91.6% and 84.3% on Quora and MSRP, respectively.

The Pearson correlation (r), Spearman's ρ, and MSE of MaLSTM on SICK were 0.8822, 0.8345, and 0.2286. The Acc of MaLSTM features + SVM was 84.2%. The GMN model based on the graph neural network had an accuracy of 84.2% and 84.6% on LCQMC and BQ, with F1 values of 84.1% and 86%. In the QQP dataset, we witness a gradual improvement in the accuracy of multiple matching methods. Initially, the recursive neural network-based matching method increases the accuracy to 89.03%, and then the interactive matching method further pushes this figure to 91.3%. Then, using the labeling information of the sentence pairs to be matched, the accuracy of the matching method climbed again to 91.6%. However, the fine-grained matching method, with its unique advantages, boosted the accuracy to a maximum of 91.8%, highlighting its sophistication in its current state. At the same time, the large language model shows performance growth on multiple datasets. Whether in MNLI, QQP, QNLI, COLA, or MRPC and RTE tasks, the big language model has grown in terms of GLUE scores, which fully proves the continuous improvement in the big language model's performance. Of note is the performance of the GPT family of models on the dataset RTE. With the evolution from GPT to GPT3, the accuracy of the models also grows steadily, which fully demonstrates the continuous enhancement of the performance of the GPT series of models. In addition, the PANGU-sigma model shows a strong performance in several tasks. In CMNLI, OCNLI, AFQMC, and CSL tasks, it achieved accuracy rates of 51.14%, 45.97%, 68.49%, and 56.93%, respectively, which further validates its ability to handle complex tasks. In the fine-grained matching approach, our accuracy rate in the MRPC dataset is also further improved by introducing different large language models as skeletons. Specifically, the accuracies increased from 88.1% and 88.9%

to 88.9% and 89.7%, and this enhancement again proves the effectiveness of the fine-grained matching approach.

This paper also outlines the APIs for the matching model, and different APIs have their advantages and disadvantages in terms of performance, ease of use, and scalability, and the selection of a suitable API (Application Programming Interface) depends on the specific application scenario and requirements, as shown in Table 11. These NLP APIs have their characteristics in several aspects, and by comparing them, we can draw the following similarities and differences. The similarity is that these APIs are launched by tech giants or specialized NLP technology providers with strong technical and R&D support capabilities. For example, Google Cloud Natural Language API, IBM Watson Natural Language Understanding API, Microsoft Azure Text Analytics API, and Amazon Comprehend API are the respective company's flagship products in the NLP field, all of which reflect their deep technical accumulation and R&D capabilities. In addition, most of the APIs support multi-language processing, covering major languages such as English and Chinese to meet the needs of different users. Whether it is a sentiment analysis of English text or an entity recognition of Chinese text, these APIs can provide effective support. In terms of interface design, these APIs are usually designed with RESTful interfaces, enabling developers to integrate and invoke them easily. This standardized interface design reduces the learning cost of developers and improves development efficiency. In terms of application scenarios, these APIs are suitable for NLP tasks such as text analysis, sentiment analysis, entity recognition, etc., and can be applied to multiple industries and scenarios. Whether it is the analysis of product reviews on e-commerce platforms or the monitoring of public opinion in news media, these APIs can provide strong support.

Table 11. Comparison of matching model APIs.

API	API-Providing Companies	Supported Languages	Interface Type	Application Scenario	Dominance	Drawbacks
Google Cloud Natural Language API	Google	polyglot	RESTful API interface	Suitable for search engine optimization, information retrieval, sentiment analysis, and many other scenarios	(1) Leading technology, based on Google's powerful AI and NLP technology. (2) Provides rich natural language processing functions. (3) High accuracy and reliability.	(1) May be subject to Google's terms of service and geographic restrictions. (2) May be expensive to use, especially for large-scale projects.
IBM Watson Natural Language Understanding API	IBM	polyglot	RESTful API interface	It is suitable for Q&A system, intelligent customer service, sentiment analysis, and other fields.	(1) Combined with Watson's cognitive computing capabilities, it provides in-depth text understanding and matching. (2) Powerful customization capabilities that can be configured according to specific needs.	(1) May require higher technical investment and learning costs. (2) IBM services and fees may be more expensive.
Microsoft Azure Text Analytics API	Microsoft	polyglot	RESTful API interface	It is suitable for scenarios such as text mining, theme analysis, and sentiment tendency judgement.	(1) Seamlessly integrates with other services on the Azure platform. (2) Provides efficient and stable text-processing capabilities.	(1) May be limited by the ecosystem and fee structure of the Azure platform. (2) Integration may be cumbersome for non-Azure users.

Table 11. *Cont.*

API	API-Providing Companies	Supported Languages	Interface Type	Application Scenario	Dominance	Drawbacks
Amazon Comprehend API	Amazon	polyglot	RESTful API interface	It is suitable for scenarios such as content analysis, topic detection, and sentiment analysis.	(1) Easy to integrate with other AWS services as part of AWS. (2) Provides efficient and scalable text-processing capabilities.	(1) May be subject to AWS terms of service and geographic coverage. (2) Additional integration work may be required for non-AWS users.
Rasa NLU	Rasa	polyglot	RESTful API interface	It is widely used in customer service, online helpdesks, education platforms, e-commerce, and intelligent assistant and other scenarios, aiming to enhance user engagement and satisfaction.	(1) Provides flexible and powerful natural language understanding functions, including user intent recognition, entity extraction and parameter optimization. (2) Supports pre-loaded training models, providing developers with a convenient way to build dialogue-driven automated intelligent customer service assistants. (3) Active community and leading technology, tracking the most cutting-edge technology and applying it to the system.	(1) For beginners, some learning cost may be required to familiarize and master its usage and API interface. (2) Additional customization and optimization work may be required when dealing with complex or domain-specific natural language understanding tasks.
Hugging Face Transformers	Hugging Face	polyglot	Model libraries and tool sets	The Hugging Face Transformers library is widely used for tasks related to natural language processing, such as text categorization, sentiment analysis, language translation, and so on. It is particularly popular with developers, providing them with easy-to-use and flexible interfaces to handle NLP tasks.	(1) Provides a range of high-quality pre-trained models, enabling developers to rapidly build and deploy NLP applications. (2) Has an active community and developer ecosystem that provides a wealth of resources and support. (3) High ease of use and flexibility, adapting to a variety of NLP tasks and scenarios.	(1) For complex NLP tasks, in-depth NLP knowledge and experience may be required for model selection and tuning. (2) In some specific domains or scenarios, additional data labeling and model training work may be required.
Semantic Scholar API	Semantic Scholar	polyglot	RESTful API interface	Focusing on text matching and searching in the academic field, it is suitable for researchers, scholars, and academic institutions to conduct literature searches, citation analysis, research trend tracking, and so on.	(1) Provides high-quality academic text-matching results that can accurately identify relevant academic literature and research content. (2) Integrates a wealth of academic resources covering a wide range of subject areas. (3) Provides a flexible API interface, which is convenient for users to integrate into their own systems or applications.	(1) Primarily for academic fields, with limited applicability to other non-academic fields. (2) May require paid subscription or purchase of specific services to enjoy full functionality. (3) May require higher computational resources and time costs when dealing with large-scale or complex academic text-matching tasks.

Table 11. *Cont.*

API	API-Providing Companies	Supported Languages	Interface Type	Application Scenario	Dominance	Drawbacks
TexSmart Text Matching API	TexSmart	polyglot	HTTP Online Service Interface	This API supports text content matching, accessed via HTTP-POST, providing efficient text matching.	(1) High accuracy to ensure precise text matching. (2) Provides flexible API interfaces and configuration options to meet specific needs. (3) Fast response time, suitable for real-time or high concurrency scenarios.	(1) Relatively low visibility and small user base. (2) May lack thorough documentation and community support.
Baidu AI Text Similarity API	Baidu	polyglot	RESTful API interface	An API provided by Baidu for calculating the similarity of two texts, supporting multiple languages and domains.	(1) Strong Chinese-processing capability, especially good at processing Chinese text. (2) High stability and reliability to ensure service quality (3) Provides rich API interface and SDK for easy integration.	(1) May be subject to geographical restrictions, and foreign use may be limited or delayed. (2) Further customized development may be required for specific needs.
AliCloud NLP Text Similarity API	Alibaba	polyglot	API Interface	AliCloud's natural language processing service also includes an API for text similarity matching.	(1) Strong technical strength and support for high scalability and flexibility. (2) Synergizes with other AliCloud services for easy integration and expansion. (3) Provides powerful data-processing and analysis capabilities.	The cost of use may be high and require some investment of resources.
Jingdong AI Text Matching API	JD.com	polyglot	API Interface	The Jingdong AI platform provides text-matching services for a variety of text-matching needs.	(1) Applicable to text-matching needs in the field of e-commerce, with industry advantages. (2) Rich in data resources, providing accurate matching potential. (3) Can help merchants achieve accurate marketing and promotion.	(1) There may be domain limitations that apply to specific scenarios. (2) Openness and usability may be limited and need to be evaluated.
Tencent Cloud NLP Text Matching API	Tencent	polyglot	RESTful API interfaces and SDKs	Tencent Cloud's natural language processing service includes a text-matching function that can support a variety of application scenarios.	(1) Leading technology, providing advanced NLP technology and algorithms. (2) Supports multi-language processing to meet global needs. (3) Provides customized solutions to meet specific industry and scenario needs.	(1) Cost of use may be high and budgetary constraints need to be considered. (2) Integration into existing systems may present some technical challenges.

However, there are some differences in these APIs. First, API-providing companies may differ in their focus and expertise in NLP. For example, the Google Cloud Natural Language API excels in syntactic analysis and semantic understanding, while IBM Watson is known for its powerful natural language generation and dialog system. Second, while both support multilingual processing, the specific types of languages supported, and the

level of coverage may vary. Some APIs may be optimized for specific languages or regions, to better suit the local culture and language habits. In terms of interface types, while most APIs utilize RESTful interfaces, some APIs may also provide other types of interfaces, such as SDKs or command-line tools. For example, Hugging Face Transformers provides a rich SDK and model library, enabling developers to train and deploy models more flexibly. Each API may also have a focus in terms of application scenarios. For example, some APIs may specialize in sentiment analysis, which can accurately identify emotional tendencies in a text, while others may focus more on entity recognition or text classification tasks, which can accurately extract key information in a text. In the marketplace, certain APIs are more advantageous due to their brand recognition, technological prowess, or marketing strategies. However, this advantage may change as technology evolves and the market changes. Therefore, developers need to consider several factors when choosing an NLP API, including technical requirements, budget constraints, data privacy, and security. In addition, each API may have some drawbacks or limitations. For example, some APIs may face performance bottlenecks and be unable to handle large-scale text data; or the accuracy of some APIs may be affected by specific domain or language characteristics. Therefore, developers need to fully understand the advantages and disadvantages of NLP APIs when choosing them, and combine them with their actual needs.

6. Comparison of Different Matching Models across Datasets and Evaluation Methods

This section focuses on outlining the performance of different matching models on different datasets as shown in Table 12. From the perspective of datasets, English datasets are widely used in matching models, while Chinese datasets are relatively few, mainly including OCNLI, LCQMC, BQ, Medical-SM, and Ant Financial. From the perspective of modeling, recursive neural network-based matching methods are mainly applied to natural language inference and implication (NLI) datasets such as SNLI and SciTail, and Q&A and matching datasets such as QQP, LCQM, and BQ. Interactive matching methods are mainly focused on NLI datasets such as SNLI and Q&A and matching datasets such as Trec-QA and WikiQA. Graph neural network matching methods are mainly applied to Q&A and matching datasets such as LCQMC and BQ, while fine-grained matching methods are widely applied to NLI datasets such as XNLI, Q&A, and matching datasets such as QQP, MQ2Q, etc., as well as other specific datasets such as MRPC and Medcial-SM. Semantic matching methods utilizing label information are mainly applied to NLI datasets such as SNLI and SciTech, SNLI and SciTail, and Q&A and matching datasets such as QQP. As for the matching methods based on large language models, their application scope is even wider, covering NLI datasets such as MNLI and QNLI, Q&A and matching datasets such as QQP, the sentiment analysis dataset SST-2, and other specific datasets such as MRPC and COLA. To summarize, different matching models are applied to all kinds of datasets, but the English dataset is more commonly used. The models perform well on specific datasets and provide strong support for the development of the natural language processing field.

From the perspective of evaluation metrics, when discussing the evaluation metrics for the sentence-matching task in Table 12, this study needs to first clarify the definition of each metric, as well as their applicability and effectiveness in different scenarios. First, accuracy (Acc) is a basic categorization evaluation metric, which is defined as the proportion of samples correctly predicted by the model to the total number of samples. In the sentence-matching task, when the distribution of positive and negative samples is more balanced, the accuracy rate can intuitively reflect the performance of the model. However, when faced with category imbalance, the accuracy rate may not accurately assess the effectiveness of the model.

For the regression prediction task of sentence similarity scores, the Pearson correlation (r) and Spearman's ρ are two important evaluation metrics. The Pearson correlation (r) measures the strength and direction of a linear relationship between two continuous variables, while Spearman's ρ measures the strength and direction of a monotonic relationship between two variables, which is less demanding on the distributional pattern of the data.

These two metrics are useful in assessing the relationship between predicted and true scores. The Mean Square Error (MSE) is another commonly used metric for regression assessment, which calculates the mean of the squares of the differences between the predicted and true values. In the regression task of sentence similarity scoring, the MSE can directly reflect the accuracy of the model's predicted scores. For the binary classification task of sentence matching, especially when the distribution of positive and negative samples is not balanced, the F1 score is a more accurate evaluation metric. It combines precision and recall and comprehensively evaluates the model's performance by calculating the reconciled average of the two.

When the sentence-matching task involves ranking multiple queries or sentence pairs, the Mean Average Precision (MAP) and Mean Reverse Ranking (MRR) are two important evaluation metrics. MAP is used to evaluate the average precision of multiple queries, while MRR computes the average of the reverse of the first relevant result ranking across all queries. These two metrics are effective in evaluating the performance of the model in ranking results. The Macro-F1 is an evaluation metric for multi-category sentence-matching tassk. When calculating the F1 score, it calculates the F1 score for each category separately and then averages the scores, regardless of the sample size of the categories. This allows Macro-F1 to treat each category equally and is suitable for sentence-matching tasks where the performances of multiple categories need to be considered together. GLUE (General Language Understanding Evaluation) is a benchmark evaluation suite that includes a range of natural language understanding tasks. If a sentence-matching task is included in the GLUE suite, we can use the evaluation metrics provided (e.g., accuracy, F1 score, etc.) to evaluate the performance of the model. Finally, metrics such as MC/F1/PC may be acronyms or specific metrics under a particular task or dataset. In the absence of a specific context, they may stand for accuracy (MC may be an abbreviation for Multi-Class), F1 score, or some kind of precision. Depending on the specific task and dataset, these metrics may be used to evaluate different aspects of the performance of the sentence-matching model. By using a combination of these evaluation metrics, we can gain a more comprehensive understanding of the performance of sentence-matching models and optimize the model design accordingly.

Table 12. Comparison of different matching model training datasets and evaluation methods.

Classification of Datasets	Natural Language Inference and Implication (NLI) Dataset								Q&A and Matching Datasets						Sentiment Analysis Datasets		Other Specific Data Sets									Evaluation Method
LLMs/Dataset	MNLI	QNLI	SNLI	XNLI	RTE	SciTail	CMNLI	OCNLI	TREC-QA	WikiQA	SelQA	LCQMC	BQ	QQP	MQ2Q	SST-2	MRPC	SICK	Ant Financial	MultiNLI	Medical-SM	CoLA	STS-B	AFQMC	CSL	MSPR	Evaluation algorithm
MaLSTM [9]	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	√	×	×	×	×	×	×	×	×	Acc/Pearson correlation (r)/Spearman's ρ/MSE
DBDIN [55]	×	×	√	×	×	√	×	×	×	×	×	×	×	√	×	×	×	×	×	×	×	×	×	×	×	×	Acc
MKPM [11]	×	×	√	×	×	×	×	×	×	×	×	√	√	√	×	×	×	×	×	×	×	×	×	×	×	×	Acc
Enhanced-RCNN [57]	×	×	×	×	×	×	×	×	×	×	×	×	×	√	×	×	×	×	√	×	×	×	×	×	×	×	Acc/F1
DRCN [13]	×	×	√	×	×	×	×	×	√	×	√	×	×	√		×	×	×	×	√	×	×	×	×	×	×	Acc/MAP/MRR
BiMPM [15]	×	×	√	×	×	×	×	×	√	√	×	×	×	√	×	×	×	×	×	×	×	×	×	×	×	×	Acc/MAP/MRR
GMN [60]	×	×	×	×	×	×	×	×	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	Acc/F1
DRr- Net [14]	×	×	√	×	×	×	×	×	×	×	×	×	×	√	×	×	×	√	×	×	×	×	×	×	×	×	ACC
DC-Match [28]	×	×	×	×	×	×	×	×	×	×	×	×	×	√	×	×	√	×	×	×	√	×	×	×	×	×	Acc/Macro-F1
MCP-SM [29]	×	×	×	√	×	×	×	×	×	×	×	×	×	√	√	×	√	×	×	×	√	×	×	×	×	×	Acc/Macro-F1
R2-Net [26]	×	×	√	×		√	×	×	×	×	×	×	×	√	×	×	×	√	×	×	×			×	×	√	Acc
BERT [17]	√	√	×	×	√	×	×	×	×	×	×	×	×	√	×	√	√	×	×	×	×	√	√	×	×	×	GLUE
SpanBERT [18]	√	√	×	×	√	×	×	×	×	×	×	×	×	√	×	√	√	×	×	×	×	√	√	×	×	×	GLUE
XLNet [19]	√	√	×	×	√	×	×	×	×	×	×	×	×	√	×	√	√	×	×	×	×	√	√	×	×	×	GLUE
Roberta [20]	√	√	×	×	√	×	×	×	×	×	×	×	×	√	×	×	√	×	×	×	×	√	×	×	×	×	GLUE
GPT [21]	√	√	√	×	√	√	×	×	×	×	×	×	×	√	×	√	√	×	×	×	×	√	√	×	×	×	Acc/MC/F1/PC
GPT3 [23]	×	×	×	×	√	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	Acc
PANGU-Σ [25]	×	×	×	×	×	×	√	√	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	√	√	×	Acc
Size	392 k	108 k	570,152	-	2.5 k	27,026	-	5.6 k	-	-	-	120,000	260,068	404,276	-	67 k	5801	10,000	492 k	433,000	48,008	10,657	5.7 k	-		5801	-
Source	Transcribed speech, novels, government reports	Converted from SQuAD	Flickr30 k corpus (Young et al., 2014, [95])	Developed by the Facebook AI Institute in collaboration with a team of NYU researchers	-	Multiple-choice science exams and web sentences (Khot et al., 2018, [96])	CLUE (Liang Xu, et al., 2020, [97])	Ocnli Hai Hu, et al., 2020, [98]	-	-	-	LCQMC (Liu et al., 2018, [99])	BQ (Chen et al., 2018a, [100])	Quora website (Iyer et al., 2017, [101])	-	-	Online News	8 K image Flickr dataset and STS MSR video description dataset (Marelli et al., 2014, [102])	Alibaba	Built by researchers at New York University, Princeton University, and Georgetown University.	Chinese search engine	Books and journals on language theory	-	CLUE Liang (Xu, et al., 2020, [97])	CLUE (Liang Xu, et al., 2020, [97])	Nes clusters from the web (Dolan and Brockett, 2005, [103])	-

(√ indicates the presence of this element, × indicates the absence of this element).

7. Field-Specific Applications

This section provides insights into the wide range of applications of matching methods in different domains, especially in education, healthcare, economics, industry, agriculture, and natural language processing. Through the detailed enumeration in Table 12, we can clearly see the main contributions of matching models within each domain, as well as reveal the limitations they may encounter in practical applications and potential future research directions. In the exploration of text-matching techniques, we recognize their central role in several practical scenarios, especially in the key tasks of natural language processing, such as named entity recognition and named entity disambiguation. To gain a more comprehensive understanding of the current status and trends of these techniques, we have expanded the scope of our literature review to cover research results in these areas. Named entity recognition (NER) [104] is a crucial task in natural language processing, whose goal is to accurately identify entities with specific meanings, such as names of people, places, organizations, etc., from the text. In recent years, with the continuous development of deep learning technology, the field of NER has made significant progress. Meanwhile, named entity disambiguation [105] (NED) is also an important research direction in the field of natural language processing. It aims to solve the problem that the same named entity may refer to different objects in different contexts, to improve the accuracy and efficiency of natural language processing. In summary, matching methods and text-matching techniques have a wide range of application prospects and an important research value in several fields. (See Table 13).

Table 13. Field-specific application of text matching.

Domains	Time/Author	Main Contributions	Limitation	Future Research Directions
education	(2024, Alkhidir T et al.) [106]	(1) An AI-based curriculum-planning methodology is proposed to improve planning efficiency by automatically aligning learning objectives within and outside of disciplines through semantic matching techniques. (2) A data science pipeline is designed to analyze the similarity of learning objectives and identify subject clustering and dependencies. (3) Provides a visualization tool for course similarity, which facilitates course cross-mapping analysis by experts. (4) Reveals new similarities between courses and provides a basis for optimizing course structure.	(1) Current research focuses mainly on semantic aspects and does not comprehensively consider the cognitive complexity of learning objectives. (2) Although prerequisite inferences based on student performance have been proposed, the specific method of implementation has yet to be investigated.	(1) Using AI technology to automatically extract learning objectives and keywords from textbooks and link courses across disciplines. (2) Combine learning objectives with cognitive skills to comprehensively assess the complexity of learning objectives. (3) Integrate student performance to study in depth the relationship between prerequisites and learning objectives and optimize curriculum planning. (4) Explore the application of AI technology in personalized course recommendation and adjustment.
	(2021, Hayden K A et al.) [107]	This study provides a comprehensive review of the research methodology of text-matching software in education, reveals its multiple application roles in higher education, and provides important evidence support and methodological insights for academic integrity research.	The failure of this study to provide definitive conclusions regarding the effectiveness of text-matching software and the continued lack of understanding of the limitations of the software and its available evidence demonstrates the limitations and challenges of the current study.	Future research should further explore the practical effects of text-matching software, focus on new developments in the field of academic integrity, and delve into the limitations of the software and its evidence to provide a more comprehensive reference for educational decision-making.

Table 13. *Cont.*

Domains	Time/Author	Main Contributions	Limitation	Future Research Directions
study of medicine	(2024, Jeong J et al.) [108]	(1) The X-REM method is proposed for radiology report retrieval generation using image and text matching, and its superiority is verified by quantitative analysis and manual evaluation. (2) X-REM better captures image–report interactions, with multimodal encoders and similarity metrics to enhance performance.	(1) There are still gaps in X-REM compared to reports prepared by radiologists. (2) The study is based on a single dataset, which may be biased, and the performance of datasets external to the model is not empirically validated. (3) The evaluation process, although optimized for clinical and natural language metrics, may have overlooked other important factors.	(1) Expand dataset diversity and size to validate model generalization capability. (2) Deepen the research on multimodal coding and similarity metrics to improve the accuracy of report generation. (3) Incorporate more medical knowledge and contextual information to enhance model applicability. (4) Study more influencing factors and develop more comprehensive evaluation criteria.
	(2019, Luo Y F et al.) [109]	A novel approach combining dictionary lookups and deep learning models is proposed for medical text entity-linking and shows advantages in capturing semantic similarities.	The method is limited in handling data with diverse semantics and no uniform representation, and is affected by the size of small datasets, resulting in limited performance improvement.	Data inconsistencies and ambiguities need to be resolved, more efficient methods of handling semantically diverse data need to be explored, and datasets need to be expanded to improve model performance
	(2022, Wang L et al.) [110]	This paper proposes the BMA integrated depth architecture, which combines BERT, MSBiLSTM, and a dual-attention mechanism, to solve the semantic similarity matching problem in automatic medical Q&A systems and significantly improve the performance.	BMA model may have limitations in processing specific types of medical texts, its generalization capabilities need to be strengthened, and it may require high computational resources. In addition, its performance is affected by the quality and quantity of training data.	In the future, the application of the model on more diverse and large-scale medical datasets can be explored to optimize the model architecture and improve its generalization capability and efficiency. At the same time, the combination of more natural language processing techniques with BMA can be investigated to further improve performance, and new ways to deal with non-canonical text and expressive diversity can be investigated.
economics	(2024, Ajaj S H) [111]	(1) highlights the importance of job-posting quality in recruitment and proposes the concept of an AI-optimized assistant to improve recruitment efficiency. (2) Proposes an AI assistant based on knowledge base and semantic matching aimed at providing accurate job postings and industry-specific recommendations.	Although AI optimization assistants can theoretically improve the quality of job postings, their effectiveness may be affected by a variety of factors, such as the type of job, industry characteristics, and the search habits of job seekers.	Developing and testing AI optimization assistants, studying their applicability across different industries and positions, exploring in-depth candidate search behaviors, and considering more recruitment-related factors.

Table 13. *Cont.*

Domains	Time/Author	Main Contributions	Limitation	Future Research Directions
	(2024, Ren H. et al.) [112]	(1) Fills the research gap of policy tourism in public management cooperation, and puts forward a new perspective on policy tourism for economic cooperation. (2) Using a non-parametric matching method, the study empirically analyzes the impact of policy tourism on inter-provincial trade flows in China, and find that it decreases over time and has a more significant impact on less developed jurisdictions.	(1) The study is mainly based on the Chinese context and may have geographical and cultural limitations. (2) Failure to fully control for all factors that may affect interprovincial trade flows and results may be biased. (3) Lack of in-depth description and classification of specific forms and mechanisms of policy tourism.	(1) Expand the scope of the study to other countries and regions to explore the universality and specificity of policy tourism. (2) Explore in depth the specific forms and mechanisms of policy tourism, and improve understanding of the path and extent of its impact. (3) Combine more data and methods to improve the accuracy and reliability of the impact estimates of policy tourism. (4) To study the interaction between policy tourism and other public management network efforts.
	(2012, Gopalakrishnan V et al.) [113]	An end-to-end unsupervised system is proposed for matching product titles in different formats. Highly accurate matching is achieved by enriching titles with search engine results, calculating word importance, and optimizing the number of search engine calls. Experiments verify that the system outperforms IDF's cosine similarity method in terms of F1 score and is efficient and accurate.	The current approach is limited when dealing with product titles with no distinguishing features, and an over-reliance on search engines can lead to performance fluctuations.	(1) Enhancing domain adaptation: improving algorithms to adapt to different product domains. (2) Combining multiple learning methods: fusing unsupervised and semi-supervised learning to improve performance. (3) Reduce reliance on external data sources: develop internal data sources or alternatives. (3) Fusing contextual information: utilizing more contextual information to improve matching accuracy. (4) Handling multimodal data: investigating methods to fuse information from images, videos, etc., to optimize matching.

Table 13. *Cont.*

Domains	Time/Author	Main Contributions	Limitation	Future Research Directions
	(2020, Akritidis L K et al.) [114]	UPM enables unsupervised product matching through clustering, where combinations of headline terms are constructed and scored, with the highest-scoring combination representing the product identity. Morphological analysis is performed to identify useful tokens and assign them to virtual fields. Virtual fields are scored and used to evaluate the combinations, considering the position and frequency of the combination in the headline. Once clusters are are formed, post-processing validation is performed by blocking conditions to avoid entities coexisting in the same cluster. UPM avoids pairwise comparisons between products, reduces computational complexity, does not rely on additional blocking strategies, and has a small dependence on hyperparameters.	The main limitation of the UPM algorithm is its restricted generality, which is designed and optimized primarily for the e-commerce domain and may not be applicable to other domains. In addition, the algorithm relies on blocking conditions that require domain expertise and in-depth knowledge of the data, increasing the complexity of implementation. The choice of hyperparameters may also affect the matching results and needs to be tuned for specific applications. Finally, UPM currently supports mainly English headings, and its adaptability to multilingual environments needs to be improved.	Future research can extend and optimize the UPM algorithm in the following areas: exploring cross-domain applications, automated blocking condition formulation, automatic tuning of hyperparameters, support for multi-language environments, and integration with other text processing techniques. At the same time, attention is paid to the real-time and scalability of the algorithm to accommodate large-scale datasets and rapid updates.
	(2013, De Bakker M et al.) [115]	This research proposes and validates a new hybrid similarity method that combines product title and attribute information to significantly improve the accuracy of online product duplicate detection. By constructing a weighted similarity based on attribute key-values and extracting model words from key-value pairs, the method in this paper outperforms the existing techniques in several metrics and provides a new and effective solution in the field of product duplicate detection.	Although the hybrid similarity method performs well, its performance depends on the accuracy and completeness of the dataset, its processing efficiency for large-scale datasets needs to be improved, and it is mainly applicable to products with explicit attribute information.	(1) Introducing more metrics and algorithm optimization: future research could explore more string distance metrics and optimize algorithms to improve processing efficiency. (2) Application of ontology and domain knowledge: research on how to use ontology and domain background knowledge to assist duplicate detection and improve accuracy and efficiency. (3) Research on method extension and adaptability: explore the extension of the method to more types of products, and study how to improve its adaptability and practicality in different domains and scenarios.

Table 13. *Cont.*

Domains	Time/Author	Main Contributions	Limitation	Future Research Directions
industry	(2024, Zheng K et al.) [116]	(1) Propose an image–text matching method based on a priori knowledge graph, which improves the model's ability to understand the real scene. (2) Transformer and BERT are utilized in the matching module to improve the model accuracy. (3) Introducing relative entropy to optimize the loss function and reduce inter-modal feature differences. (4) Experimentally verify the contribution of each module to prove the effectiveness of the method.	(1) Challenges of knowledge graph construction and maintenance of updates are not mentioned. (2) High complexity may affect real-time performance. (3) Other potential problems of cross-modal matching are not explored in depth.	(1) Explore more advanced fusion mechanisms to improve matching performance. (2) Study the effective use and dynamic update of knowledge graph. (3) Optimize the model structure to reduce complexity and improve real-time performance. (4) Study other issues of cross-modal matching in depth.
agriculture	(2024, Song Y et al.) [117]	To propose a local matching-based retrieval method for digital agricultural text information to improve retrieval efficiency and provide an effective and scientific retrieval method for digital agricultural text information.	(1) The text notes that the data collected were not comprehensive, which may affect the method's widespread use and accuracy. (2) Although the experimental results show high recall and precision, the text does not discuss in detail how the method performs on different types or sizes of agricultural text data.	(1) Research in this area needs to be further strengthened to collect more comprehensive data to improve the broad applicability and accuracy of the method. (2) Explore the performance of the method on different types and sizes of agricultural text data, as well as the possibility of its application in other related fields. (3) It is possible to study how to integrate more advanced natural language processing techniques into the method to further enhance the retrieval of digital agricultural text information
natural language processing (NLP)	(2022, Xu B et al.) [118]	The multimodal named entity recognition framework proposed in this paper solves the limitation of strict matching's in traditional approach through a cross-modal matching and alignment mechanism, and effectively fuses the semantic information of text and image. The specially designed cross-modal matching (CM) module dynamically adjusts the utilization of visual information according to the similarity between text and image, while the cross-modal alignment (CA) module narrows the semantic gap between the two modalities and improves the recognition accuracy.	The limitations of the framework are mainly in terms of dataset dependency, high model complexity, and limited interpretability. Specific datasets may not fully reflect the complexity of real social media data, model complexity may affect the efficiency of handling large-scale data, and the opacity of the model decision-making process increases the difficulty of user understanding and trust.	Future research can explore the directions of multimodal data extension, dynamic parameter adjustment, contextual information fusion, and model interpretability enhancement. By combining research in these directions with existing frameworks, the performance of multimodal named entity recognition can be further improved and its applications in areas such as social media can be expanded.

Table 13. *Cont.*

Domains	Time/Author	Main Contributions	Limitation	Future Research Directions
	(2018, Zhu G et al.) [105]	This paper proposes two innovative approaches, SCSNED and Category2Vec, to improve the performance of entity disambiguation by utilizing semantic similarity in knowledge graphs and jointly learned embedding models, respectively. These two approaches are validated in real-world datasets.	Although the proposed method is effective, the aspects of universality, accuracy of semantic similarity computation, and real-time performance still need to be further verified and optimized.	Future research could explore more advanced semantic representations, cross-domain cross-language disambiguation techniques, optimization of real-time performance, and interactive disambiguation methods incorporating user feedback.
	(2024, Gong J et al.) [119]	A unified framework named MORE is proposed to effectively solve the over-segmentation and over-merging problems in the author name disambiguation (AND) task. By combining LightGBM, HAC, and improved HGAT (iHGAT) techniques, and introducing the SimCSE algorithm to optimize the paper representation, MORE significantly improves the performance on the AND task.	(1) Complexity of academic domains: existing author name disambiguation methods have limitations in dealing with the complexity and diversity of academic domains, especially when dealing with special or complex author names. (2) Maintenance of knowledge graph: with the growth and update of academic data, it is a challenge to effectively maintain and update the knowledge graph in terms of timeliness and accuracy. (3) Granularity of disambiguation: current research focuses on disambiguation at the paper level, with less attention paid to disambiguation at finer granularity, such as at the paragraph or sentence level.	(1) Technological innovation: explore and apply more advanced natural language processing and machine learning techniques to improve the accuracy and efficiency of author name disambiguation. (2) Dynamic updating of knowledge graph: to study how to construct and maintain a knowledge graph that can be dynamically updated and expanded to adapt to changes in academic data. (3) Fine-grained disambiguation: in-depth study of fine-grained disambiguation methods to improve the ability to identify and handle author name conflicts in fine-grained text analysis tasks. (4) Practical activities and competitions: promote the development of related technologies and encourage researchers to explore new methods and ideas by conducting practical activities such as online knowledge alignment competitions.
	(2011, Arifoğlu D et al.) [120]	Two innovative techniques are proposed for solving difficult problems in historical document analysis. One is a cross-document word-matching-based approach, which realizes the automatic segmentation of words in damaged and deformed historical documents; the other is an image analysis method based online and sub-pattern detection, which has been successfully applied to pattern matching and duplicate pattern detection in images of Islamic calligraphy (Kufic script). These two techniques not only provide new tools for the study of historical documents, but also expand the technical boundaries of the document analysis field.	Although certain results have been achieved in word segmentation of historical documents and pattern matching of Islamic calligraphy images, they are still facing problems such as limited recognition accuracy, strong dependence on data sources, difficulties in processing complex images, and algorithmic performance bottlenecks, which are challenges that limit the wide application and in-depth development of the existing technologies.	Future research will be dedicated to improving the accuracy and efficiency of historical document recognition and Islamic calligraphy image analysis through technological innovation and algorithm optimization. At the same time, it is possible to explore the possibility of multimodal data fusion, develop user-friendly interactive tools, and expand the application of the research results to the digitization of ancient books and the preservation of cultural heritage to promote the progress and development of related disciplines.

In education, matching technologies have significantly improved planning efficiency by automatically aligning learning objectives within and outside of subjects. In addition, the design of data science pipelines has enabled the analysis of learning objective similarities and the identification of subject clustering and dependencies. Meanwhile, text-matching software has demonstrated its versatility by being used not only as a tool for educating students but also for detecting inappropriate behavior. In the medical field, matching technology provides powerful support for doctors. It is used in the retrieval and generation of radiology reports, in the linking of medical text entities, and in Q&A systems to help doctors carry out their diagnosis and treatment more efficiently. In the economic domain, matching technology provides powerful support for the accurate provision of job information, the formulation of industry-specific recommendations, and the promotion of policy tourism and economic cooperation. Especially in the e-commerce field, matching technology has made a big difference, not only improving the accuracy of matching product titles in online shops and e-commerce platforms so that users can find the products they need more quickly but also significantly improving the efficiency and accuracy of product retrieval in the e-commerce industry. In addition, matching technology also solves the problem of duplicate product detection in online shops, which improves the operational efficiency and market competitiveness of the platform.

In the industrial field, text-matching technology is applied to robots, which enhances the model's ability to understand the real scene and promotes the development of industrial automation. In the field of agriculture, the application of text-matching methods improves the retrieval efficiency of digital agricultural text information and provides an effective method for the scientific retrieval of agricultural information. In named entity recognition, named entity disambiguation, author name disambiguation, and historical documents in natural language processing, matching techniques play a crucial role. In named entity recognition, the matching technique, which dynamically adjusts visual information, can more effectively fuse the semantic information of text and images to improve the accuracy of recognition. In this process, the cross-modal alignment (CA) module plays a key role, narrowing the semantic gap between the two modalities of text and image, making the recognition results more accurate. In named entity disambiguation, the matching technique is also crucial. By exploiting the semantic similarity in the knowledge graph, the same entity in different contexts can be judged more accurately for disambiguation. In addition, the embedding model of joint learning also provides strong support for entity disambiguation, improving the performance of entity disambiguation by capturing the intrinsic relationship between entities. In summary, matching techniques play a crucial role in the field of named entity recognition and named entity disambiguation, which not only improves the accuracy of recognition but also provides strong support for other tasks in natural language processing.

In author name disambiguation, LightGBM creates a paper similarity matrix and hierarchical clustering method (HAC) for clustering, which enables paper clustering to be implemented more accurately and efficiently, thus avoiding the problem of over-merging papers. The automatic segmentation of words in damaged and deformed historical documents is achieved based on cross-document word matching in historical documents. An image analysis method based on online and sub-pattern detection is successfully applied to pattern matching and duplicate pattern detection in Islamic calligraphy (Kufic script) images. Through the above analyses, matching techniques have shown a wide range of application prospects and significant results in different fields, but at the same time, there are some limitations and challenges that require further research and exploration. To enable the text-matching technique to be applied in the specific application areas mentioned above, this study provides practical insights or guidelines for implementing the sentence-matching technique in real-world applications, including some hints on hardware requirements, software tools, and optimization strategies.

When implementing sentence-matching technology in the specific application areas mentioned above, it is first necessary to define the specific application requirements, in-

cluding application scenarios, the desired accuracy, and response time. These requirements will directly guide the subsequent technology selection and model design. Subsequently, a high-quality, clearly labeled dataset is the basis for training an efficient sentence-matching model, so the data preparation stage is crucial. In terms of hardware, to meet the training and inference needs of deep learning models, it is recommended to choose servers equipped with high-performance GPUs or utilize cloud computing services. Meanwhile, a sufficient memory is essential to support data loading and computation during model training and inference. For storage needs, it is recommended to use high-speed SSD hard disks or distributed storage systems to accommodate huge datasets and model files.

In terms of software tool selection, deep learning frameworks such as TensorFlow and PyTorch are indispensable tools that provide a rich set of APIs and tools that greatly simplify the model development and deployment process. Text-processing libraries such as NLTK and spaCy, on the other hand, can help us efficiently perform text preprocessing and feature extraction. In addition, the combination of pre-trained models and specialized semantic matching tools, such as BERT [17] and Roberta [20], can further improve the accuracy of sentence matching.

To further improve the performance of the model, we can employ a series of optimization strategies. Data enhancement techniques [63] can increase the diversity of training data and improve the generalization ability of the model. Model compression techniques [75] can reduce the size and computations of the model, thus speeding up the inference. Multi-stage training strategies, especially pre-training and fine-tuning, can make full use of large-scale unlabeled data and small amounts of labeled data. Integrated learning [58] can further improve sentence-matching accuracy by combining predictions from multiple models. Finally, hardware optimization techniques such as GPU acceleration and distributed computing can be utilized to significantly improve the efficiency of model training and inference.

8. Open Problems and Challenges

8.1. Limitations of State-of-the-Art Macro Models in Sentence-Matching Tasks

Despite the dominance of the BERT [17] family and its improved versions in sentence-matching tasks, some state-of-the-art macro models, such as GPT4 [24] and GPT-5, are currently difficult to apply in sentence-matching tasks due to their not-yet-open-source nature. This limits to some extent the possibility that sentence-matching tasks can be advanced and developed using new technologies.

The application of state-of-the-art macro models to sentence-matching tasks is limited by multiple challenges. Among them, the issue of model openness driven by commercial interests is particularly critical. These models are usually owned by large technology companies or research organizations as a core competency and thus are not readily made public. Even partial disclosure may be accompanied by usage restrictions and high licensing fees, making it unaffordable for researchers and small developers. This openness issue not only limits the wide application of the model in sentence-matching tasks but also may hinder research progress in the field. The lack of support from advanced models makes it difficult for researchers to make breakthroughs in sentence-matching tasks and to fully utilize the potential capabilities of these models. Therefore, addressing the issue of model openness driven by commercial interests is crucial to advancing research and development in sentence-matching tasks.

8.2. Model-Training Efficiency of Sentence-Matching Methods

Most of the sentence-matching methods are based on large language models, which leads to high requirements on device memory during the training process, and the training time is up to several hours or even dozens of hours. This inefficient training process not only increases the consumption of computational resources but also limits the rapid deployment and iteration of the model in real applications. Therefore, we need to investigate how

to optimize the model-training process, improve the training efficiency, and reduce the dependence on device memory, as well as reducing the training time.

It is difficult to solve the problem of the model-training efficiency of sentence-matching methods, mainly because of the following reasons: the high complexity of large language models leads to a large demand for memory, which poses a challenge to users with limited hardware; the training time is long, and the processing of a large amount of data requires multiple rounds of iterations, which consumes computational resources and affects the efficiency; how to reduce the dependence on memory and the training time, while guaranteeing performance, involves many technical challenges in terms of the model structure, the algorithm design, and data processing.

8.3. Assertion Comprehension and Non-Matching Information Interference in Sentence Matching

A key challenge in the sentence-matching task is the lack of in-depth understanding of the sentence claim, while being susceptible to interference from non-matching information. Sentence claims are the embodiment of the core meaning of a sentence, while noisy information or non-matching content may mislead the matching process. To address this problem, research is needed to design more effective model architectures or introduce new semantic understanding mechanisms to enhance the capture and recognition of sentence claims, while reducing the interference of non-matching information.

The main challenges of claim comprehension and non-matching information interference in sentence matching are the complexity and subtlety of sentence claim comprehension, which require the model to deeply analyze the semantics and context of the sentence; at the same time, non-matching information interference is difficult to overcome, as it requires the model to have a strong semantic differentiation ability to filter the interfering content. Therefore, solving these problems requires the model to realize significant breakthroughs in semantic understanding and information filtering.

8.4. Confusion or Omission in Handling Key Detail Facts

Key Detail Facts in Sentence-Matching Processing Detail information is crucial for accurate matching in sentence-matching tasks [121]. However, existing methods for sentence matching often suffer from the problem of confusion or omission when dealing with key detail facts. Losing certain detailed information may lead to inaccurate matching results. Therefore, there is a need to investigate how to optimize sentence-matching algorithms to capture and process key detail facts more accurately. This may involve the optimization of extraction, representation, and matching strategies for detailed information.

The main difficulties in processing key detail facts in sentence-matching tasks are the accurate extraction of detail information, effective representation, and the optimization of matching strategies. Due to the diversity and complexity of languages, it is extremely challenging to extract key details in sentences; meanwhile, existing methods may not be able to fully capture the full meaning of the detail information, leading to confusion or omission during the matching process; in addition, it is also a major challenge to design a matching strategy that can fully take into account the importance of the detail information, which may lead to inaccurate matching results.

8.5. Diversity and Balance Challenges of Sentence-Matching Datasets

Currently, most sentence-matching datasets focus on the English domain, while there are relatively few Chinese datasets. In addition, there is a relative lack of application-specific datasets to meet the application requirements in different scenarios. This imbalance and lack of datasets limit the generalization ability and application scope of sentence-matching models. Therefore, there is a need to work on constructing and expanding diverse and balanced sentence-matching datasets, especially increasing Chinese datasets and datasets of specific application domains, to promote the development and application of sentence-matching technology.

The diversity and balance challenges of sentence-matching datasets are significant, and are mainly reflected in the uneven distribution of languages, the lack of application-specific datasets, and the difficulty of guaranteeing the balance of datasets, etc. English datasets are relatively rich, while Chinese and other language datasets are scarce, which limits the generalization ability of the model in different languages. Meanwhile, the lack of domain-specific datasets results in limited model performance in specific scenarios. In addition, constructing a balanced dataset faces difficulties in data collection and labeling, and an unbalanced dataset may affect the training effect and matching accuracy of the model.

9. Future Research Directions

9.1. Future Research Direction I: The Problem of Sentence-Matching Model Publicity Driven by Commercial Interests and Solutions

Future research directions should focus on solving the problem of model openness driven by commercial interests, promoting open-source models and community construction, innovating model authorization and sharing mechanisms, strengthening privacy protection and model security, optimizing model performance and adaptive capability, and strengthening interdisciplinary cooperation and innovative applications to promote the research and development of sentence-matching tasks.

9.2. Future Research Directions II: Optimization Strategies That Focus on Model-Training Efficiency for Sentence-Matching Methods

Many users face the problem of a low training efficiency due to the high complexity of large language models, the high memory requirements, and long training time. To address this challenge, future research will explore the lightweight design of the model structure, the efficient tuning of algorithms, and the improvement of data-processing techniques. By optimizing the model structure, adopting efficient algorithms, and fine-tuning the data processing, we aim to reduce memory dependency and training time, thus promoting the development and application of sentence-matching techniques. These optimization strategies will help address current and future training efficiency issues and promote continued progress in the field of sentence matching.

9.3. Future Research Directions III: Optimization Strategies Focusing on the Efficiency of Model Training for Sentence-Matching Methods

These two challenges place high demands on the model's semantic comprehension and information-filtering abilities. Propositional understanding is the core problem, which requires the model to deeply parse the semantic and contextual information of sentences to capture and understand deeper meanings and subtle differences. However, existing models have limitations in semantic understanding. Meanwhile, the interference of non-matching information is also a major challenge, and the model needs to filter out irrelevant information and focus only on matching key content. However, the existing models are still insufficient in dealing with interference. Therefore, future research should be devoted to improving the model's semantic understanding and information-filtering capabilities, to better address these two challenges and promote new progress in the research and application of sentence-matching tasks.

9.4. Future Research Direction IV: Difficulties and Breakthroughs in Processing Key Detail Facts in Sentence-Matching Tasks

Sentence-matching tasks face multiple challenges in processing key detail facts, including the extraction of detailed information, representation, and the optimization of matching strategies. First, accurately extracting key detail information in sentences is a basic but challenging task. Due to the complexity and diversity of language, it is often difficult for existing methods to capture these details comprehensively, leading to confusion or omission in the matching process. Therefore, future research should explore more efficient information extraction techniques, such as utilizing recent advances in deep learning or natural language processing to enhance the ability to capture detailed infor-

mation. Second, effective representation of this critical detail information is crucial for matching. Existing representation methods may not be able to convey the meaning of detailed information completely, leading to matching bias. Therefore, research should focus on more advanced representation-learning techniques, such as distributed representation or representation models that incorporate contextual information, to reflect the connotations of detailed information more accurately. Finally, the optimization of matching strategies is equally critical. Existing strategies often ignore the importance of detailed information, leading to inaccurate matching results. Future research should explore smarter matching strategies, such as using the attention mechanism to emphasize the role of key details in matching or combining machine learning techniques to automatically learn the optimal matching strategy.

9.5. Future Research Direction V: Addressing the Challenges of Diversity and Balance in Sentence-Matching Datasets

Although sentence-matching tasks are widely used, the diversity and balance of datasets are still the key challenges for performance improvement and generalization ability. Currently, English datasets are relatively rich, while Chinese and other languages are relatively scarce, which limits the model's matching ability in different languages. At the same time, datasets for specific domains such as finance, healthcare, law, etc., are also insufficient, which limits the application of the model in these domains. In addition, the problem of dataset balance is an important factor that affects the model-training effect and matching accuracy. Therefore, future research should focus on constructing multilingual and domain-specific datasets, such as generating data using GPT4 [24] and GPT5, and exploring effective data-balancing methods, such as over-sampling, under-sampling, and adopting appropriate loss functions, to improve the model's matching performance and generalization ability. These efforts will help promote the application of sentence-matching techniques in more languages and domains and facilitate the continuous development of the natural language processing field.

10. Conclusions

Text-matching technology has experienced a stepwise development from character-based matching methods, recursive neural network-based matching methods, and sentence vector matching methods based on large models, to large model-based matching methods utilizing different fine-grained features, and it has made significant progress. While string-based methods are simple and intuitive, they lack an in-depth understanding of semantics, while corpus-based methods make up for this by utilizing large-scale data to capture semantic relationships. With the development of deep learning, recurrent neural networks and sentence semantic interaction matching methods can further capture the hierarchical structure of text and contextual information, improving the accuracy of matching. However, these methods have high computational complexity. Graph structure-based matching methods are suitable for specific tasks such as relationship extraction but are challenging for unstructured text processing. In recent years, with improvements in arithmetic power and the enhancement of data resources, sentence vector matching methods based on large language models capture rich sentence semantic information through large-scale corpus pre-training and then utilize it to directly complete the calculation of the matching probability. In addition, large model-based matching methods utilizing different fine-grained features can accurately capture differences, but the computational complexity is also higher. These methods are developing in a complementary way, and jointly promote the progress of text-matching technology. In the future, with the promotion of downstream applications, matching technology will make breakthroughs in multimodal matching, model optimization, multi-granularity semantic understanding, and the domain adaptability of images, videos, and texts, which will further promote the development of matching technology and bring broader application prospects for its downstream applications.

Author Contributions: Conceptualization, P.J. and X.C.; methodology, P.J. investigation, P.J.; resources, X.C.; data curation, P.J.; writing—original draft preparation, P.J.; writing—review and editing, P.J.; visualization, X.C.; supervision, X.C.; project administration, X.C.; funding acquisition, X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the intelligent integrated media platform r&d and application demonstration project (PM21014X).

Data Availability Statement: Not applicable.

Conflicts of Interest: The author(s) declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

1. Hunt, J.W.; Szymanski, T.G. A Fast Algorithm for Computing Longest Common Subsequences. *Commun. ACM* **1977**, *20*, 350–353. [CrossRef]
2. Levenshtein, V.I. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Sov. Phys. Dokl.* **1966**, *10*, 707–710.
3. Winkler, W.E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. 1990. Available online: https://files.eric.ed.gov/fulltext/ED325505.pdf (accessed on 21 May 2024).
4. Dice, L.R. Measures of the Amount of Ecologic Association between Species. *Ecology* **1945**, *26*, 297–302. [CrossRef]
5. Jaccard, P. The Distribution of the Flora in the Alpine Zone. 1. *New Phytol.* **1912**, *11*, 37–50. [CrossRef]
6. Salton, G.; Buckley, C. Term Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523. [CrossRef]
7. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.
8. Landauer, T.K.; Foltz, P.W.; Laham, D. An Introduction to Latent Semantic Analysis. *Discourse Process.* **1998**, *25*, 259–284. [CrossRef]
9. Mueller, J.; Thyagarajan, A. Siamese Recurrent Architectures for Learning Sentence Similarity. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 2786–2792.
10. Neculoiu, P.; Versteegh, M.; Rotaru, M. Learning Text Similarity with Siamese Recurrent Networks. In Proceedings of the 1st Workshop on Representation Learning for NLP, Berlin, Germany, 11 August 2016; pp. 148–157.
11. Lu, X.; Deng, Y.; Sun, T.; Gao, Y.; Feng, J.; Sun, X.; Sutcliffe, R. MKPM: Multi Keyword-Pair Matching for Natural Language Sentences. *Appl. Intell.* **2022**, *52*, 1878–1892. [CrossRef]
12. Deng, Y.; Li, X.; Zhang, M.; Lu, X.; Sun, X. Enhanced Distance-Aware Self-Attention and Multi-Level Match for Sentence Semantic Matching. *Neurocomputing* **2022**, *501*, 174–187. [CrossRef]
13. Kim, S.; Kang, I.; Kwak, N. Semantic Sentence Matching with Densely-Connected Recurrent and Co-Attentive Information. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Hilton Hawaiian Village, Honolulu, HI, USA, 27 January–1 February 2019; pp. 6586–6593.
14. Zhang, K.; Lv, G.; Wang, L.; Wu, L.; Chen, E.; Wu, F.; Xie, X. Drr-Net: Dynamic Re-Read Network for Sentence Semantic Matching. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Hilton Hawaiian Village, Honolulu, HI, USA, 27 January–1 February 2019; pp. 7442–7449.
15. Wang, Z.; Hamza, W.; Florian, R. Bilateral Multi-Perspective Matching for Natural Language Sentences. *arXiv* **2017**, arXiv:1702.03814.
16. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.
17. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
18. Joshi, M.; Chen, D.; Liu, Y.; Weld, D.S.; Zettlemoyer, L.; Levy, O. Spanbert: Improving Pre-Training by Representing and Predicting Spans. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 64–77. [CrossRef]
19. Yang, Z.L.; Dai, Z.H.; Yang, Y.M.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv* **2019**, arXiv:1906.08237.
20. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A Robustly Optimized Bert Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
21. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://www.mikecaptain.com/resources/pdf/GPT-1.pdf (accessed on 21 May 2024).
22. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models Are Unsupervised Multitask Learners. *OpenAI Blog* **2019**, *1*, 9.
23. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A. Language Models Are Few-Shot Learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
24. Achiam, J.; Adler, J.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F.L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.

25. Ren, X.; Zhou, P.; Meng, X.; Huang, X.; Wang, Y.; Wang, W.; Li, P.; Zhang, X.; Podolskiy, A.; Arshinov, G. PanGu-{\Sigma}: Towards Trillion Parameter Language Model with Sparse Heterogeneous Computing. *arXiv* **2023**, arXiv:2303.10845.
26. Zhang, K.; Wu, L.; Lv, G.Y.; Wang, M.; Chen, E.H.; Ruan, S.L.; Assoc Advancement Artificial, I. Making the Relation Matters: Relation of Relation Learning Network for Sentence Semantic Matching. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; pp. 14411–14419.
27. Mysore, S.; Cohan, A.; Hope, T. Multi-Vector Models with Textual Guidance for Fine-Grained Scientific Document Similarity. *arXiv* **2021**, arXiv:2111.08366.
28. Zou, Y.; Liu, H.; Gui, T.; Wang, J.; Zhang, Q.; Tang, M.; Li, H.; Wang, D.; Assoc Computa, L. Divide and Conquer: Text Semantic Matching with Disentangled Keywords and Intents. In Proceedings of the 60th Annual Meeting of the Association-for-Computational-Linguistics, Dublin, Ireland, 22–27 May 2022; pp. 3622–3632.
29. Yao, D.; Alghamdi, A.; Xia, Q.; Qu, X.; Duan, X.; Wang, Z.; Zheng, Y.; Huai, B.; Cheng, P.; Zhao, Z. A General and Flexible Multi-Concept Parsing Framework for Multilingual Semantic Matching. *arXiv* **2024**, arXiv:2403.02975.
30. Asha, S.; Krishna, S.T. Semantics-Based String Matching: A Review of Machine Learning Models. *Int. J. Intell. Syst.* **2024**, *12*, 347–356.
31. Hu, W.; Dang, A.; Tan, Y. A Survey of State-of-the-Art Short Text Matching Algorithms. In Proceedings of the Data Mining and Big Data: 4th International Conference, Chiang Mai, Thailand, 26–30 July 2019; pp. 211–219.
32. Wang, J.; Dong, Y. Measurement of Text Similarity: A Survey. *Information* **2020**, *11*, 421. [CrossRef]
33. Deza, E.; Deza, M.M.; Deza, M.M.; Deza, E. Encyclopedia of Distances. In *Encyclopedia of Distances*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–583.
34. Li, B.; Han, L. Distance Weighted Cosine Similarity Measure for Text Classification. In Proceedings of the Intelligent Data Engineering and Automated Learning—IDEAL 2013: 14th International Conference, Hefei, China, 20–23 October 2013; pp. 611–618.
35. Sidorov, G.; Gelbukh, A.; Gómez-Adorno, H.; Pinto, D. Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model. *Comput. Sist.* **2014**, *18*, 491–504. [CrossRef]
36. Dean, J.; Ghemawat, S. Mapreduce: Simplified Data Processing on Large Clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
37. Bejan, I.; Sokolov, A.; Filippova, K. Make Every Example Count: On Stability and Utility of Self-Influence for Learning from Noisy NLP Datasets. *arXiv* **2023**, arXiv:2302.13959.
38. Yedidia, J.S.; Freeman, W.T.; Weiss, Y. Understanding Belief Propagation and Its Generalizations. *Explor. Artif. Intell. New Millenn.* **2003**, *8*, 0018–9448.
39. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
40. Robertson, S.E.; Walker, S. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In Proceedings of the International ACM Sigir Conference on Research and Development in Information Retrieval SIGIR '94, Dublin, Ireland, 3–6 July 1994; pp. 232–241.
41. Katz, S. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Trans. Acoust. Speech Signal Process.* **1987**, *35*, 400–401. [CrossRef]
42. Akritidis, L.; Alamaniotis, M.; Fevgas, A.; Tsompanopoulou, P.; Bozanis, P. Improving Hierarchical Short Text Clustering through Dominant Feature Learning. *Int. J. Artif. Intell. Tools* **2022**, *31*, 2250034. [CrossRef]
43. Bulsari, A.B.; Saxen, H. A recurrent neural network model. In Proceedings of the 1992 International Conference (ICANN-92), Brighton, UK, 4–7 September 1992; pp. 1091–1094.
44. Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Zettlemoyer, L. Deep Contextualized Word Representations. *arXiv* **2018**, arXiv:1802.05365.
45. Levy, O.; Goldberg, Y. Dependency-Based Word Embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Tutorials, Baltimore, MD, USA, 22 June 2014; pp. 302–308.
46. Le, Q.; Mikolov, T. Distributed Representations of Sentences and Documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 22–24 June 2014; pp. 1188–1196.
47. Chen, M. Efficient Vector Representation for Documents through Corruption. *arXiv* **2017**, arXiv:1707.02377.
48. Petroni, F.; Rocktäschel, T.; Lewis, P.; Bakhtin, A.; Wu, Y.; Miller, A.H.; Riedel, S. Language Models as Knowledge Bases? *arXiv* **2019**, arXiv:1909.01066.
49. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
50. Tabassum, A.; Patil, R.R. A Survey on Text Pre-Processing & Feature Extraction Techniques in Natural Language Processing. *Int. Res. J. Eng. Technol.* **2020**, *7*, 4864–4867.
51. Elsafty, A. Document Similarity Using Dense Vector Representation. 2017. Available online: https://www.inf.uni-hamburg.de/en/inst/ab/lt/teaching/theses/completed-theses/2017-ma-elsafty.pdf (accessed on 22 May 2024).
52. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv* **2014**, arXiv:1409.0473.
53. Shorten, C.; Khoshgoftaar, T.M.; Furht, B. Text Data Augmentation for Deep Learning. *J. Big Data* **2021**, *8*, 34. [CrossRef]
54. Pan, S.J.; Yang, Q.A. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]

55. Liu, M.; Zhang, Y.; Xu, J.; Chen, Y. Deep Bi-Directional Interaction Network for Sentence Matching. *Appl. Intell.* **2021**, *51*, 4305–4329. [CrossRef]
56. Park, D.S.; Chan, W.; Zhang, Y.; Chiu, C.C.; Zoph, B.; Cubuk, E.D.; Le, Q.V.; Int Speech Commun, A. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *arXiv* **2019**, arXiv:1904.08779.
57. Peng, S.; Cui, H.; Xie, N.; Li, S.; Zhang, J.; Li, X. Enhanced-RCNN: An Efficient Method for Learning Sentence Similarity. In Proceedings of the Web Conference 2020: Proceedings of the World Wide Web Conference WWW 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2500–2506.
58. Mahajan, P.; Uddin, S.; Hajati, F.; Moni, M.A. Ensemble Learning for Disease Prediction: A Review. *Healthcare* **2023**, *11*, 1808. [CrossRef]
59. Zhu, G.; Iglesias, C.A. Computing Semantic Similarity of Concepts in Knowledge Graphs. *IEEE Trans. Knowl. Data Eng.* **2016**, *29*, 72–85. [CrossRef]
60. Chen, L.; Zhao, Y.; Lyu, B.; Jin, L.; Chen, Z.; Zhu, S.; Yu, K. Neural Graph Matching Networks for Chinese Short Text Matching. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 6–8 July 2020; pp. 6152–6158.
61. Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K. Large Scale Distributed Deep Networks. In Proceedings of the Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 25.
62. Goldar, P.; Rai, Y.; Kushwaha, S. A Review on Parallelization of Big Data Analysis and Processing. *IJETCSE* **2016**, *23*, 60–65.
63. Pluščec, D.; Šnajder, J. Data Augmentation for Neural NLP. *arXiv* **2023**, arXiv:2302.11412.
64. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713.
65. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning Both Weights and Connections for Efficient Neural Networks. In Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS), Palais des Congrès de Montréal Convention and Exhibition Center, Montreal, QC, Canada, 8–10 December 2015; Volume 28.
66. Chen, Z.; Qu, Z.; Quan, Y.; Liu, L.; Ding, Y.; Xie, Y. Dynamic n: M Fine-Grained Structured Sparse Attention Mechanism. In Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, Montreal, QC, Canada, 25 February–1 March 2023; pp. 369–379.
67. Tenney, I.; Das, D.; Pavlick, E. BERT Rediscovers the Classical NLP Pipeline. *arXiv* **2019**, arXiv:1905.05950.
68. Howard, J.; Ruder, S. Universal Language Model Fine-Tuning for Text Classification. *arXiv* **2018**, arXiv:1801.06146.
69. Fedus, W.; Goodfellow, I.; Dai, A.M. Maskgan: Better Text Generation via Filling in the _. *arXiv* **2018**, arXiv:1801.07736.
70. Dai, Z.H.; Yang, Z.L.; Yang, Y.M.; Carbonell, J.; Le, Q.V.; Salakhutdinov, R. Acl Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In Proceedings of the 57th Annual Meeting of the Association-for-Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 2978–2988.
71. He, M.; Liu, Y.; Wu, B.; Yuan, J.; Wang, Y.; Huang, T.; Zhao, B. Efficient Multimodal Learning from Data-Centric Perspective. *arXiv* **2024**, arXiv:2402.11530.
72. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.Q.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 67.
73. Vinyals, O.; Le, Q. A Neural Conversational Model. *arXiv* **2015**, arXiv:1506.05869.
74. Sahin, U.; Kucukkaya, I.E.; Toraman, C. ARC-NLP at PAN 2023: Hierarchical Long Text Classification for Trigger Detection. *arXiv* **2023**, arXiv:2307.14912.
75. Neill, J.O. An Overview of Neural Network Compression. *arXiv* **2020**, arXiv:2006.03669.
76. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv* **2019**, arXiv:1910.01108.
77. Bordia, S.; Bowman, S.R.; Assoc Computat, L. Identifying and Reducing Gender Bias in Word-Level Language Models. *arXiv* **2019**, arXiv:1904.03035.
78. Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; Choi, Y. The Curious Case of Neural Text Degeneration. *arXiv* **2019**, arXiv:1904.09751.
79. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the 2021 ACM conference on fairness, accountability, and transparency, Virtual Event, Toronto, ON, Canada, 3–10 March 2021; pp. 610–623.
80. Treviso, M.; Lee, J.-U.; Ji, T.; van Aken, B.; Cao, Q.; Ciosici, M.R.; Hassid, M.; Heafield, K.; Hooker, S.; Raffel, C. Efficient Methods for Natural Language Processing: A Survey. *Trans. Assoc. Comput. Linguist.* **2023**, *11*, 826–860. [CrossRef]
81. He, W.; Dai, Y.; Yang, M.; Sun, J.; Huang, F.; Si, L.; Li, Y. Space-3: Unified Dialog Model Pre-Training for Task-Oriented Dialog Understanding and Generation. *arXiv* **2022**, arXiv:2209.06664.
82. He, W.; Dai, Y.; Zheng, Y.; Wu, Y.; Cao, Z.; Liu, D.; Jiang, P.; Yang, M.; Huang, F.; Si, L. Galaxy: A Generative Pre-Trained Model for Task-Oriented Dialog with Semi-Supervised Learning and Explicit Policy Injection. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 10749–10757.
83. He, W.; Dai, Y.; Hui, B.; Yang, M.; Cao, Z.; Dong, J.; Huang, F.; Si, L.; Li, Y. Space-2: Tree-Structured Semi-Supervised Contrastive Pre-Training for Task-Oriented Dialog Understanding. *arXiv* **2022**, arXiv:2209.06638.

84. Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C.; Wexler, J.; Viegas, F.; Sayres, R. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 2668–2677.
85. Brundage, M.; Avin, S.; Clark, J.; Toner, H.; Eckersley, P.; Garfinkel, B.; Dafoe, A.; Scharre, P.; Zeitzoff, T.; Filar, B. The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation. *arXiv* **2018**, arXiv:1802.07228.
86. Lee, A.; Miranda, B.; Koyejo, S. Beyond Scale: The Diversity Coefficient as a Data Quality Metric Demonstrates LLMs Are Pre-Trained on Formally Diverse Data. *arXiv* **2023**, arXiv:2306.13840.
87. Mondal, R.; Tang, A.; Beckett, R.; Millstein, T.; Varghese, G. What Do LLMs Need to Synthesize Correct Router Configurations? *arXiv* **2023**, arXiv:2307.04945.
88. Mumtarin, M.; Chowdhury, M.S.; Wood, J. Large Language Models in Analyzing Crash Narratives—A Comparative Study of ChatGPT, BARD and GPT-4. *arXiv* **2023**, arXiv:2308.13563.
89. Tsai, C.-M.; Chao, C.-J.; Chang, Y.-C.; Kuo, C.-C.J.; Hsiao, A.; Shieh, A. Challenges and Opportunities in Medical Artificial Intelligence. *APSIPA Trans. Signal Inf. Process.* **2023**, *12*, e205. [CrossRef]
90. Zhong, T.; Wei, Y.; Yang, L.; Wu, Z.; Liu, Z.; Wei, X.; Li, W.; Yao, J.; Ma, C.; Li, X. Chatabl: Abductive Learning via Natural Language Interaction with Chatgpt. *arXiv* **2023**, arXiv:2304.11107.
91. Liu, Y.; Han, T.; Ma, S.; Zhang, J.; Yang, Y.; Tian, J.; He, H.; Li, A.; He, M.; Liu, Z. Summary of ChatGPT-Related Research and Perspective Towards the Future of Large Language Models. *Meta-Radiology* **2023**, *1*, 100017. [CrossRef]
92. Sellam, T.; Das, D.; Parikh, A.P. BLEURT: Learning Robust Metrics for Text Generation. *arXiv* **2020**, arXiv:2004.04696.
93. Rahm, E.; Do, H.H. Data Cleaning: Problems and Current Approaches. *IEEE Data Eng. Bull.* **2000**, *23*, 3–13.
94. Candemir, S.; Nguyen, X.V.; Folio, L.R.; Prevedello, L.M. Training Strategies for Radiology Deep Learning Models in Data-Limited Scenarios. *Radiol. Artif. Intell.* **2021**, *3*, e210014. [CrossRef]
95. Young, P.; Lai, A.; Hodosh, M.; Hockenmaier, J. From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions. *Trans. Assoc. Comput. Linguist.* **2014**, *2*, 67–78. [CrossRef]
96. Khot, T.; Sabharwal, A.; Clark, P. Scitail: A Textual Entailment Dataset from Science Question Answering. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), Hilton New Orleans Riverside, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
97. Xu, L.; Hu, H.; Zhang, X.; Li, L.; Cao, C.; Li, Y.; Xu, Y.; Sun, K.; Yu, D.; Yu, C. CLUE: A Chinese Language Understanding Evaluation Benchmark. *arXiv* **2020**, arXiv:2004.05986.
98. Hu, H.; Richardson, K.; Xu, L.; Li, L.; Kübler, S.; Moss, L.S. Ocnli: Original Chinese Natural Language Inference. *arXiv* **2020**, arXiv:2010.05444.
99. Liu, X.; Chen, Q.; Deng, C.; Zeng, H.; Chen, J.; Li, D.; Tang, B. Lcqmc: A Large-Scale Chinese Question Matching Corpus. In Proceedings of the the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 1952–1962.
100. Chen, J.; Chen, Q.; Liu, X.; Yang, H.; Lu, D.; Tang, B. The Bq Corpus: A Large-Scale Domain-Specific Chinese Corpus for Sentence Semantic Equivalence Identification. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4946–4951.
101. Shankar Iyer, N.D. First Quora Dataset Release: Question Pairs. 2017. Available online: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs (accessed on 21 May 2024).
102. Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; Zamparelli, R. A SICK Cure for the Evaluation of Compositional Distributional Semantic Models. In Proceedings of the the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, 26–31 May 2014; pp. 216–223.
103. Dolan, B.; Brockett, C. Automatically Constructing a Corpus of Sentential Paraphrases. In Proceedings of the Third International Workshop on Paraphrasing, Jeju Island, Korea, 14 October 2005; pp. 9–16.
104. Li, J.; Sun, A.; Han, J.; Li, C. A Survey on Deep Learning for Named Entity Recognition. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 50–70. [CrossRef]
105. Zhu, G.; Iglesias, C.A. Exploiting Semantic Similarity for Named Entity Disambiguation in Knowledge Graphs. *Expert. Syst. Appl.* **2018**, *101*, 8–24. [CrossRef]
106. Alkhidir, T.; Awad, E.; Alshamsi, A. Understanding the Progression of Educational Topics via Semantic Matching. *arXiv* **2024**, arXiv:2403.05553.
107. Hayden, K.A.; Eaton, S.E.; Pethrick, H.; Crossman, K.; Lenart, B.A.; Penaluna, L.-A. A Scoping Review of Text-Matching Software Used for Student Academic Integrity in Higher Education. *Int. Educ. Res.* **2021**, *2021*, 4834860. [CrossRef]
108. Jeong, J.; Tian, K.; Li, A.; Hartung, S.; Adithan, S.; Behzadi, F.; Calle, J.; Osayande, D.; Pohlen, M.; Rajpurkar, P. Multimodal Image-Text Matching Improves Retrieval-Based Chest X-Ray Report Generation. In Proceedings of the Medical Imaging with Deep Learning, Paris, France, 3–5 July 2024; pp. 978–990.
109. Luo, Y.-F.; Sun, W.; Rumshisky, A. A Hybrid Normalization Method for Medical Concepts in Clinical Narrative Using Semantic Matching. *AMIA Jt. Summits Transl. Sci. Proc.* **2019**, *2019*, 732.
110. Wang, L.; Zhang, T.; Tian, J.; Lin, H. An Semantic Similarity Matching Method for Chinese Medical Question Text. In Proceedings of the 8th China Health Information Processing Conference, Hangzhou, China, 21–23 October 2022; pp. 82–94.

111. Ajaj, S.H. AI-Driven Optimization of Job Advertisements through Knowledge-Based Techniques and Semantic Matching. *Port-Said Eng. Res. J.* **2024**. [CrossRef]
112. Ren, H.; Zhou, L.; Gao, Y. Policy Tourism and Economic Collaboration Among Local Governments: A Nonparametric Matching Model. *Public Perform. Manag. Rev.* **2024**, *47*, 476–504. [CrossRef]
113. Gopalakrishnan, V.; Iyengar, S.P.; Madaan, A.; Rastogi, R.; Sengamedu, S. Matching Product Titles Using Web-Based Enrichment. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, 29 October–2 November 2012; pp. 605–614.
114. Akritidis, L.; Fevgas, A.; Bozanis, P.; Makris, C. A Self-Verifying Clustering Approach to Unsupervised Matching of Product Titles. *Artif. Intell. Rev.* **2020**, *53*, 4777–4820. [CrossRef]
115. De Bakker, M.; Frasincar, F.; Vandic, D. A Hybrid Model Words-Driven Approach for Web Product Duplicate Detection. In Proceedings of the 25th International Conference on Advanced Information Systems Engineering, Valencia, Spain, 17–21 June 2013; pp. 149–161.
116. Zheng, K.; Li, Z. An Image-Text Matching Method for Multi-Modal Robots. *J. Organ. End User Comput.* **2024**, *36*, 1–21. [CrossRef]
117. Song, Y.; Wang, M.; Gao, W. Method for Retrieving Digital Agricultural Text Information Based on Local Matching. *Symmetry* **2020**, *12*, 1103. [CrossRef]
118. Xu, B.; Huang, S.; Sha, C.; Wang, H. MAF: A General Matching and Alignment Framework for Multimodal Named Entity Recognition. In Proceedings of the 15th ACM International Conference on Web Search and Data Mining, Tempe, AZ, USA, 21–25 February 2022; pp. 1215–1223.
119. Gong, J.; Fang, X.; Peng, J.; Zhao, Y.; Zhao, J.; Wang, C.; Li, Y.; Zhang, J.; Drew, S. MORE: Toward Improving Author Name Disambiguation in Academic Knowledge Graphs. *Int. J. Mach. Learn. Cybern.* **2024**, *15*, 37–50. [CrossRef]
120. Arifoğlu, D. Historical Document Analysis Based on Word Matching. 2011. Available online: https://www.proquest.com/openview/b2c216ab3f6a907e7ad65bbe855fa8cd/1?pq-origsite=gscholar&cbl=2026366&diss=y (accessed on 23 May 2024).
121. Li, Y. Unlocking Context Constraints of Llms: Enhancing Context Efficiency of Llms with Self-Information-Based Content Filtering. *arXiv* **2023**, arXiv:2304.12102.