Journal of University Studies for inclusive Research
Vol.1, Issue 1 (2020), 1-16
USRIJ Pvt. Ltd.

A comparative analysis of searching algorithms

Dr: Ali Ahmed Alsalmi

University of Jeddah – Saudi Arabia

Email: aa.salmi@gmail.com

Abstract

Due to the growth of computer science applications in many domains, there is are many concerning issues in algorithms. The development of algorithms received huge interest in research community. In this paper, we present a review of some algorithms used in searching. These include greedy algorithm, dynamic programming algorithm, brute force algorithm, divide and conquer algorithm, and branch and bound algorithm. We compare between them and finally highlight a set of recommendations for future studies in the light of current issues. Despite an in-depth quantity of work devoted to this region of studies, there is a lack of an up-to-date survey inside the field. This paper pursuits to cope with this trouble with examine this is cantered on an evaluation of the literature with a focal point on recent methods that are not covered in previous surveys. It is believed that this work could be a precious reference for researchers of lexical semantics and significantly aid the studies activities in this field.

الملخص بالعربية

بسبب نمو تطبيقات علوم الكمبيوتر في العديد من المجالات، هناك العديد من المسائل المتعلقة بالخوارزميات. وتلقت تطوير الخوارزميات اهتماما كبيرا في مجتمع البحوث. في هذه الورقة العلمية، نقدم مراجعة لبعض الخوارزميات المستخدمة في البحث. وتشمل الخوارزمية الجشعة، خوارزمية البرمجة الديناميكية ، خوارزمية القوة الغاشمة ، خوارزمية التفريق والجمع، والخوارزمية الفرعية والتجميعية. سنقوم بالمقارنة بينهما ونسلط الضوء في النهاية على مجموعة من التوصيات للدراسات المستقبلية في ضوء القضايا الحالية. على الرغم من وجود كمية متعمقة من العمل المكرس لهذا المجال من الدراسات ، إلا أن هناك نقصًا في إجراء مسح حديث داخل هذا المجال. تسعى هذه الورقة إلى التغلب على هذه المشكلة حيث يتم تقييم هذا البحث على تقييم للأدبيات مع التركيز على الأساليب الحديثة التي لم يتم تغطيتها في الدراسات السابقة. ويعتقد أن هذا العمل يمكن أن يكون إشارة ثمينة للباحثين وتساعد بشكل كبير الباحثين في هذا المجال.

1. Introduction

There is a growing necessity to provide efficient algorithms for the type of search and decision problems, considered by concurrently determining the target object. There are several algorithms used for search purpose, and they should be compared and evaluated. This paper presents a comparison between some algorithmic approaches. Though exhaustive search is theoretically simple and often operative, such an approach to problem solving is occasionally considered unsophisticated. This may be a bequest of the fact that computer science focused for decades on fine-tuning highly efficient polynomial-time algorithms [1].

An updated assessment of the literature is presently unavailable. Although some have tried to cope with this trouble a precis of these researches is presently lacking. Third, we identify an opening among the studies of lexical semantics in widespread and domains [2]. On the one hand, an increasing number of new methods were proposed for popular purposes; however, most effective a very small amount of these had been adapted to any domain. We believe that both communities can gain advantage considerably by distributing the knowledge and lessons learnt [3].

This study surveys the literature on ways for measuring lexical semantic, specializing in recent research that have now not been covered inside the preceding surveys, and connecting studies from both well-known and domains.

The term semantics is used otherwise in distinctive contexts. For the objectives of this paper, we describe a semantic interpretation as one that displays the implies of the text as its miles understood by means of a language speaker [4].

The conventional formal semantics is an inheritor of Montague grammar and therefore is ontologically as an alternative unsophisticated. A rich ontology of types is but vital for the analysis of natural language lexicons [5].

This paper is organized as follows. Section two provides a review of related studies that have focused on similar algorithms, section three is a methodology of this paper, section 4 is a discussion of the reviewed algorithms, and section five and six provide a summary and recommendations for future research.

Research importance

The significance of data with zero error is also affects the role of model understanding, meanwhile any effort to repair and understand the result of the model must be stranded on the suppose that the data is sufficiently clean. All these remarks point to the rule that there is a necessity to advance data to the top of citizen in ML pipelines [2].

Minor data errors might lead to gradual regression regarding model performance over a period. Consequently, it is powerful to closure data errors initially, earlier they spread through these multifaceted loops and defect more of the pipeline's situation [5].

Research Problem

There is a small amount of consideration in the correspondingly significant problem of auditing the quality of data taken by machine learning while an extreme focus of machine learning research has been directed to improve the accuracy and efficiency of training algorithms.

The significance of this issue is difficult to be expressed in terms of errors in the input data especially for production pipelines. These errors may cancel any usefulness on speed and accuracy for training. This supposing arguments to data centric approach for machine learning that processes training and processing data as a significant creative advantage, on parity with the algorithm and structure used for knowledge [6].

We are focusing on the problem of confirming the input data given to ML pipelines. Regardless the ML algorithms applied, data errors can poorly influence the quality of the produced models. Besides, it is regularly the situation that the estimates from the produced model are recorded and utilized to produce additional data for training [4].

It must to repentantly audit and confirm data through the several phases of the pipeline [4]. To validate data, we don't have a new problem nor exceptional case to ML, thus we take solutions from associated domains. Nevertheless, we claim that the problem gains exceptional issues in the framework of ML and therefore we want to reconsideration current solutions.

Research objectives

Although of an excessive process of machine learning studies has engrossed on refining the accuracy and efficiency of training algorithms, there is little consideration of the similarly significant problem of tracking the data quality inputted to machine learning. The essence of this problem is difficult to negotiate faults in the input data can invalidate the usefulness of accuracy and speed. This supposes indicates to a data-centric method to machine learning that deals training and helping data as a significant construction advantage, on parity with the algorithm utilized for learning.

Research Questions

In this research, we aim to answer the following questions:

1. What is the mean of data validation?
2. What is the purpose of data validation?
3. What are the approaches of data validation?

4. How to improve data validation system?

Lexicon is statistical, semantics is universal

A semantic idea or semantics describes the structure of meaning. The relation between expressions and meanings as well as among the expressions themselves is a site of lexical semantics. Two factors of the relation between phrase and it's that means play a vital role within the definition of the problem remember of semantics and lexical semantics: (a) semasiological factor, i.e. What an expression can mean, and (b) onomasiology element, i.e. Which expressions can represent positive meaning? These two elements are necessarily carefully related, especially in cross-linguistic research. Lexical semantics is essentially worried with the way that language lexicalizes the arena and right here the semasiological element takes over [7].

1. All parts of a composed expression would possibly make a contribution actively to the very last meaning.

2. The that means of atomic expressions (lexical items) may be decomposed into semantic systems that take part in the compositional process.

Lexical Semantics is a kind of computational linguistics, which investigates relationships between two words and how phrase provides to the sentence that it is living in. The required data for the semantics of the word may be obtained from a textual content corpus, for example, the relationship among "wood" and "carpenter" may be decided by looking at the context that these phrases seem in several report instances [8].

WordNet

Lexicon is a concept which corresponds to phrase lists with some subset of information, such as parts of speech. WordNet is an assignment first emerged in 1990, which is a big lexicon with properties of a semantic internet. The WordNet's semantic net includes the following relations; synonymy, polysemy, metonymy, hyponymy/hypernymy, meronymy, and antonymy in addition to the brand-new additions such as organizations of similar words, links between derivationally and semantically related noun/verb pairs [8].

Relations in WordNet

Synonymy corresponds to the relationship of synonyms, that are exclusive approaches of expressing associated concepts along with "gathering" or "meeting". However, they are rarely substitutable, meaning we cannot use them interchangeably in all contexts [9].

Homonymy corresponds to the phrases that have same syntax, however unrelated meanings. On the opposite hand, polysemy corresponds to the words that have same syntax and have associated meanings [10].

Metonymy is the use of a thing of a word to describe something similar. Using the phrase "Ankara" in information to indicate "the Grand National Assembly of Turkey" might be an instance [11].

Hyponymy/hypernymy corresponds to the "is-a" relationship. For example, polygamy (having more than one spouse at a time) is a hypernym of polygyny (having more than one wife at a time). And polygyny is hyponym of polygamy [12].

Meronymy corresponds to the relation between phrases in which one is a part of another, along with the connection between "bird" and "wing" [13].

Semantic properties and semantic features

Speakers of a language proportion a primary vocabulary. This mental shop of phrases and morphemes and their meanings is referred to as a lexicon. Because we are not talking of a written dictionary or a lexicon within the ordinary sense, this shared know-how which is stored somewhere inside the mind is known as the intellectual lexicon. It incorporates the phonological form of a word (pronunciation and stress) and the minimal, shared which means definition which permits audio system of the same language to speak [14].

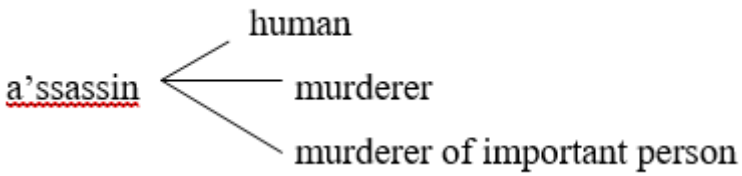

Figure 1: example of phonological form of a word

The first stage within the records of lexical semantics runs from transcription the records. Its dominant feature is the ancient orientation of lexical semantic research; its main difficulty lies with modifications of phrase which means—the identification, classification, and clarification of semantic changes [15].

Along these strains of research, a wealth of theoretical proposals and empirical descriptions was produced. Most of this has by using now sunk into oblivion, however. In realistic terms, the older monographs could be absent from all but the oldest and the most important educational libraries, and where they're available, there's possibly to be a language barrier: maximum of the applicable works are written in German or French, languages that aren't handy to all. As a result, a number of the topics that were investigated thoroughly within the older way of life are later being reinvented in preference to rediscovered; we can see proof of this in later chapters [10].

Lexical semantics as an educational area in its own right originated within the early nineteenth century, but that doesn't mean that topics of word which means had now not been mentioned earlier. Three traditions are applicable: the culture of speculative etymology, the coaching of rhetoric, and the compilation of dictionaries. Let us briefly see what every of the 3 traditions involves, and the way they play a role inside the start of lexical semantics as an educational enterprise [16].

2. Literature Survey

Greedy algorithm

Greedy concept is based on process the input in some direction, short-sightedly creation irreversible results [6]. A greedy algorithm follows the problem-solving heuristic of creation the close by optimum choice at each phase with the aim of finding a global optimal. In numerous problems, a greedy approach does not in overall yield an optimal solution, but however a greedy heuristic may produce locally optimal solutions that near a global optimal solution in a realistic time. In a greedy algorithm, the optimal solution is made up one part at a time. At each phase the best reasonable candidate is selected as the following part of the solution. There is no backtracking [17].

It creates a solution through an order of phases. Each phase increases a partly created solution so far, till a whole solution to the problem is gotten. On each phase, the choice finished must be reasonable. It has to content the problem's restrictions, or close by optimal, it has to be the top local choice amongst all reasonable choices obtainable on that phase or unchangeable. Once complete, it cannot be transformed on subsequent phases of the problem [6]. If a greedy algorithm can be confirmed to produce the global optimum for a specified problem class, it naturally suits the process of choice because it is quicker than other optimization approaches like dynamic programming. Instances of such greedy algorithms are Kruskal's algorithm and Prim's algorithm for finding smallest spanning trees, and the algorithm for finding optimal Huffman trees [6].

Brute force algorithm

Brute force is a forthright method of problem solving, typically straight based on the problem's declaration and descriptions of the concepts included. The straightforward idea behind it is that the pattern and text are associated character by character. In situation if a character is not corresponding, then the pattern is erased on location to the right and the judgment is repetitive till a match is found or the end of the text is gotten.

One of the eldest methods to problem solving with the aid of computers is brute-force list and search: produce and examine all data settings in a big state space that is certain to contain the anticipated solutions, and you are bound to succeed— if you can pause long enough [1].

In spite of the fact that once in a while a smart or effective algorithm, the brute-force approach ought not to be ignored as a significant algorithm structure technique. In contrast to a portion of different systems, brute-force is material to a wide assortment of issues. For some significant issues (e.g., Arranging, looking, string coordinating), the brute-force approach yields a sensible algorithm of probably some common sense incentive with no restriction on occasion size Even if excessively wasteful when all is said in done, a brute-force algorithm can in any case be helpful for settling little size examples of an issue. A brute-force algorithm can serve a significant hypothetical or instructive reason.

Divide-and-conquer approach

Divide-and-conquer approach breaks up a problem into autonomous sub-problems and resolve each sub-problem, then merge solutions to sub-problems to create a solution to the whole problem. Divide and conquer approach defines a group of sub-problems (typically, only a polynomial number of sub-problems). Its solution to original problem can be found from sub-problems. Natural ordering of sub-problems from the smallest to the largest that allows defining a solution to a sub-problem from solutions to smaller sub-problems [2].

The divide and conquer algorithms are applicant problem for the multicore programming since divide and conquer algorithm splits up a problem into sub-problems which can be resolved by dividing the sub-problems amongst the diverse cores and parallel resolve them. A widespread series of divide and conquer algorithm has been parallelized [2]. Techniques of divide and conquer approach are binary choice, which is designed for weighted interval scheduling, multiway choice that is based on segmented least squares, adding a new variable for knapsack problem, intervals RNA secondary structure, and Top-down and bottom-up dynamic programming [2].

Quicksort is one of the furthermost used sorting algorithms. The performance quicksort variedly varies upon the type of the input. The bad situation occurs when the input is sorted in reverse order. The applications of quicksort are routing algorithm, the scheduling processes etc. It uses a separating technique which puts the hinge values at its right location in each repetition. The quicksort can be parallelized by parallel sorting the array and then merging the sorted array parallel [2].

Branch-and-bound (B&B) algorithm

A bounded search repeated greedy algorithm for the spread variation arrangement problem. The branch and bound algorithm is an optimum feature selection method that is famous for its computational efficiency. Nevertheless, when the dimensionality of the original feature space is huge, the performance essential by the branch and bound algorithm converts very extreme. If the optimality of the algorithm can be cooperated, the search time can be significantly minimized by engaging the look-ahead search policy to remove numerous results estimated to be suboptimal early in the search [3].

The branch-and-bound (B&B) algorithm has been used effectively to look for precise solutions for a widespread array of optimization problems. B&B applies a tree search approach to covertly compute all probable solutions to a specified problem, concerning clipping guidelines to remove sections of the search space that cannot produce a better solution. There are three algorithmic items in B&B that can be quantified by the user to fine tune the behaviour of the algorithm. These items are the search approach, the branching approach, and the clipping rules [4].

Branching strategies, the choice of branching strategy regulates how children are created from a sub-problem. Splitting approaches can be branded into two collections: dual splitting approaches and non-dual, or extensive, splitting approaches. Moreover, due to the occurrence of integer programming problems, there is an excess of literature keen to splitting approaches in integer programming [4].

Dynamic programming (DP)

Dynamic programming breaks up a problem into a sequence of overlying sub-problems and combine solutions to slighter sub-problems to create a solution to the big sub-problem. The dynamic programming approach can be joint with a very effective and applied pruning approach so that very big search places can be held. Second, the dynamic programming approach has twisted out to be really exile in familiarising to new needs [18].

Some well-known dynamic programming algorithms include Avidan–Shamir for seam carving, Unix diff for comparing two files, Viterbi for hidden Markov models, De Boor for evaluating spline curves, Bellman–Ford–Moore for shortest path, Knuth–Plass for word wrapping text, Cocke–Kasami–Younger for parsing context-free grammar, Needleman–Wunsch/Smith–Waterman for sequence alignment [2].

Search approaches based on dynamic programming (DP) are presently being used effectively for a huge number of speech recognition tasks, ranging from digit string recognition through medium-size vocabulary recognition using seriously controlled grammars to big vocabulary continuous speech recognition with almost unrestricted speech input [19].

Table 1 shows a comparison between the algorithms in terms of complexity and the ability to parallelized. As shown, greedy algorithm and brute force are very costly, and they cannot be parallelized. While others are computationally efficient and can be parallelized [20].

Table 1: A comparison of searching algorithms in terms of complexity and parallelized

Algorithm	Complexity	Parallelized
Greedy	Very costly	Not applicable
Brute force	Very costly	Not applicable
Dynamic programming	Efficient	Applicable
Divide and conquer	Relatively efficient	Applicable
Branch and bound	Computational efficiency	Applicable

3. Methodology

There are many search methods, but there are some efficient algorithms while there are more complex and slow ones [21]. In this paper, we follow an analytical approach to review the literature about search algorithms namely greedy algorithm, dynamic programming algorithm, brute force algorithm, divide and conquer algorithm, and branch and bound algorithm.

4. Discussion

Greedy algorithm is appropriate to optimization problems only [22]. Greedy algorithms regularly (but not always) cannot find the generally optimal solution, because they typically do not work thoroughly on all the data [23]. They can make assurances to convinced choices too initial which avoid them from finding the finest overall solution later [17].

The strengths of brute force approach include wide applicability, simplicity, yields reasonable algorithms for some important problems (e.g., Matrix multiplication, sorting, searching, string matching) [24]. On the other hand, the weaknesses rarely yielding efficient algorithms, some brute-force algorithms are unacceptably slow, and not as constructive as some other design techniques [25].

The easiest approach for search problem is the Brute Force-Algorithm**,** which is also recognized as Naive algorithm**.** The algorithm does not need any pre-processing. Use of the basic BB algorithm is not recommended for a large-dimensional database [3].

When the dimensionality of the original feature space is huge, the performance essential by the branch and bound algorithm converts very extreme. If a greedy algorithm can be confirmed to produce the global optimum for a specified problem class, it naturally suits the process of choice because it is quicker than other optimization approaches like dynamic programming.

5. Gaps in the study

From this, we can accomplish that nothing of these methods work consistently in a method that would measure to more complex problems. Consequently, we suppose that original model and methodology want to be established to demand to solve real–world consecutive decision problems, which are becoming progressively problematic.

The main drawback of the brute-force method is that, for numerous real-world problems, the number of usual candidates is excessively large. For example, if we look for the divisors of a number, the number of candidates verified will be the specified number *n*. So if *n* has sixteen decimal digits, say, the search will involve completing at least 10^{15} computer commands, which will take some days on a usual PC. This vertical growing in the number of candidates, as the scope of the data growths, happens in all kinds of problems.

5. Conclusion

In this paper, we review some algorithms used in search function. Some algorithms are very efficient such as dynamic programming, but others become slow when data increases such as greedy algorithm, the core idea of some algorithms is dividing a problem into subproblems like divide and conquer and branch and bound. We can conclude that there is a trade-off between algorithms in terms of complexity and size.

Some algorithms can be parallelized to find the higher and lower angles in analogous. In future research, some algorithms might be combined to achieve an effective search function. Another direction of research is to add an optimal random number to dynamic programming to reach better results.

References

[1] Hua, W., Wang, Z., Wang, H., Zheng, K., & Zhou, X. (2015, April). Short text understanding through lexical-semantic analysis. In *2015 IEEE 31st International Conference on Data Engineering* (pp. 495-506). IEEE.

[2] Batet, M., Sanchez, D., and Valls, A.(2011). An ontology-based measure to compute semantic ´similarity in biomedicine. Journal of Biomedical Informatics 44(1), 118–25.

[3] Chang, F. S., Wu, J. S., Lee, C. N., & Shen, H. C. (2014). Greedy-search-based multi-objective genetic algorithm for emergency logistics scheduling. *Expert Systems with Applications*, *41*(6), 2947-2956.

[4] Dash, S. P., & Dora, K. P. K. (2013). *Analysing the Performance of Divide-and-Conquer Algorithms on Multicore Processors* (Doctoral dissertation).

[5] Egozi, O., Markovitch, S., and Gabrilovich, E. (2011). Concept-based information retrieval using explicit semantic analysis. ACM Transactions of Information Systems 29(2), 8:1–8:34

[6] Elsherif, M., Wheeldon, L. R., & Frisson, S. (2019). Lexical-semantic precision for word recognition in skilled readers.

[7] Nievergelt, J. (2000, November). Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *International Conference on Current Trends in Theory and Practice of Computer Science* (pp. 18-35). Springer, Berlin, Heidelberg.

[8] Omri Abend and Ari Rappoport. (2013a). UCCA: A semantic-based grammatical annotation scheme. In *Proc. of IWCS*. pages 1–12.

[9] Schneider, N., Danchik, E., Dyer, C., & Smith, N. A. (2014). Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *Transactions of the Association for Computational Linguistics*, *2*, 193-206.

[10] Shu Cai and Kevin Knight. (2013). Smatch: an evaluation metric for semantic feature structures. In *Proc. of ACL*. pages 748–752.

[11] Song, B. C., Kim, M. J., & Ra, J. B. (2001). A fast multiresolution feature matching algorithm for exhaustive search in large image databases. *IEEE Transactions on Circuits and Systems for Video Technology*, *11*(5), 673-678.

[12] van Dam, W. O., van Dijk, M., Bekkering, H., & Rueschemeyer, S. A. (2012). Flexibility in embodied lexical-semantic representations. *Human brain mapping*, *33*(10), 2322-2333.

[13] Yih, W. T., Chang, M. W., Meek, C., & Pastusiak, A. (2013, August). Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1744-1753).

[14] Yoav Artzi and Luke Zettlemoyer. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL* 1:49–62.

[15] Ney, H., & Ortmanns, S. (2000). Progress in dynamic programming search for LVCSR. *Proceedings of the IEEE*, *88*(8), 1224-1240.

[16] Nobre, A. D. P., & Salles, J. F. D. (2016). Lexical-semantic processing and reading: Relations between semantic priming, visual word recognition and reading comprehension. *Educational Psychology*, *36*(4), 753-770.

[17] Jiang, D. R., Pham, T. V., Powell, W. B., Salas, D. F., & Scott, W. R. (2014, December). A comparison of approximate dynamic programming techniques on benchmark energy storage problems: Does anything work?. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)* (pp. 1-8). IEEE.

[18] Lieber, R. (2016). Compounding in the lexical semantic framework. *The semantics of compounding*, 38-53.

[19] Nakariyakul, S. (2009). A Review of Suboptimal Branch and Bound Algorithms. In *International Conference on Computer Engineering and Applications* (Vol. 2, No. 1, pp. 492-496).

[20] Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2009). Efficient subwindow search: A branch and bound framework for object localization. *IEEE transactions on pattern analysis and machine intelligence*, *31*(12), 2129-2142.

[21] Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques* (pp. 234-243). Springer, Berlin, Heidelberg.

[22] Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, *19*, 79-102.

[23] Narendra, P. M., & Fukunaga, K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, (9), 917-922.

[24] Ney, H., & Ortmanns, S. (1999). Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, *16*(5), 64-83.

[25] Bassac, C., Mery, B., and Retoré, C. (2010). Towards a type-theoretical account of lexical semantics. *Journal of Logic, Language and Information*, 19(2):229–245. (Cited on pages 11, 23, 40, 41, and 42.)