PROCEEDINGS OF

2019 INTERNATIONAL CONFERENCE ON INFORMATION & COMMUNICATION TECHNOLOGY AND SYSTEMS (ICTS)

Surabaya, July 18th, 2019

(ISBN: 978-1-7281-2133-8)

Organized by
Department of Informatics
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember (ITS)
Surabaya, Indonesia

Use Case Diagram Similarity Measurement: A New Approach

Reza Fauzan
Informatics Department
Institut Teknologi Sepuluh Nopember,
Politeknik Negeri Banjarmasin
Surabaya, Indonesia
reza.fauzan@poliban.ac.id

Daniel Siahaan
Informatics Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
daniel@if.its.ac.id

Siti Rochimah
Informatics Department
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
siti@if.its.ac.id

Evi Triandini
Information Systems
STMIK STIKOM Bali
Bali, Indonesia
evi@stikom-bali.ac.id

***Abstract*—Building a UML diagram from scratch usually requires a lot of resources of software developer. UML reuse is the solution for speeding up software development process. Previous researches utilized diagram similarity measurement for finding reusable diagrams. Many researchers focus on class and sequence diagrams. This research introduces a new approach for measuring use case diagram similarity. This approach combines structural information and lexical information for measuring the similarity of two diagrams. It used to measure structural similarity, i.e. relationship among and between actors and use cases. It also uses Wordnet, WuPalmer, and Cosine Similarity to measure the lexical information semantically, i.e. and text extracted from actors and use cases. The initial result shows that the semantic and structural information could be used as parameters to measure similarity between two use case diagrams.**

Keywords—semantic similarity, structural similarity, UML similarity, use case similarity

I. INTRODUCTION

Unified Modeling Language (UML) is a standard displaying dialect for determining, recording, and building a software product [1]. UML can likewise be alluded to as a standard dialect in demonstrating that is frequently utilized by programming engineers long back [2], [3]. UML advancement has a few issues. One issue that is regularly discovered when making UML is that it requires a long investment in the event that it is required to make it from the earliest starting point [4].

At that point, reusing UML diagram shows up as an answer for the issue. Reusing UML diagram should be possible utilizing programming that was created previously, not from the earliest starting point [6]. Reusing UML diagram can help quicken the product advancement process. What's more, reusing UML can decrease the expenses and dangers utilized [5].

There are four steps of UML reuse, i.e. portrayal, recovery, adjustment and consolidation [6], [7]. During the portrayal, a model of the new UML artefact is introduced. During the retrieval, a UML artefact is like the inquiry, and the adjustment cost is insignificant which is chosen from the segment library or repository. During adaptation, the UML artefact is modified to obtain a new component. At last, the new UML artefact is joined or incorporated into the repository.

There are some researches that reuse UML artefact. The artefacts are class diagram and sequence diagram. Previous researches [8]–[12] measure the similarity between two class diagrams. They gathered lexical information from a class diagram. Then they measured the lexical information by semantic similarity. Then, they measured the diagram structure by the neighbor information.

The other research [9], [13], [14] measured the similarity between two sequence diagrams. Their measurements are like sequence diagram similarity measurement. They measure lexical information as a semantic similarity. And they measured the structure by communication between the objects or they converted sequence diagram into a graph.

Siahaan [13] indicates that the semantic and structural information could be used as parameters to measure the design diagram similarity. It proposes the use case diagram similarity measurement using semantic and structural similarity. The semantic similarity was measured by the lexical information. Siahaan called property as the collection of lexical information in use case diagram. On the other hand, the structural similarity was measured by the relationship between actor and actor, actor and use case, and use case and use case.

There are two researches [15], [16] that examines the similarity of use case diagrams. Blok and Cybulski [16] performed use case diagram clustering related to the use of semantic words for the reuse process. They used the contents of each use case as a single meaning. The results of this study indicate that using semantic similarity can provide a higher similarity value. Storrle [15] measured the similarity of many diagrams including the use case diagram. The research aims to detect cloning from UML diagrams. It took the lexical information of actors and use cases in use case diagram. The contribution of our research is to adapt our previous research in class diagram [17] and sequence diagram [13]. We introduced a new approach in use case diagram similarity measurement. The approach changes the syntactic similarity measurement from previous work [15] into semantic similarity measurement in lexical information comparison. The approach also combines the semantic similarity with structural similarity of the two diagrams to improve the accuracy of the use case diagram similarity. The structural similarity is based on the combination of lexical information from actor's name, use case's name, relationship's name and relationship's type.

978-1-7281-2133-8/19/$31.00 ©2019 IEEE

II. Research Method

A. Diagram Preprocessing

Diagram preprocessing helped this research to get the lexical information and structural information. In the beginning, UML use case diagram was built by a UML designing tool. Then, the tool converted the design into XMI-format. XMI consist of all information in the diagram and showed structured.

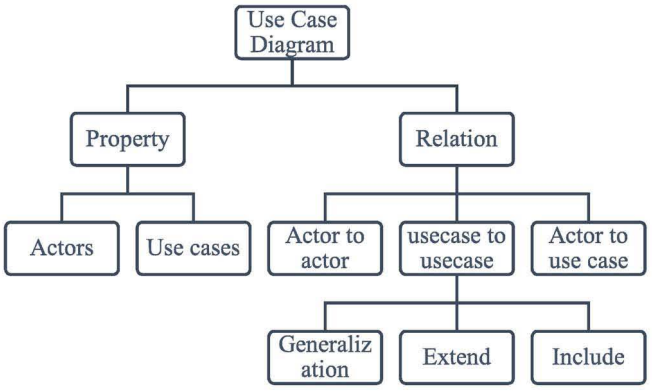

Fig.1. Use Case Diagram Metadata

XMI would be extracted into a use case diagram metadata. The metadata in the use case diagram is actor, use case, and the relations [18]. This paper divided metadata become property information and relation information. Fig. 1 showed the metadata which used in this paper. The property information consists of actor's name and use case's name. the relation information consists of the relation between actor and actor, the relation between actor and use case, and the relation between use case and use case.

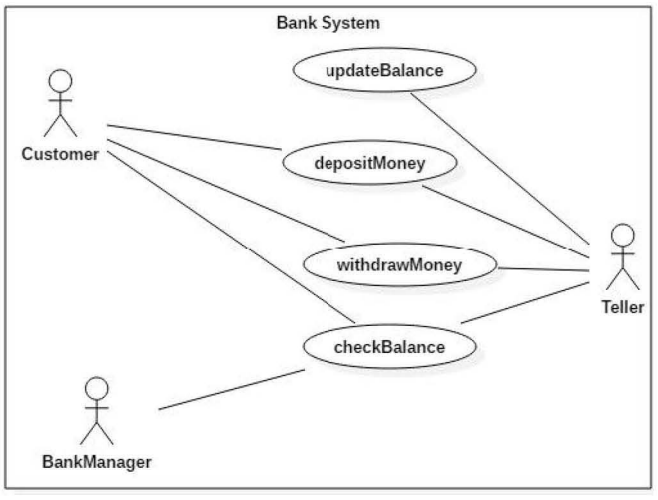

Fig.2. Use Case Diagram 1 (UCD1)

Based on Fig. 2, we mapped the metadata information. The result of metadata retrieved are as follows:

Property Information:

→ *actor1: customer*

→ *actor2: bank manager*

→ *actor3: teller*

→ *usecase1: update balance*

→ *usecase2: deposit money*

→ *usecase3: withdraw money*

→ *usecase4: check balance*

Relation Information:

→ *rel1: (actor: customer, uc: deposit money)*

→ *rel2: (actor: teller, uc: update balance)*

→ . . .

→ *Rel8: (actor: bank manager, uc: check balance)*

Property information has 3 actors and 4 use cases. Relation information has 8 relations. The relation consists of no relation between actor and actor, 8 relations between actor and use case, no relations between use case and use case.

B. Words Similarity

The method of measuring similarity used cosine similarity for all metadata [19]. This paper utilized Wu Palmer and Wordnet to discover the similarity between lexical information semantically [20]–[22]. Wu Palmer can calculate the closeness between two words in Wordnet. As we know, wordnet is knowledge in graph form. So that Wu Palmer is one of the right algorithms because it can measure the closeness between two words based on depth in the graph. The similarity range value is 0 to 1. 0 means there is no similarity between them. 1 means an exact similarity between two words.

In use case diagram, we sometimes cannot measure the semantic similarity only between two words. We need to measure between two pair words, for instance between two use case. Every use case at least includes two words for example "withdraw cash". So, we combine Wu Palmer and cosine similarity [23] to measure the similarity.

C. Finding Optimal Value

This study calculates all comparisons of relations from two diagrams then made into a matrix. So, every relation of the two diagrams will be fair to each other. For example, relations between actors and use cases will not be compared with the relations between actors and actors or relations between use cases and use cases. Based on the many comparisons that appear in the matrix, this study requires an algorithm to find the most optimal value. Some previous studies [11], [17] used greedy in finding the most optimal value in the comparison because it was simple and fast. Therefore, this study also uses greedy algorithm in finding optimal values.

D. Diagram Similarity Calculation

Based on Figure 1, the metadata consists of property information (*propSim*) and relation information (*relSim*). Equation 1 shows the similarity measurement between two UML use case diagrams (*ucdSim*), i.e. d_1 and d_2.

$$ucdSim = w_{prop} \times propSim(d_1, d_2) + w_{struc} \times strucSim(d_1, d_2) \quad (1)$$

where w_{prop} is the weight of the property similarity from d_1 and d_2. And w_{rel} is the weight of relation similarity from d_1 and d_2.

They are arbitrary weight. The next phase is to measure property similarity (*propSim*) from d_1 and d_2. This calculation is portrayed in Equation 2.

$$propSim = w_{ac} \times acSim(d_1, d_2) + w_{uc} \times ucSim(d_1, d_2) \quad (2)$$

where w_{ac} is the weight of the actor similarity from d_1 and d_2. And w_{rel} is the weight of relation similarity from d_1 and d_2. The weights are arbitrary weight. Then, Equation 3 calculates the actor similarity (*acSim*) between two use case diagrams.

$$acSim(d_1, d_2) = \frac{CosineSim(ac_i, ac_j)}{|AC_1|,|AC_2|} \quad (3)$$

where AC_1 and AC_2 are a collection of actor's lexical information from two use case diagrams (ac_i, ac_j). Next, Equation 2 showed the use case similarity (*ucSim*). This calculation is portrayed in Equation 4.

$$ucSim(d_1, d_2) = \frac{CosineSim(uc_i, uc_j)}{|UC_1|,|UC_2|} \quad (4)$$

where UC_1 and UC_2 are a collection of use case's lexical information from two use case diagrams (uc_i, uc_j). Next, Equation 1 showed the relation similarity (*relSim*). This calculation is portrayed in Equation 5.

$$relSim(d_1, d_2) = w_{aa} \times aaSim(d_1, d_2) + w_{au} \times auSim(d_1, d_2) + w_{uu} \times uuSim(d_1, d_2) \quad (5)$$

where w_{aa} is the weight of the relation similarity between actor and actor from d_1 and d_2. w_{au} is the weight of the relation similarity between actor and use case from d_1 and d_2. And w_{uu} is the weight of the relation similarity between use case and use case from d_1 and d_2. They are arbitrary weight. In this case, we used equal weight. The next phase is to measure the relation similarity between actor and actor (*aaSim*) from d_1 and d_2. This calculation is portrayed in Equation 6.

$$aaSim(d_1, d_2) = \frac{Max(\sum_{i,j=1}^{Max(|AA_1|,|AA_2|)} daaSim(aa_i, aa_j))}{|AA_1|,|AA_2|} \quad (6)$$

where AA_1 and AA_2 are a collection of the relation similarity between actor and actor from two use case diagrams (aa_i, aa_j). To make it detail, this relation consists of source actor (*srcA*) and target actor (*tgtA*). So, an advance calculation is needed to measure the detail of relation similarity between actor and actor (*daaSim*). This calculation is portrayed in Equation 7.

$$daaSim(d_1, d_2) = w_{srcA} \times WuP(srcA_1, srcA_2) + w_{tgtA} \times WuP(tgtA_1, tgtA_2) \quad (7)$$

where w_{srcA} and w_{tgtA} are arbitrary weight assign to source actor and target actor, respectively. The relation similarity between actor and actor can be calculated from the lexical of each component in the relation. Next, Equation 5 raises the relation similarity between actor and use case (*auSim*). This calculation is portrayed in Equation 8.

$$auSim(d_1, d_2) = \frac{Max(\sum_{i,j=1}^{Max(|AU_1|,|AU_2|)} dauSim(au_i, au_j))}{|AU_1|,|AU_2|} \quad (8)$$

where AU_1 and AU_2 are a collection of the relation similarity between actor and use case from two use case diagrams (au_i, au_j). To make it detail, this relation consists of actor (*a*) as a source and use case (*uca*) as a target. So, an advance calculation is needed to measure the detail of relation similarity between actor and use case (*dauSim*). This calculation is portrayed in Equation 9.

$$dauSim(d_1, d_2) = w_a \times WuP(a_1, a_2) + w_{uca} \times WuP(uca_1, uca_2) \quad (9)$$

where w_a and w_{uca} are arbitrary weight assign to actor and use case, respectively. The relation similarity between actor and use case can be calculated from the lexical of each component in the relation. Next, Equation 5 raises the relation similarity between use case and use case (*uuSim*). This calculation is portrayed in Equation 10.

$$uuSim(d_1, d_2) = \frac{Max(\sum_{i,j=1}^{Max(|UU_1|,|UU_2|)} duuSim(uu_i, uu_j))}{|UU_1|,|UU_2|} \quad (10)$$

where UU_1 and UU_2 are a collection of the relation similarity between use case and use case from two use case diagrams (uu_i, uu_j). To make it detail, this relation consists of use case source (*ucA*), relation type (*ty*), and use case target (*ucT*). So, an advance calculation is needed to measure the detail of relation similarity between use case and use case (*duuSim*). This calculation is portrayed in Equation 11.

$$duuSim(d_1, d_2) = w_{ucA} \times WuP(ucA_1, ucA_2) + w_{ty} \times WuP(ty_1, ty_2) + w_{ucT} \times WuP(ucT_1, ucT_2) \quad (11)$$

where w_{ucA}, w_{ty}, and w_{ucT} are arbitrary weight assign use case source, relation type, and use case target, respectively. The relation similarity between use case and use case can be calculated from the lexical of each component in the relation.

III. Empirical Result and Analysis

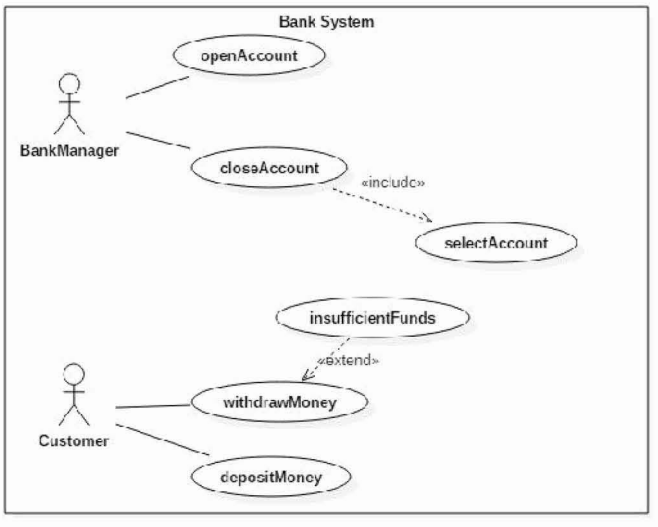

Fig.3. Use Case Diagram 2 (UCD2)

In this part, we used a simple case to showed that semantic and structural similarity is a good parameter in use case diagram similarity measurement. We calculate the similarity of use case diagram in Fig. 2 as UCD1 and use case diagram in Fig. 3 as UCD2. Both are use case diagram in bank transactions. But the lexical information and diagram's structure is different.

Based on Equation 3, we used cosine similarity to find the actor similarity between two use case diagrams. Table I shows the vector made and the similarity between the vector and actor in each diagram.

TABLE I. ACTOR'S VECTOR COMPARISON

Vector	bank	manager	teller	customer
UCD1	1	1	1	1
UCD2	1	1	0	1

The actor's lexical information from UCD1 are bank, manager, teller, and customer. Table I shows that all of them has the same meaning semantically with the vector. The actor's lexical information from UCD2 are bank, manager, and customer. Table I shows that one of the vectors has no similarity with them. The values shown in Table I become input in the formula of cosine similarity. The result is 0.866.

Then we do the same thing in the use case names according to Equation 4. The resulting vector is balance, money, update, deposit, check, withdraw, account, select, insufficient, close and open. UCD1 has balance, money, updates, deposits, checks and withdrawals. UCD2 has an account, select, insufficient, money, fund, deposit, close, open and withdraw. So that the similarity value of the use case name is 0.433.

After knowing the similarity of actors and use cases, we can find out the similarity values of properties of UCD1 and UCD2. Equation 2 is used to calculate this value with weights used 0.5 for actors and 0.5 for use cases. The resulting value is 0.649.

TABLE II. RELATION OF ACTOR AND USECASE SIMILARITY BETWEEN UCD1 AND UCD2

auSim	UCD21	UCD22	UCD23	UCD24
UCD11	0	0	0.5	1
UCD12	0	0	1	0.5
UCD13	0	0	0.5	0.5
UCD14	0.5	0.5	0	0
UCD15	0	0	0.0769	0.0769
UCD16	0	0	0.0769	0.5769
UCD17	0	0	0.5769	0.0769
UCD18	0	0	0.0769	0.0769

Based on Equation 5, we need to calculate three kinds of relation. First, we calculate the similarity the relation of actor and actor between two use case diagrams using Equation 6. But both of them don't have this kind of relation. It makes the similarity value in this relation is 0. Second, we calculate the similarity of the relation of actor and use case between two use case diagram using Equation 8. The calculation can be shown in Table II.

Based on Fig. 2, UCD 1 has eight relations of actor and use case. They are customer - deposit money (UCD11), customer - withdraw money (UCD12), customer - check balance (UCD13), bank manager - check balance (UCD14), teller - update balance (UCD15), teller - deposit money (UCD16), teller - withdraw money (UCD17) and teller - check balance (UCD18). Based on Fig. 3, UCD 2 has four relations of actor and use case. They are bank manager - open account (UCD21), bank manager - close account (UCD22), customer - withdraw money (UCD23) and customer - deposit money (UCD24). Equation 9 is used to calculate the similarity between the relation. For instance, the calculation's result between UCD11 and UCD 21 is 0. It was because there is no semantic similarity between actors (customer - bank manager) and also the semantic similarity between use cases (deposit money - open account). After we calculate the detail relation of actor and use case, we can find the optimal value of the matrix in Table II using the greedy algorithm as explained in Equation 8. So, the optimal value from the relation of actor and use case is 0.416. Third, we calculate the similarity the relation of use case and use case between two use case diagrams using Equation 10. UCD1 does not have this kind of relationship. But UCD2 has this kind of relationship. And the similarity value becomes 0. Finally, we combine these three types of relations using Equation 5 with the equivalent weight, which is one-third. The value generated from the similarity of the relation is 0.1375.

By knowing the value of similarity of property and relations, we can calculate the similarity between two use case diagrams using Equation 1. The weights used are 0.5 and 0.5 respectively. The similarity value generated from Equation 1 is 0.3935.

Based on the results, this study shows all parameters that could affect the similarity of diagrams in use cases. All metadata can be completely displayed. So, structural and semantic similarity measurement are good parameters in calculating UML diagrams [13]. And we need to combine both of them to assess the similarity between two use case diagrams. On the other hand, we need an adaptive weight to measure the similarity based on the availability. Both use case diagrams do not have actor relations with actors. Although the two diagrams are exactly the same, the total weight of the structural similarity will not be more than two-thirds of the maximum. This can be seen from the low similarity value of the relations between UCD1 and UCD2.

IV. CONCLUSION

This paper introduces the method to measure the similarity between two models designed as use case diagrams. The algorithm uses word similarity methods to measure the word similarity between elements of use case diagram, and cosine similarity to find the similarity values between the two use case diagrams. The proposed method consists of two parts, namely the semantic similarity of the property and the structural similarity of relationship information. The initial investigation of this paper show all parameters could determine the similarity of the two use case diagrams. Every detail information of use case diagram can determine to use case diagram similarity.

Further research to determine the use of larger data sets should be carried out. This research should determine a set of weights that can achieve the most accurate measurement. And also we need to consider related adaptive weights based on the availability of both diagrams. Therefore, it is necessary to search for alternative algorithms that are more precise than cosine to find the best pairs of similarities. And this study has not measured the similarity between relations types. For example, what is the comparative value between include and extend relations, include and generalization, and extend and generalization.

ACKNOWLEDGMENT

This research is in cooperation between Institut Teknologi Sepuluh Nopember, Politeknik Negeri Banjarmasin, and STMIK STIKOM Bali.

REFERENCES

[1] M. J. Chonoles, "What is UML?," in *OCUP Certification Guide: UML 2.5 Foundational Exam*, 2018, pp. 17–41.

[2] D. O. Siahaan, *Software Engineering*. Surabaya: Penerbit Andi, 2012.

[3] D. O. Siahaan and F. Irhamni, "Advanced methodology for requirements engineering technique solution (AMRETS)," *Int. J. Adv. Comput. Technol.*, vol. 4, no. 5, pp. 75–80, 2012.

[4] W. N. Robinson and H. G. Woo, "Finding reusable UML sequence diagrams automatically," *IEEE Softw.*, vol. 21, no. 5, pp. 60–67, 2004.

[5] I. Sommerville, *Software Engineering*. 2010.

[6] E. K. Park, "Reuse System : An Artificial Approach Intelligence - Based," pp. 207–221, 1994.

[7] H. O. Salami and M. Ahmed, "A framework for reuse of multi-view UML artifacts," *Int. J. Soft Comput. Softw. Eng. [JSCSE]*, vol. 3, no. 3, pp. 156–162, 2013.

[8] M. A. R. Al-Khiaty and M. Ahmed, "Matching UML class diagrams using a Hybridized Greedy-Genetic algorithm," *Proc. 12th Int. Sci. Tech. Conf. Comput. Sci. Inf. Technol. CSIT 2017*, vol. 1, pp. 161–166, 2017.

[9] A. Adamu and W. M. N. W. Zainon, "Multiview Similarity Assessment Technique of UML Diagrams," *Procedia Comput. Sci.*, vol. 124, pp. 311–318, 2017.

[10] M. A.-R. Al-Khiaty and M. Ahmed, "UML Class Diagrams: Similarity Aspects and Matching," *Lect. Notes Softw. Eng.*, vol. 4, no. 1, pp. 41–47, 2016.

[11] M. A.-R. M. Al-Khiaty and M. Ahmed, "Similarity assessment of UML class diagrams using a greedy algorithm," in *Computer Science and Engineering ...*, 2014.

[12] M. A.-R. Al-Khiaty and M. Ahmed, "Similarity assessment of UML class diagrams using simulated annealing," *Softw. Eng. Serv. ...*, 2014.

[13] D. O. Siahaan, Y. Desnelita, Gustientiedina, and Sunarti, "Structural and Semantic Similarity Measurement of UML Sequence Diagrams," in *International Conference on Information & Communication Technology and System (ICTS)*, 2017, pp. 227–234.

[14] A. Adamu and W. M. N. W. Zainon, "Similarity Assessment of UML Sequence Diagrams Using Dynamic Programming," in *International Visual Informatics Conference*, 2017, vol. 1, pp. 270–278.

[15] H. Storrle, "Towards Clone Detection in UML Domain Models," *Softw. Syst. Model.*, 2011.

[16] M. C. Blok and J. L. Cybulski, "Reusing UML Specifications in a Constrained Application Domain," in *Proceedings 1998 Asia Pacific Software Engineering Conference*, 1998.

[17] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Class Diagram Similarity Measurement : A Different Approach," in *International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2018, vol. 3, pp. 216–220.

[18] Omg, "UML 2.4.1 Superstructure Specification," *October*, vol. 02, no. August, pp. 1–786, 2004.

[19] A. Kutuzov, A. Panchenko, S. Kohail, M. Dorgham, O. Oliynyk, and C. Biemann, "Learning Graph Embeddings from WordNet-based Similarity Measures," 2018.

[20] Z. Wu and M. Palmer, "Verb Semantics and Lexical Selection," 1994.

[21] Arthur M. Jacobs and A. Kinder, "Features of word similarity," *Comput. Lang.*, pp. 1–20, 2018.

[22] J. Gao, B. Zhang, and X. Chen, "A WordNet-based semantic similarity measurement combining edge-counting and information content theory," *Eng. Appl. Artif. Intell.*, vol. 39, pp. 80–88, 2015.

[23] G. Majumder, P. Pakray, A. Gelbukh, and D. Pinto, "Semantic textual similarity methods, tools, and applications: A survey," *Comput. y Sist.*, vol. 20, no. 4, pp. 647–665, 2016.