SEMANTIC OPPOSITENESS FOR INCONSISTENCY AND DISAGREEMENT DETECTION IN NATURAL LANGUAGE

by

NAIDA HEWA NISANSA DILUSHAN DE SILVA

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

December 2020

DISSERTATION APPROVAL PAGE

Student: Naida Hewa Nisansa Dilushan de Silva

Title: Semantic Oppositeness for Inconsistency and Disagreement Detection in Natural Language

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Dejing Dou	Chair
Stephen Fickas	Core Member
Christopher Wilson	Core Member
Heidi Kaufman	Institutional Representative

and

Kate Mondloch	Interim Vice Provost and Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded December 2020

DISSERTATION ABSTRACT

Naida Hewa Nisansa Dilushan de Silva

Doctor of Philosophy

Department of Computer and Information Science

December 2020

Title: Semantic Oppositeness for Inconsistency and Disagreement Detection in Natural Language

Semantic oppositeness is the natural counterpart of the rather more popular natural language processing concept, semantic similarity. Much like how semantic similarity is a measure of the degree to which two concepts are similar, semantic oppositeness yields the degree to which two concepts would oppose each other. This complementary nature has resulted in most applications and studies incorrectly assuming semantic oppositeness to be the inverse of semantic similarity. In other trivializations, "semantic oppositeness" is used interchangeably with "antonymy," which is as inaccurate as replacing semantic similarity with simple synonymy. These erroneous assumptions and over-simplifications exist due, mainly, to either a lack of information, or the computational complexity of calculation of semantic oppositeness. This dissertation considers the following question: How can we convert the linguistic concept of semantic oppositeness to the computing domain? To answer this question, we follow the linguistic definition of oppositeness and develop a novel methodology based on antonymy as well as similarity. We also propose a novel method to embed the obtained semantic oppositeness in a vector space for increased generalization and efficiency. We then consider two realms of applications: inconsistency and disagreements. The inconsistency application helped us track changes in a medical

research domain. The disagreement application accentuated the ability to detect rumours in the social media domain. Finally, we extract the commonalities and patterns in these methodologies to provide a comprehensive summary and a set of recommendations and future work. This dissertation is a culmination of previously published, co-authored material.

CURRICULUM VITAE

NAME OF AUTHOR: Naida Hewa Nisansa Dilushan de Silva

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
University of Moratuwa, Katubedda, Sri Lanka

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2020, University of Oregon
Master of Science, Computer and Information Science, 2016, University of Oregon
Bachelor of Science, Computer Science Engineering, 2011, University of Moratuwa

AREAS OF SPECIAL INTEREST:

Natural Language Processing, Machine Learning

PROFESSIONAL EXPERIENCE:

Graduate Research Teaching Assistant, Department of Computer and Information Science, University of Oregon, 2014 to present
Givens Associate, Argonne National Laboratory, Department of Energy, USA, 2018
Lecturer, Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka, 2011 to present (On study leave since September 2014)
Researcher, LIRNEasia, Sri Lanka, 2013 to 2014

GRANTS, AWARDS AND HONORS:

Graduate Teaching Research Fellowship, Computer and Information Science, 2014 to present

SELECTED PUBLICATIONS:

de Silva, N., Dou, D. & Huang, J. (2017). Discovering Inconsistencies in PubMed Abstracts through Ontology-Based Information Extraction. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 362–371). ACM

de Silva, N. & Dou, D. (2019). Semantic Oppositeness Embedding Using an Autoencoder-based Learning Model. *International Conference on Database and Expert Systems Applications* (pp. 159–174). Springer. doi:10.1007/978-3-030-27615-7_12

de Silva, N., Dou, D. (2020). Semantic Oppositeness Assisted Deep Contextual Modeling for Automatic Rumor Detection in Social Networks. (under review).

de Silva, N., Dou, D. Huang, J. (2020). Discovering Inconsistencies and Similarities in PubMed Abstracts through Ontology-Based Information Extraction. (under review).

Huang, J., Gutierrez, F., Strachan, H. J., Dou, D., Huang, W., Smith, B., ... de Silva, N. et al. (2016b). OmniSearch: a semantic search system based on the Ontology for MIcroRNA Target (OMIT) for microRNA-target gene interaction data. *Journal of Biomedical Semantics*, *7*(1), 1

Huang, J., Eilbeck, K., Smith, B., Blake, J. A., Dou, D., Huang, W., ... de Silva, N. et al. (2016a). The development of non-coding RNA ontology. *International Journal of Data Mining and Bioinformatics*, *15*(3), 214–232

de Silva, N. H. N. D. (2015a). SAFS3 Algorithm: Frequency Statistic and Semantic Similarity Based Semantic Classification Use Case. *Proceedings of Advances in ICT for Emerging Regions (ICTer), 2015 Fifteenth International Conference on* (pp. 77–83). IEEE

de Silva, N. H. N. D., Perera, A. S. & Maldeniya, M. K. D. T. (2013). Semi-Supervised Algorithm for Concept Ontology Based Word Set Expansion. *Proceedings of Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on* (pp. 125–131). IEEE

de Silva, N. H. N. D. (2017b). Relational Databases and Biomedical Big Data. *Bioinformatics in MicroRNA Research*, 69–81

Ratnayaka, G., Rupasinghe, T., de Silva, N., Warushavithana, M., Gamage, V., Perera, M. & Perera, A. S. (2019a). Classifying Sentences in Court Case Transcripts using Discourse and Argumentative Properties. *ICTer*, *12*(1)

Jayawardana, V., Lakmal, D., de Silva, N., Perera, A. S., Sugathadasa, K., Ayesha, B. & Perera, M. (2017a). Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population. *International Journal on Advances in ICT for Emerging Regions*, *10*(1), 1

Wang, P., Ji, L., Yan, J., Dou, D., de Silva, N., Zhang, Y. & Jin, L. (2018a). Concept and Attention-Based CNN for Question Retrieval in Multi-View Learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *9*(4), 41

Gutierrez, F., Dou, D., de Silva, N. & Fickas, S. (2017). Online Reasoning for Semantic Error Detection in Text. *Journal on Data Semantics*. doi:10.1007/s13740-017-0079-6

Lokanathan, S., Kreindler, G. E., de Silva, N. H. N., Miyauchi, Y., Dhananjaya, D. & Samarajiva, R. (2016). The Potential of Mobile Network Big Data as a Tool in Colombo's Transportation and Urban Planning. *Information Technologies & International Development*, *12*(2), pp–63

Ratnayaka, G., Rupasinghe, T., de Silva, N., Warushavithana, M., Gamage, V. & Perera, A. S. (2018). Identifying Relationships Among Sentences in Court Case Transcripts Using Discourse Relations. *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)* (pp. 13–20)

Sugathadasa, K., Ayesha, B., de Silva, N., Perera, A. S., Jayawardana, V., Lakmal, D. & Perera, M. (2018). Legal Document Retrieval using Document Vector Embeddings and Deep Learning. *Science and Information Conference* (pp. 160–175). Springer

Sugathadasa, K., Ayesha, B., de Silva, N., Perera, A. S., Jayawardana, V., Lakmal, D. & Perera, M. (2017). Synergistic Union of Word2Vec and Lexicon for Domain Specific Semantic Similarity. *IEEE International Conference on Industrial and Information Systems (ICIIS)*, 1–6

Upeksha, D., Wijayarathna, C., Siriwardena, M., Lasandun, L., Wimalasuriya, C., de Silva, N. H. N. D. & Dias, G. (2015). Comparison Between Performance of Various Database Systems for Implementing a Language Corpus. *International Conference: Beyond Databases, Architectures and Structures* (pp. 82–91). Springer

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Dejing Dou, for his continuous guidance, unwavering support, and encouragement throughout the Ph.D. process. Dr. Dou is an excellent advisor who patiently guided me to develop the essential skills for a PhD student: how to be a better researcher, how to identify and define an important and influential research problem, how to review papers, how to write better research papers, and how to give effective presentations. This work would not have been possible without his guidance and many insightful discussions with him.

I would like to thank the members of my committee, Dr. Stephen Fickas, Dr. Christopher Wilson, and Dr. Heidi Kaufman, for their timely feedback and suggestions which steered my research, and thus this dissertation, toward better outcomes.

I thank my family for the selfless support they provided for me to pursue this degree. Especially, I am grateful to my parents for everything they have done for me, and to my sister for her love and never-ending support. They have always encouraged me to follow my dreams and explore the world, even though this meant that we have been separated by a great distance and a 12-hour time difference while I have been pursuing this degree.

I spent a Summer working for the Argonne National Laboratory under the supervision of wonderful mentors. I would like to thank Dr. Boyana Norris for supporting me in securing the internship and then mentoring me throughout its duration. I also thank Dr. Anshu Dubey, my on-site mentor, for making my internship experience productive and welcoming.

I wish to thank Dr. Amal Shehan Perera for introducing me to academic research when I was an undergraduate. Dr. Amal Shehan Perera, Dr. Chandana Gamage, Ms. Vishaka Nanayakkara, Dr. Shahani Markus, and Dr. Daya Chinthana Wimalasuriya were instrumental in my deciding to pursue a graduate degree. Thank you for pushing me beyond my comfort zone and for providing opportunities for my professional development.

During my years in Oregon, I met friends who became integral in both my academic and personal life. I have enjoyed their company throughout the years at many joyous occasions and savoured their support many sour days. Especially at the very end of the degree, while I was writing this dissertation and I lost my apartment: Shravan Kale and Abhishek Yenpure helped me with transporting my belongings and myself to the residence of Manish Mathai, who graciously agreed to let me stay at his place while I finished the dissertation. All of them did this while a pandemic was happening. Without their help and kindness, this dissertation would not have finished on time. Sam Pollard helped me in contacting the appropriate authorities and legal officers to hear my plea. I would also like to thank my friend Ellen Klowden, who helped me with editing this dissertation, as well as most of the papers that make up the basis of this dissertation.

Finally, my fellow friends in the AIM Lab and the CBL research group, the Computer and Information Science department, and the University of Oregon made my Ph.D. life excellent; many thanks to all of you for your support, feedback, and friendship.

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

Semantic similarity measures are widely used in Natural Language Processing (NLP) applications (Gomaa & Fahmy, 2013; Stavrianou, Andritsos & Nicoloyannis, 2007; Turney, 2001). The reason for this popularity is the fact that mining methods built around the simple exact match approach would yield results with a weaker recall in comparison with the golden standard. Free text has a tendency to use synonyms and similar text in substitution, which may go unnoticed if a direct match method is used in text mining.

Semantic oppositeness is the natural counterpart of the semantic similarity function (de Silva et al., 2017). While semantic similarity yields the degree to which two concepts are similar in a given domain, to be used for the purpose of confidence calculation in text mining applications, semantic oppositeness yields the degree to which two concepts oppose each other in a given domain, for the same purpose.

The use of an oppositeness measure in text mining is especially relevant in the case of contradiction finding or in the case of mining for negative (negation) rules from a corpus. This, in turn, helps in building reasoning chains and other utilities for various front-end applications, such as question-answer systems and chat bots. It can also be used in text mining applications that concern fake news or propaganda, given that the candidate text that is being analyzed would contain opposing concepts to the generally accepted corpus of knowledge.

An important point to note in the case of semantic oppositeness is its relation to *antonymy*. The discussion provided by Jones, Murphy, Paradis and Willners (2012) on the differences in the definitions of *antonymy* and *oppositeness* is an important perspective. They define *antonymy* as a pair-wise relation of lexical items, following

the earlier work by Murphy (2003), which specifically labelled these words as *canonical antonyms*. On the other hand, *oppositeness* is defined as a semantic relation between word pairs which are understood to have meanings that are opposed to one another in a given context.

The importance of oppositeness as a linguistic feature is supported by many studies. Ye (2014) states that oppositeness is a fundamental part of language, even more so than synonymy. They bring up the claim by Murphy (2003), which comments, "[u]nlike synonymy, everyone agrees that antonymy exists, and it is robustly evident in natural language." Mikołajczak-Matyja (2018) claims oppositeness is a universal feature of language structure which obtains its role from thinking and culture, emphasising the structuralist approaches to language. The relevance of oppositeness to the organisation of text and discourse in natural languages is discussed in a number of works (Cruse, 1986; Jones, 2003; Jones et al., 2012; Justeson & Katz, 1991; Lyons, 1987; Murphy, 2003; van de Weijer, Paradis, Willners & Lindgren, 2014; Willners, 2001). In contrast, in the language acquisition perspective, oppositeness is considered *cognitively primary* (Cruse, 2011; Cruse, 1986; Phillips & Pexman, 2015). The lack of a proper oppositeness measure, in fact, is the reason for most text mining tasks engage the wrong generalization of reducing semantic oppositeness to antonymy (Paradis, Goldblum & Abidi, 1982) or to the inverse of semantic similarity (de Silva et al., 2017).

In this dissertation, we answer the following question: *How can we convert the linguistic concept of semantic oppositeness to the computing domain?* To answer this question, we need to develop a novel methodology based on antonymy as well as similarity. Additionally, we propose a novel method to embed the obtained semantic oppositeness in a vector space for better generalization and efficiency.

Then, we consider two realms of application: inconsistency and disagreements. Inconsistency finding helped us track changes in a medical research domain, as well as it helped us suggest enrichment for an associated ontology. The disagreement detection accentuated the ability to distinguish true rumours and false rumours in the social media domain. Finally, we look for commonalities and patterns in these methodologies, and synthesize our findings with a set of recommendations and future directions.

The culmination of this work answers our dissertation question, *How can we convert the linguistic concept of semantic oppositeness to the computing domain?*, by creating a methodology for calculating semantic oppositeness and embedding it in a vector space for generalized usage.

1.1 Dissertation Outline

This dissertation is organized into the following three parts:

A. Techniques: Chapters II through IV.

B. Applications (Use Cases): Chapters V and VI.

C. Summary and Future Work: Chapters VII and VIII.

Part A first surveys the resources and techniques in both linguistic and computing domains, which are utilised in the rest of this dissertation. Then it introduces the new semantic oppositeness measure, as well as its more generalized and efficient embedded form. Part B incorporates the techniques of Part A into two important use cases: inconsistency detection and disagreement detection. Part C synthesizes the findings of Parts A and B and gives recommendations for future work.

In particular, the content of the individual dissertation chapters is as follows. Chapter II provides a background to the linguistic base of semantic oppositeness, as well as the computing resources and techniques that we employ to convert

semantic oppositeness to the computational linguistics domain and test on subsequent use cases. Chapter III introduces the new semantic oppositeness measure in a formal mathematical setting, with computerized implementations following the linguistic observations and principals of Chapter II. Chapter IV takes the formalized computational model of semantic oppositeness from Chapter III and embeds the measure in a vector space to attain better generalizability and efficiency. Chapter V applies the basic semantic oppositeness measure of Chapter III to find inconsistencies between research paper abstracts, as well as to find potential strong relationships to enrich an ontology in a relevant domain. Chapter VI applies the embedded semantic oppositeness measure of Chapter IV to detect rumors in social networks by means of discovering disagreements. Chapter VII synthesizes findings and best practices of the previous chapters; and Chapter VIII concludes this dissertation by offering recommendations for the future research.

1.2 Co-Authored Material

A significant portion of the content in this dissertation is adopted from collaborative research work and manuscripts that I have completed, as leading author, during my PhD program. Each manuscript either has been already published or is currently under review by a conference or a journal. The content of each manuscript includes the text, figures, and experimental results, all of which are primarily composed by myself. The following listing indicates the chapters that contain manuscript content and the authors who contributed to the manuscript (i.e., myself and co-authors); note that a detailed division of labour for each manuscript is provided at the beginning of its corresponding chapter.

- Chapter II is an amalgamation of backgrounds and related work for all the subsequent Chapters. As such, it is a collaboration between myself, Dejing Dou, and Jingshan Huang.
- Chapter III is based on two published conference publications and one journal paper under review. The first conference paper and the journal paper are a collaboration between myself, Dejing Dou, and Jingshan Huang. The second conference paper is a collaboration between myself and Dejing Dou.
- Chapter IV is based on a published conference publication and is a collaboration between myself and Dejing Dou.
- Chapter V is based on a published conference publication and is a collaboration between myself, Dejing Dou, and Jingshan Huang.
- Chapter VI is based on a conference publication under review and is a collaboration between myself and Dejing Dou.

Part A

Techniques

In this part of the dissertation, we provide a background on semantic oppositeness and other relevant concepts, and then we introduce our computational modelling of the linguistic concept of semantic oppositeness, followed by a methodology to embed it in a vector space for increased generalization and efficiency.

CHAPTER II

BACKGROUND

2.1 Semantic Oppositeness

Albeit not as extensively as its counterpart, semantic similarity (Jiang & Conrath, 1997; Wu & Palmer, 1994), there have been a few studies on the derivation and uses of semantic oppositeness (de Silva et al., 2017; Mettinger, 1994; Paradis et al., 1982; Rothman & Parker, 2009; Schimmack, 2001). However, almost all of these studies reduce oppositeness from a continuous scale to bipolar scales (Rothman & Parker, 2009; Schimmack, 2001) or anonymity (Jones et al., 2012; Paradis et al., 1982).

Lobanova (2012) claim that antonymy, and by extension oppositeness, as a relation causes a great amount of confusion and disagreement among scholars. As mentioned in Section I, Jones et al. (2012) builds on the fundamentals laid by Murphy (2003) to claim that *antonymy* is a pair-wise relation of lexical items, as opposed to *oppositeness* being defined as a semantic relation between word pairs which are are understood to have meanings that are opposed to one another in a given context. Murphy, Jones and Koskela (2015) study the presence of oppositeness and a few other linguistic features in parallel contexts. Mikołajczak-Matyja (2018) describes two types of oppositeness from genesis: logical necessity (e.g., top/bottom) and accidental or pragmatic contrast (e.g. coffee/tea). They further claim that the later is felt by the language user, only due to the frequency of the use of a choice between the two options (Cruse, 2011; Cruse, 1986). In this work, we encompass the entirety of the oppositeness concept, regardless of genesis, because this work is focused on the computational application of the linguistic phenomenon, rather than analysing the phenomenon in a linguistic perspective.

One extremely important aspect of this study is presenting oppositeness as a continuous scale. This perspective of oppositeness is supported by, Mikołajczak-Matyja (2018) who claim that oppositeness exists in various degrees and scales, where some pairs are more opposite than others. They support this claim with the analysis of the opposite pairs *huge-tiny* and *large-small*, following an example from Cruse (1994). With linguistic analysis, they show that the pair *huge-tiny* is symmetrical but does not exhaust the dimension to the same degree as *large-small* does. Thus, on an oppositeness scale, *huge-tiny* would appear closer to each other than *large-small* would. The scale difference of opposite pairs is discussed by Lehrer and Lehrer (1982) where they distinguish symmetrically placed opposite pairs as *perfect opposites*.

This approach is also supported by Lobanova (2012) who define the oppositeness scale as *broad sense antonymy* and traditional antonymy as *narrow sense antonymy*. Despite the difference in the name, the explanation in the text makes it clear that *broad sense antonymy* refers to the oppositeness scale and that the authors support the adaptation of it over traditional antonymy, which they dub as *narrow sense antonymy*. Further, under the analysis by Lobanova (2012), the pairs that fall on an oppositeness scale are further described as *gradable opposites*, given that there are degrees of oppositeness achievable on the scale. This is supported by literature (Cruse, 1986; Lyons, 1987) while rejecting the older *non-gradable opposites* concept by Kempson (1977). Not confining oppositeness to *canonical* pairs is the recommendation of Jones (2003) as well.

Jones (2003) further claim that oppositeness transcends grammatical category, given that corpus studies have not yielded a relevant link between the grammatical category of an opposing word pair and the grammatical category which that pair

fulfills in the text. In our study, following this observation, we do not distinguish among the grammatical categories of words.

In defining and discovering semantic oppositeness, the literature gives us two main principles to follow. These are the *Minimal difference with maximal similarity principle* and the *Irrelevancy principle*. We discuss the background and support for these principles in Section 2.1.1 and Section 2.1.2 respectively.

2.1.1 Minimal Difference with Maximal Similarity Principle. The *minimal difference with maximal similarity* principle is commonly explained using Act 3 Scene 1 from *Julius Caesar* by Shakespeare (1955). In this classic scene, *Julius Caesar* is attacked by the senators, such as *Casssius*, *Casca*, and *Cinna*, with the intent of assassinating him. He stands amidst multiple stabbings, but it is when *Brutus*, an individual who is akin to a son to him, stabs him, that he falls at the end. This is used to explain the idea of the *minimal difference with maximal similarity* principle, given a candidate word W_1 which is akin to *Caesar* in the example, the oppositeness against other words W_2, W_3, W_4, W_5 needs to be determined (*Casssius*, *Casca*, *Cinna*, and *Brutus*). Given that W_1 (*Caesar*) shares very little in common with W_2 (*Casssius*), W_3 (*Casca*), W_4 (*Cinna*), the impact or the degree of oppositeness W_1 has with them is minimal. But when W_1 (*Caesar*) is considered against W_5 (*Brutus*), the shared commonalities between W_1 and W_5 magnify the impact, or the degree of oppositeness which exists between them, to be beyond what it was with W_2, W_3, W_4.

The words that are to be put on the scale are to follow the *minimal difference* with *maximal similarity* principle as established by fundamental studies in the literature (Clark, 1972; Cruse, 1986; Lyons, 1987; Murphy, 2003). The property of *minimal difference* with *maximal similarity* in semantic oppositeness which results in simultaneous proximity and distance between words distinguishes it from other

semantic relations by giving it an exceptional status according to some studies (Cruse, 1986; Muehleisen & Isono, 2009). While the idea is unequivocally supported, different linguistic studies rationalize the principle with different arguments. Nevertheless, all of them come to the same conclusion on the validity of the principle. In this section we discuss such support of this key principle.

Jones (2003) argue that in an opposite pair, it is not enough to have opposition of meaning. The pair must also have a strong and established lexical relationship. They provide the example that on the idea of height, an expected relationship exists in the pair *tall-short*, but it does not in *lofty-petite*. Thus, they propose a methodology based on observed-to-expected ratio of co-occurrence. This idea is also supported by Muehleisen and Isono (2009), by the emphasis they put on the context in the task of semantic oppositeness. They propose that the *goodness* of the suggestion in adjectival opposition is increased when there is similarity in the collocation profile of the relevant pair. This follows that both words in question are describing concepts of the same *type*.

Phillips and Pexman (2015) define oppositeness as a function when two words "differ maximally but only on a single dimension," thus rendering the meaning simultaneously similar and different. Yet again, this claim is supported by literature (Clark, 1970a; Jones, 2003; Murphy, 2003; Owens Jr, 2015). Mikołajczak-Matyja (2018) note that for oppositeness, minimal semantic contrast between units is important. The definition of incompatibility of meanings is given as "the existence of at least one feature contrasting those meanings in describing the narrower category of opposition the emphasis is on the smallest difference between units" (Leech, 1990).

The minimal contrast rule is claimed to be an evolution of the contrast principle proposed by Clark (1970b), which Mikołajczak-Matyja (2018) formulates from Kostić

(2015) as: "For two words to be minimally different, they must share all their crucial semantic properties but one, i.e., they differ in only one relevant criterion". Thus, they postulate that the work on semantic oppositeness should start on the focus of *significant similarity* rather than that of the *maximal difference* first. These similarities are argued to be able to be found using:

- Hyponym-Hypernym structure (Mettinger, 1994)[1].
- Being in a common semantic domain (Jones, 2003; Muehleisen & Isono, 2009).
- Having a set of common features (Murphy, 2003).
- Exhibiting a plan of equivalence (Davies, 2012).

Once the similarity is established, the contrast needs to be brought forward. These differences can be emphasised using:

- Having values that are distant from each other in a given semantic dimension (Mettinger, 1994).
- A single feature (componential) being distinct (Murphy, 2003).
- Exhibiting a plan of difference (Davies, 2012).

On the question of balancing *minimal difference* against *maximal similarity*, Murphy (2003) states, "The best antonyms are those that [...] extend their similarities to as many properties as possible while maintaining a single relevant difference". This idea is explained with the example pairs *yesterday-tomorrow* and *Monday-Wednesday*. The yesterday-tomorrow pair is shaped by the shared bound definition to *today*, while Monday-Wednesday is not shaped by such a bound definition to *Tuesday* (Cruse, 2011; Mikołajczak-Matyja, 2018). The latter has to be inferred. This type of oppositeness is defined as latent (Cruse, 2011; Cruse, 1986).

[1]Note that Mettinger (1994) uses the French words archisememe/archilexeme to mean hyponym-hypernym.

Mikołajczak-Matyja (2018) further discusses encapsulated oppositeness in the context of *minimal difference* and *maximal similarity*. The provided example is that the pair *giant-dwarf* has encapsulated the oppositeness basis of the pair *large-small*. This encapsulation brings the abstract pair large-small to the concrete pair giant-dwarf by means of adding dimensions such as *humanoid*, *living organism*, and others. Nevertheless, note that in all these encapsulating new dimensions, the pair stay *similar* preserving the *maximal similarity* property. This follows the rule laid by Cruse (1986) that, "proportional participation of the differentiating feature in the totality of meaning of the units" is the most important factor in determining oppositeness while being relevant. This also fulfills the criterion of "a number of opposed values of features contributing to the dimension of opposition" set by Hermann, Conti, Peters, Robbins and Chaffin (1979).

These observations in the literature on *maximal similarity* being both compulsory and primal over the more obvious *minimal difference* translates to a nonlinear mapping function in computing terms.

2.1.2 Irrelevancy Principle. The *irrelevancy principle* states that all words which exist in the vocabulary V will not feature on the oppositeness scale of a given word W. Only a subset S of V will feature on the oppositeness scale of the word W. The rest of the words in V are deemed irrelevant or orthogonal. An important point to note is that this irrelevancy does not reduce oppositeness to antonymy. It still remains a scale with words within a certain threshold. Without violating this principle, Jones et al. (2012) provides examples of such pairs being used in British newspapers in an opposite context, to show how words that are not traditional antonyms are used in opposite contexts.

Mikołajczak-Matyja (2018) claims that when the amount of *differences in similarity* approaches one, the oppositeness approaches a state which they call *prototypical*. The definition of *prototypical* is built from Murphy (2003), which states: "The most prototypical examples of contrast relations involve items that differ on one point of meaning". This oppositeness property is then described with the examples of *male-female* and *ivy-mystery*. The argument raised by Mikołajczak-Matyja (2018) on grounds of Murphy et al. (2015) is that, male-female marks distant points in a singular gender dimension, whilst being equal in all other aspects. In contrast, ivy-mystery differs in multiple dimensions, including and not limited to *concrete-abstract* and *living organism-not a living organism*.

Jones et al. (2012) further point out, with the example *limerick* and *pencil*, that the given pair is unlikely to be construed as opposites. They postulate that the reason for this is the fact that semantic opposition is defined in tandem with similarities as much as it is with differences. Therefore the oppositeness of these words is undefined. Here we observe how this principle is tied with arguments in Section 2.1.1.

On the question of *irrelevancy*, Lobanova (2012) discusses the *Substitutability Hypothesis* as tested by Charles and Miller (1989). The basic idea is that if the candidate opposite pair is relevant, one word in the pair can be seamlessly replaced by the other in a sentence taken from a corpus. They further discuss the biases where the probability of one word being substituted by the other is different when considered the other way round. But nonetheless, it stands that as long as the pair is relevant, the substitution can be made.

These observations in literature on the *irrelevancy principle*, which preserves the oppositeness scale property by not reducing oppositeness to anonymity, while still defining a limit to which the words of the vocabulary can be admitted to a scale

of oppositeness with respect to a given word, translates to a threshold function, in computing terms.

2.2 WordNet

WordNet (Princeton University [Princeton], 2020) is a well-known, large, lexical, ontological (Arvidsson & Flycht-Eriksson, 2016; Gruber, 1993) database. It was created by the Cognitive Science Laboratory of Princeton University, United States (Miller, Beckwith, Fellbaum, Gross & Miller, 1990). By grouping words together into sets of synonyms called synsets, it represents semantic relationships between words. In total, the database consists of 150,000 different words. Each of the words is coupled with a short description, for the applications that need the system to function as a *Dictionary* in addition to as a *Thesaurus*. The accompanying software tools, as well as the database, are released under a BSD-type license. Out of the various semantic mappings present in WordNet, this study utilizes the *Hyponym - Hypernym* mapping and the *Antonym* mapping. Fig. 1 shows an example of an extract of the Hyponym – Hypernym tree present in WordNet. (Adapted from de Silva et al. (2013))

2.3 Semantic Similarity

Semantic similarity of two entities is a measure of the likeness of the semantic content of the said two entities (Li, Bandar & McLean, 2003; Lord, Stevens, Brass & Goble, 2002). It is common to define semantic similarity using topological similarity by means of ontologies. Using WordNet (Miller et al., 1990), Wu and Palmer (1994) proposed a method to derive the similarity between two words in the 0 to 1 range. The approach proposed by Jiang and Conrath (1997) measures the semantic similarity between word pairs using corpus statistics and lexical taxonomy. By means of Shima (2016), the strengths of these algorithms were evaluated by de Silva (2015a). In

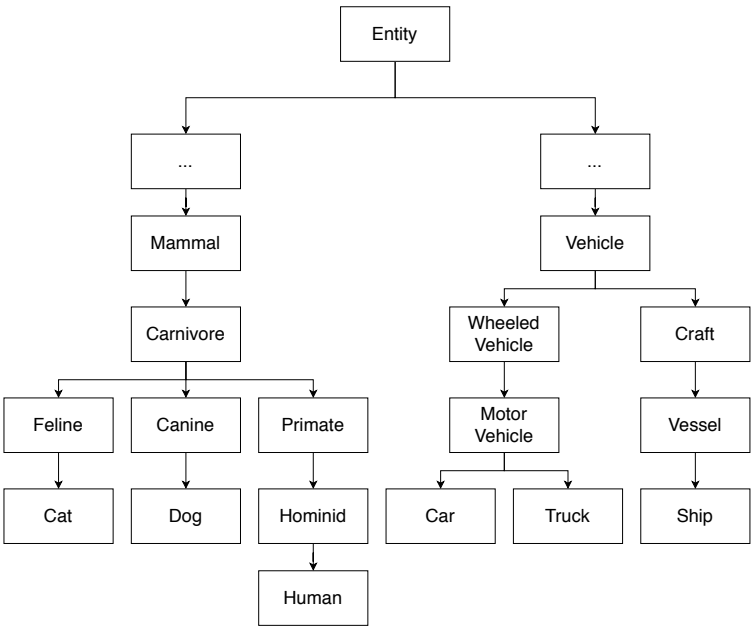


Figure 1. Hyponym – Hypernym graph

accord with that evaluation, we selected the implementation by Wu and Palmer for the purposes of this work.

A set of examples of word similarities are shown in Table 1 of de Silva (2015a). For the similarity with Car, the same word gets the perfect score of 1. $Truck$ gets a higher score than $Ship$, because a $Truck$ too, is a land vehicle, like a Car. However, $Ship$ gets a higher score than $Book$ because a $Ship$ is a vehicle and a $Book$ is not. $Book$ gets a higher score than Air because the $Book$ is solid and Air is not. Air gets a higher score than $Thought$ because Air is a physical entity and a $Thought$ is not.

Table 1. Word Similarities Using Wu and Palmer Method

Word 1	**Word 2**	**Similarity**
Car	Car	1.0000
Car	Truck	0.9231
Car	Ship	0.7200
Car	Book	0.5217
Car	Air	0.3158
Car	Thought	0.2105

A useful observation from this is the fact that, no matter how dissimilar two words are, if both of those words exist in WordNet, this method will return a greater-than-zero value. Thus, there exists an inherent bias towards declaring that two words have a non-zero similarity; rather than declaring that there exists a difference. Thus, in the Sections 3.3, 3.5, and 5.4.4, we use dissimilar weights named "yes weight" (W_{yes}) and "no weight" (W_{no}), where W_{no} is larger than W_{yes} by a considerable amount.

2.4 Information Extraction

Information extraction is a process, in the domain of Artificial Intelligence (AI), for acquiring knowledge by looking for occurrences of a particular class of objects, and looking for relationships among objects, in a given domain. The objective of information extraction is to find and retrieve certain types of information from text. However, it does not attempt to comprehend natural language. Comprehending natural language is handled by the research area, *natural language understanding*. *Natural language understanding* is what chat bot AIs or personal assistant AIs attempt to do. Information extraction is also different from *information retrieval*, which retrieves documents, or parts of documents, related to a user query from a large collection of documents. *Information retrieval* is what search engines do. The main difference between *information retrieval* and *information extraction* is that the latter goes one step further by providing the required information itself, instead of providing a pointer to a document.

In an information extraction task, the input is text which is either unstructured, or slightly structured, such as HTML or XML. Usually the output is a template set, filled in with various information that the system was supposed to find. Thus, the information extraction process is a matter of analyzing document(s) and filling template slots with values extracted from document(s).

There are two main methods of information extraction in literature: (a) attribute-based extraction; and (b) relation extraction. In attribute-based extraction, the system assumes the entire text to be referring to a single object. Thus, the task is to extract attributes of said object. This is typically done using regular expressions. Relation extraction, on the other hand, extracts multiple objects, and relationships among them, from a document. One famously efficient way to do this is the FASTUS method by Hobbs, Appelt, Bear, Israel et al. (1993).

2.5 Ontologies and OBIE

An ontology is defined in information science as "formal, explicit specification of a shared conceptualisation" (Gruber, 1993). Ontologies are used to organize information in many areas as a form of knowledge representation (Gruber, 1995). These areas include: artificial intelligence (Maynard, Yankova, Kourakis & Kokossis, 2005), linguistics (de Silva, 2015a; Wijesiri et al., 2014), biomedical informatics (Huang et al., 2016b; Pisanelli, Gangemi & Steve, 1999), law (Bruckschen et al., 2010; Jayawardana et al., 2017b; Letia & Cornoiu, 2010; Wyner, 2010), library science, enterprise bookmarking, and information architecture (Vargas-Vera et al., 2002). In each of these use cases, the ontology may model either the world, or a part of it, through the said area's lens (de Silva et al., 2013).

2.5.1 Ontology for MIcroRNA Targets (OMIT).

The Ontology for MIcroRNA Targets (OMIT) was created by Huang et al. (2016b) with the purpose of establishing data exchange standards and common data elements in the microRNA (miRNA) domain. Biologists and bioinformaticians can make use of OMIT to leverage emerging semantic technologies in knowledge acquisition and discovery for more effective identification of important roles performed by miRNAs (through their respective target genes) in humans' various diseases and biological processes. The

OMIT has reused and extended a set of well-established concepts from existing bio-ontologies; e.g., Gene Ontology (Ashburner, Ball, Blake, Botstein et al., 2000), Sequence Ontology (Eilbeck, Lewis, Mungall, Yandell et al., 2005), PRotein Ontology (PRO) (Natale, Arighi, Barker, Blake et al., 2011), and Non-Coding RNA Ontology (NCRO) (Huang et al., 2016a). Metrics of OMIT are shown in table 2.

Table 2. Metrics of OMIT

Number of classes:	2226
Number of individuals:	1158
Number of properties:	126
Maximum depth:	35
Maximum number of children:	316
Average number of children:	14
Classes with a single child:	280
Classes with more than 25 children:	104
Classes with no definition:	2226

2.5.2 Ontology-Based Information Extraction. Ontology-Based Information Extraction (OBIE) is a sub-field of information extraction. In this, ontologies are used to make the information extraction process more efficient and effective. In most cases, the output is also presented through an ontology. But that is not a requirement. As mentioned in 2.5, generally, ontologies are specified for particular domains. Given that information extraction is essentially concerned with the task of retrieving information for a particular domain (as mentioned in the first paragraphs of Section 2.4), it is rational to conclude that an ontology that has formally and explicitly specified the concepts in that domain would be helpful in this process.

A more formal definition of OBIE was given by Wimalasuriya and Dou (2010): "a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents

the output using ontologies." Following is a brief introduction as to how an ontology can improve the information extraction process.

Consider the short paragraph about hormones given in Fig. 2. It can be observed that a named entity recognition process would extract the proper nouns; *Insulin*, *Testosterone*, *beta cells*, *Sertoli cells*. But a general information extraction system would not know what each of these proper nouns are. A human with enough bio-medical knowledge, on the other hand, would know that *Insulin* is a *peptide hormone*, while *Testosterone* is an *anabolic steroid*. This exactly is the problem solved by OBIE. For the hormone domain, a simple ontology can be introduced, as shown in Fig 3.

> Hormones are a class of signaling molecules, found in multicellular organisms, which regulates various functions. One of the most commonly known hormones is Insulin, which regulates the metabolism of carbohydrates, fats, and protein. It is produced at the pancreatic islets by beta cells. Another well-known hormone, Testosterone, has various functions, such as activating genes in Sertoli cells.

Figure 2. A short paragraph about hormones

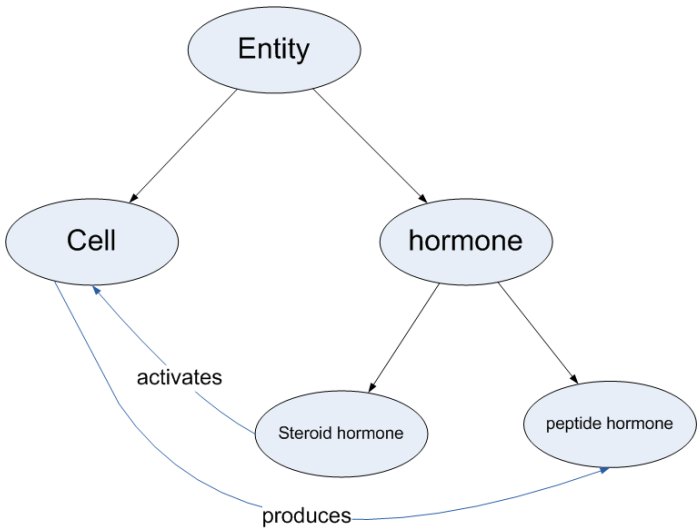


Figure 3. A simple ontology for Hormones

Now with the *is_a* relationship, the OBIE system can tag the proper nouns *Insulin* and *Testosterone* to be of type *hormone* in Fig 3. Next, with the *activates* relationship, the OBIE system can tag *Sertoli cells* as of type *Cell*, as well as refining the tag of *Testosterone* to the more granular *Steroid hormone*. Similarly, the relationship *produces* can be used to tag *beta cells* as of type *Cell* as well as refining the tag of *Insulin* to the more granular *peptide hormone*. Thus, OBIE successfully tags all the proper nouns that were extracted.

OBIE systems that use the GATE architecture rely, at least partly, on Linguistic rules represented by regular expressions such as this. Another way that an ontology can facilitate information extraction is by creating gazetteer lists. The process of creating a gazetteer list from an ontology is rather straightforward: The tree of the concept hierarchy that is rooted at the desired concept is selected, and all the instances that occur in the said rooted tree are then added to the gazetteer list.

When information extraction is performed with machine learning algorithms, it is possible to use ontologies in several ways. Classification algorithms can be used to recognize instances and property values from the ontology. Maximum entropy models can be used to predict attribute values in a sentence. Similarly, Conditional Random Fields (CRF) can be used to identify attribute values in a sentence. The above-described methodologies make up the *Information Extraction Module* of a typical Ontology-Based Information Extraction system.

Other than the *Ontology* and the above-described *Information Extraction Module*, there are two other main components in an OBIE system. The first one is the *Preprocessor*. The text input of an OBIE system first goes through a preprocessor component, which converts the text to a format that can be handled by the IE module. For example, tags from an HTML file can be removed in this component. Thus,

the *Information Extraction Module* would be receiving pure text content. A *semantic lexicon* for the language is usually used as a helper for the *Information Extraction Module*. As mentioned in section 2.2, for the general English language-based information extraction tasks, it is most common to use the WordNet (Princeton, 2020) lexical database and the toolkit thereof. One of the most important components of an ontology for an OBIE system is the set of relationships present in the ontology. Those are the ones that can be used to build extraction rules for the information extraction system. This is exactly the problem with OMIT. Even though it has a very extensive hierarchy of concepts and instances, it contains few or no relationships between the said entities, other than the *is_a* relationship compulsory for the *Hyponym–Hypernym* tree. Thus, some of the most powerful conventional OBIE methods cannot be used alongside OMIT.

2.6 Open Information Extraction

The requirement of having pre-specified relations of interest is the main drawback of the traditional information extraction systems. Open Information Extraction systems (Etzioni, Banko, Soderland & Weld, 2008; Etzioni, Fader, Christensen, Soderland & Mausam, 2011; Fader, Soderland & Etzioni, 2011; Levy, Dagan & Goldberger, 2014; Mausam, Schmitz, Soderland, Bart & Etzioni, 2012; Wu & Weld, 2010) solve this problem by extracting relational triples from text, by identifying relation phrases and associated arguments in arbitrary sentences without requiring a pre-specified vocabulary. Thus, it is possible to discover important relationships that are not pre-specified.

Usually, Open Information Extraction systems automatically identify and extract binary relationships from sentences given the parsed text of the target language. The parsed text provides the dependency relationships among the various phrases

of the sentence. The Open Information Extraction system used in this study, OLLIE (Mausam et al., 2012), is different from others in its genre, due to the fact that it works on a tree-like representation (a graph with only small cycles) of the dependencies of the sentence, based on the Stanford compression of the dependencies, while other Open Information Extraction systems operate on flat sequences of tokens. Thus, OLLIE is uniquely qualified to capture even long-range relations. Given that open information extraction does not depend on pre-configured rules, we are using Open Information Extraction as a bridge between OMIT, which is an ontology with few or no relations as described in section 2.5.1, and the conventional OBIE methods described in 2.5.2. (More information on this is discussed in Section 5.9.)

2.7 TF-IDF

In information retrieval tasks, to indicate how important a given word is in a document within the context of a certain corpus, the TF-IDF (term frequency–inverse document frequency) (de Silva, 2015b; Leskovec, Rajaraman & Ullman, 2014) can be used as a statistic. There are two components in the TF-IDF statistic. The first component is the term frequency (tf), which indicates how important the given word is in the given document. Usually, it is used with 0.5 double normalization, where $f(T, d)$ is the frequency of term t in document d as follows in Equation 2.1.

$$tf(t, d) = 0.5 + \frac{0.5 * f(T, d)}{max\{f(w, d) : w \in id\}} \tag{2.1}$$

Second component of the statistic is the inverse document frequency (idf), where N is the total number of documents in the corpus and d is the number of documents in which t appears. The formula is as follows in Equation 2.2.

$$idf(t, D) = log\frac{N}{1 + |\{d \in D : t \in d\}|} \tag{2.2}$$

2.8 Word Embedding

Embedding has recently risen as an emerging field in the domain of Natural Language Processing (NLP), parallel to its rise in popularity in other domains such as knowledge representation (Wang, Dou, Wu, de Silva & Jin, 2019). As defined by Mikolov, Sutskever, Chen, Corrado and Dean (2013a), a word embedding system consists of a set of techniques for modeling a selected natural language and learning features thereof. The objective of these systems is to map the words in the domain to vectors so that a model which has a distributed representation of words is created in a multi-dimensional vector space. It has been shown that applying vector calculus on the vectors would yield semantic or linguistic relationships among them as shown in Fig 4.

Some examples for the leading algorithms for this task are; Word2vec (Mikolov, Sutskever, Chen, Corrado & Dean, 2013b), GloVe (Pennington, Socher & Manning, 2014), BERT (Devlin, Chang, Lee & Toutanova, 2018), XLNet (Yang et al., 2019), and Latent Dirichlet Allocation (LDA) (Das, Zaheer & Dyer, 2015). In considering the flexibility, the ease of customization, and the wide usage, in this study we use word2vec as the starting point for our embedding system. Even-though this study is focused on embedding oppositeness, rather than embedding words, given that oppositeness is an emergent property between pairs of words, the points of embedding in this study remain as words. This is the reason it is possible to use word2vec as a reasonable starting point.

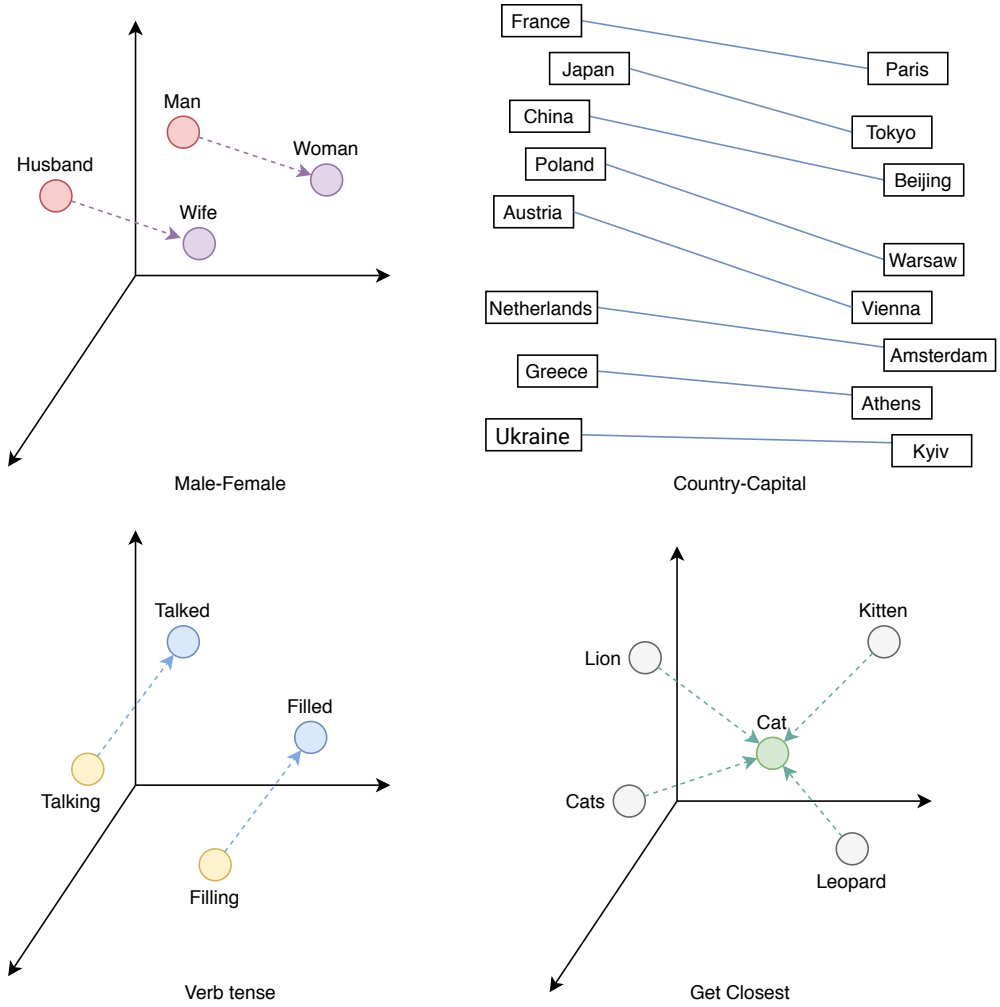


Figure 4. Uses of Word Embedding

2.9 Autoencoders

Autoencoders are the simplest form of the representation learning algorithms. They consist of two components, an encoder and a decoder. The encoder takes an unlabeled input and derives a latent representation of the input. The decoder takes the said latent representation and attempts to reconstruct the input. Hence, the error of an autoencoder is defined as the difference between the input to the encoder and the output of the decoder. A basic autoencoder is shown in Fig 5.

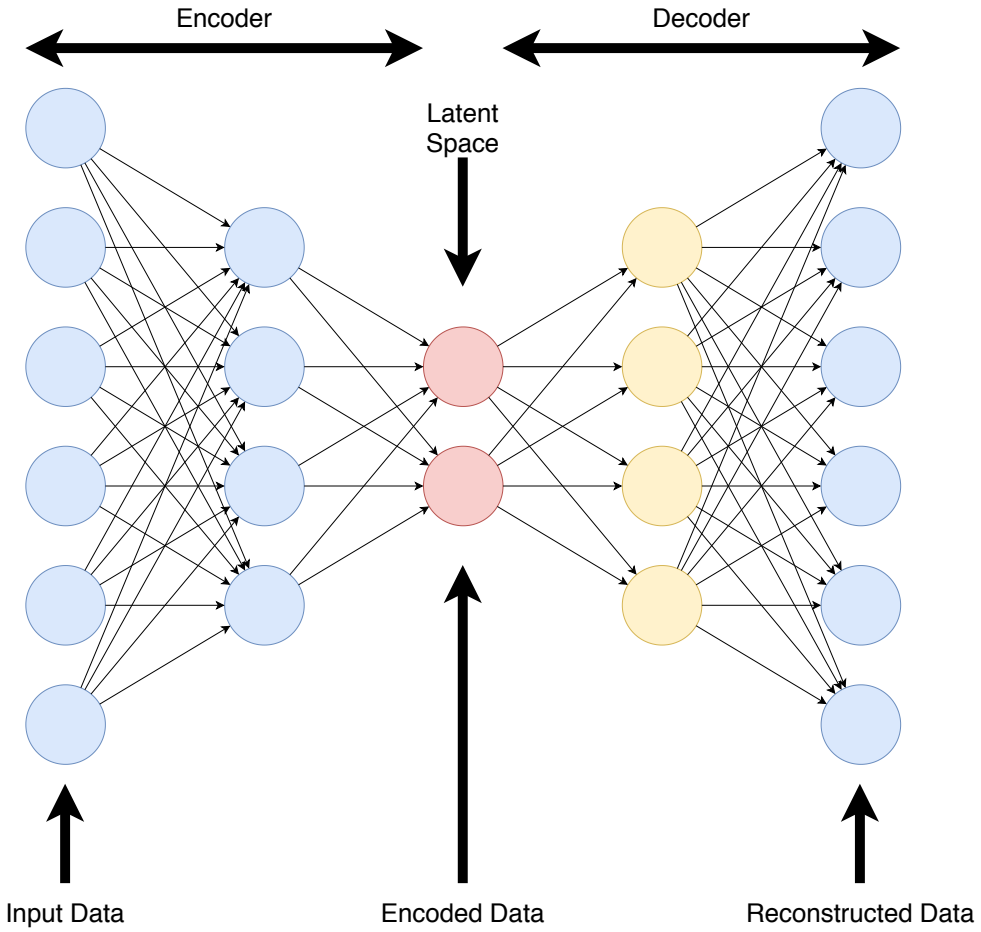


Figure 5. A basic autoencoder

While autoencoders are trained to preserve as much information as possible, special steps are taken to prevent them from learning the identity function (Goodfellow, Bengio & Courville, 2016). Autoencoders are fairly common in the contemporary research (Alsheikh, Niyato, Lin, Tan & Han, 2016; Hinton & Salakhutdinov, 2006; Lv, Duan, Kang, Li & Wang, 2015). A study by Lv et al. (2015) proved that stacked autoencoders can out perform Backpropagation NN (BP NN), Random Walk (RW), Support Vector Machine (SVM), and Radial Basis Function (RBF) models. Traditionally, autoencoders are mostly used in the image domain, with data sets such as hand-written digit recognition MNIST training set (LeCun, Bottou, Bengio

& Haffner, 1998) and the Olivetti face data set[2]. However implementations in the language domain, such as studies based on the Reuter Corpus[3], bilingual word representations (Chandar et al., 2014), and word meta-embeddings (Bollegala & Bao, 2018) do exist.

2.10 Transfer Learning

Transfer learning is a machine learning technique, mainly employed when there is a classification task in a domain of interest with a scarcity of data, while another related domain exists containing sufficient training data (Pan & Yang, 2010). It is possible for these data sets to be in different feature spaces or follow different data distributions. The basic transfer learning process is shown in Fig 6.

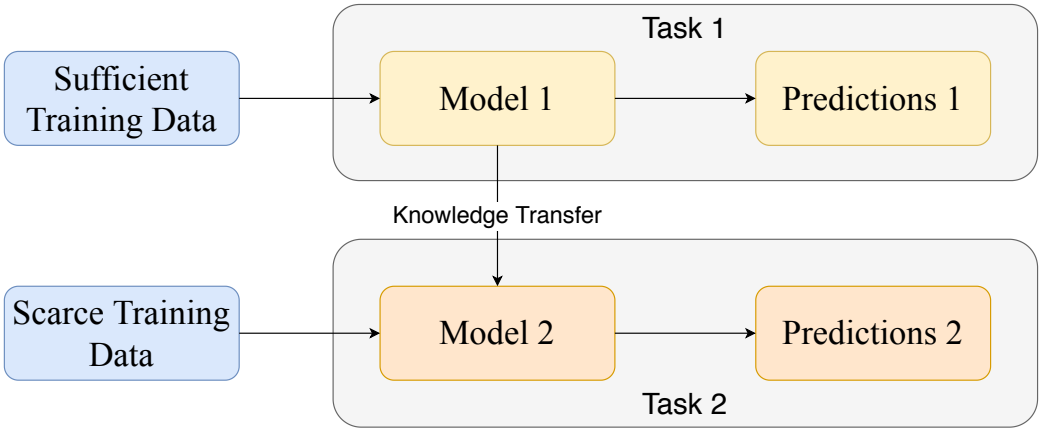


Figure 6. Basic Autoencoder Transfer Learning Process

The task employed in this work uses transfer learning in such a way that source and target domains are the same, while the source and target tasks are different but related, by the definition given by Pan and Yang (2010); given this, it is possible to declare that this methodology is based on the principals of *inductive transfer learning*. In the NLP domain, transfer learning is commonly used for the task of document

[2]http://www.cs.nyu.edu/~roweis/data.html

[3]http://trec.nist.gov/data/reuters/reuters.html

classification (Fung, Yu, Lu & Yu, 2006; Al-Mubaid & Umair, 2006; Sarinnapakorn & Kubat, 2007) and sentiment analysis (Blitzer, Dredze & Pereira, 2007; Gamage et al., 2018).

2.11 Inconsistency Detection

Inconsistency finding in text is mostly a field researched within NLP for the education domain. This has brought to light a number of methods. The first among them is based on the identification of coincident words and n-grams (Lin, 2004). While this method is adequate for automatic text grading, which is based on evaluating characteristics such as the fluency of the text, it is not suitable for the application in Chapter V, due to each of the abstracts being independent documents and not descriptions of nor summarizations of a source document. The second method is the popular NLP technique, Latent Semantic Analysis (LSA) (Foltz, Laham & Landauer, 1999; Franzke & Streeter, 2006). Here also, the vector representations of the students' documents are matched against those of a gold standard (i.e., a correct text). This approach would have been very difficult to scale for the use case in Chapter V, where all abstracts are compared against each other. The third method is based on Information Extraction (IE) (Brent, Atkisson & Green, 2010; Gutierrez, Dou, Fickas, Wimalasuriya & Zong, 2016; Mitchell, Russell, Broomhead & Aldridge, 2002). It intends to capture the underlying semantics of the text. Given that the objective of the use case in Chapter V matches well with that intention, we move in that direction. Out of the IE studies, the closest one to the use case in Chapter V is the one proposed by Gutierrez et al. (2016).

But in many ways, our methodology is significantly different than that of Gutierrez et al. (2016). That difference exists even though both approaches have inconsistency finding in common. The main difference is the fact that in Gutierrez et al. (2016), the

inconsistencies were found by adding the discovered triplets to the existing ontology and running reasoners on it, to see if the ontology has become inconsistent. Our method in Chapter V, on the other hand, uses the ontology as a tool in information extraction, as per the concept of OBIE, and does the inconsistency detection outside.

2.12 Rumour Detection

The rumour detection task has been approached on three fronts, according to Cao et al. (2018): feature engineering, propagation-based, and deep learning. In the *feature engineering* approach, posts are transformed into feature representations by hand-designed features, and they are sent to a statistical model to be classified. In addition to textual information, structural evidences (Castillo, Mendoza & Poblete, 2011; Yang, Liu, Yu & Yang, 2012) and media content (Gupta, Zhao & Han, 2012) are also utilized. Given that this approach depends heavily on the quality of the hand-designed feature sets, it is neither scalable, nor transferable to other domains. The *propagation-based* approach is built on the assumption that the propagation pattern of a rumour is significantly different to that of a non-rumour. It has been deployed to detect rumours in social networks (Ma, Gao & Wong, 2017). However, this method does not pay any heed to the information in the post content, itself. As expected, the *deep learning* approach automatically learns effective features (Ma et al., 2016; Ma, Gao & Wong, 2018b; Veyseh, Thai, Nguyen & Dou, 2019). Ma et al. (2016) claim that these discovered features capture the underlying representations of the posts, and hence, improve the generalization performance, while making it easy to be adapted into a new domain or a social medium for the purpose of rumour detection.

Our work in Chapter VI is most related to the rumour detection model on Twitter by means of deep learning to capture contextual information (Veyseh et al., 2019). However, we also derive inspiration from earlier work on the same

topic (Ma et al., 2018b), which utilized the tree-like structures of the posts, and the work in Chapter IV, which introduced the oppositeness embedding model. The early work by Ma et al. (2018b) uses Recursive Neural Networks (RvNN) for the construction of the aforementioned tree-like structures of the posts, based on their *tf-idf* representations. The following work by Veyseh et al. (2019) acknowledges the usefulness of considering the innate similarities among replies, but further claims that only considering the replies along the tree-like structure only exploits the explicit relations between the main posts and their replies, and thus ignores the implicit relations among the posts from different branches based on their semantics. Under this claim, they disregard the tree-like structure entirely. In our work, we preserve the idea of considering semantic similarities to discover the implicit relationships among posts, as proposed by Veyseh et al. (2019). However, we augment the model and re-introduce the explicit relationships proposed by Ma et al. (2018b), in a balancing of information between implicit and explicit. Further, we note that all these prior works have been solely focused on the similarity among the posts and have ignored the oppositeness metric. To the best of our knowledge, we are the first to utilize oppositeness information in the rumour detection task.

CHAPTER III

SEMANTIC OPPOSITENESS MEASURE

3.1 Introduction

Semantic oppositeness, as discussed in Chapter I, is the natural counterpart of the semantic similarity function. It yields the degree to which two concepts oppose each other in a given domain for the same purpose of confidence calculation in text mining. In this chapter, we introduce a new semantic oppositeness measure to be used to calculate the oppositeness between two words. We illustrate how this novel semantic oppositeness measure is superior to the antonym method and to the naïve similarity inverse method. The work of this chapter is adopted primarily from two collaborative papers that were published at two conferences and one collaborative journal paper under review.

The first conference paper, de Silva et al. (2017), was published at the *8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* and was composed by myself, Dejing Dou, and Jingshan Huang. The journal paper currently under review is an extension of the conference paper. As such, it too was composed by myself, Dejing Dou, and Jingshan Huang. The contributions of these papers are covered in Section 3.2 and Section 3.3 of this chapter. As lead author, I developed and implemented the contributed techniques, and I wrote the majority of the text contained in this chapter. Dejing Dou provided valuable guidance towards the motivation and application of this work. Jingshan Huang helped verify the results obtained using the proposed Semantic Oppositeness model on the OMIT project (Huang, Dang, Borchert, Eilbeck et al., 2014), on which he is working, and which is covered in other publications of which I was a co-author (Huang et al., 2016a; Huang et al., 2016b). This chapter only discusses the formalization of the Semantic

Oppositeness measure discussed in the study (de Silva et al., 2017). The concrete use-case which was involved in that study is discussed in Chapter V.

The second conference paper, de Silva and Dou (2019), was published at the *30th International Conference on Database and Expert Systems Applications* and was composed by myself and Dejing Dou. The contributions of this paper are covered in Section 3.4 to Section 3.8 of this chapter. As lead author, I developed and implemented the contributed techniques, and I wrote the majority of the text contained in this chapter. Dejing Dou provided valuable guidance towards the motivation and application of this work. This chapter only discusses the extensions and adaptations of the Semantic Oppositeness measure discussed in the study (de Silva & Dou, 2019). The semantic oppositeness embedding introduced in that paper is discussed in Chapter IV.

There is also a fundamental contribution to this chapter and the two above works from my earlier paper (de Silva, 2015a), in which I have conducted a semantic similarity measure comparison. The published result of this comparison is utilized in the above two papers and in extension, this chapter.

3.2 Semantic Oppositeness in Light of Semantic Similarity

First, the word pair is checked for similarity by Wu and Palmer (1994) semantic similarity measure (sim), as shown in Equation 3.1, where c_1 and c_2 are variables which can be utilized for applications of phrases, as discussed in Chapter V. However, for an application of similarity between two words, c_1 and c_2 can be set to the same constant, such as 1. The reasoning for selecting of Wu and Palmer (1994) for semantic similarity, over the other methods, stems from the earlier work (de Silva, 2015a), in which we comprehensively compared various semantic similarity measures.

$$simil = \frac{sim(w_1, w_2)}{c_1 + c_2} \tag{3.1}$$

Checking for oppositeness is not as straightforward as finding similarity. First, it should be noted that a simple antonym system, to be used in lieu of oppositeness, would be ill-suited for the requirement. This is because, while all words that are antonyms to a given word are, in fact, indicating an oppositeness, all words that indicate an oppositeness are not necessarily antonyms of each other. To overcome this, we need a value on a continuous scale, similar to that of the similarity measure discussed above. Given that the word similarity is between 0 to 1, as mentioned in the Section 2.3, it is possible to naïvely assume that just finding whatever the similarity value would be, and taking its complement, suffices for finding the oppositeness. This, sadly, is not the case. What this means is, semantic difference is not the same as semantic oppositeness.

We demonstrate this with the following example. Assume we have the word *increase*, in one hand, and the words *expand*, *decrease*, *change*, and *cat* on the other hand, to be checked against *increase* to see which one of the given words is the most contradictory in nature to the word *increase*. A simple antonym system will report *decrease* to be the antonym of *increase*. But it will report all the rest of the words under the umbrella term; *not-antonym*. Obviously, that is not an adequate result.

In comparison, a human would look at these words and see that the word *cat* is irrelevant here. It is neither similar to nor different from *increase*. In fact, the meaning of *cat* is orthogonal to the meaning of *increase*. Next, the human might point out that the word *expand* is semantically similar to the word *increase*. Both words are discussing adding to an amount that already exists. The word *decrease*, the human might say, is the antonym of the word *increase*. Finally, the human would

argue that, *change* should sit somewhere between *increase* and *decrease*, because it can go either way; given that a *change* can mean a *increase* or a *decrease* depending upon the context in which it was used. Therefore, *change* is not completely irrelevant to the meaning of *increase*, like *cat* is. Thus, it is possible to use this as the golden standard, to order these words in a way that each of these (or at least the opposite words) are easily identifiable. We demonstrate this expected optimal word order with demarcations in Fig 7.

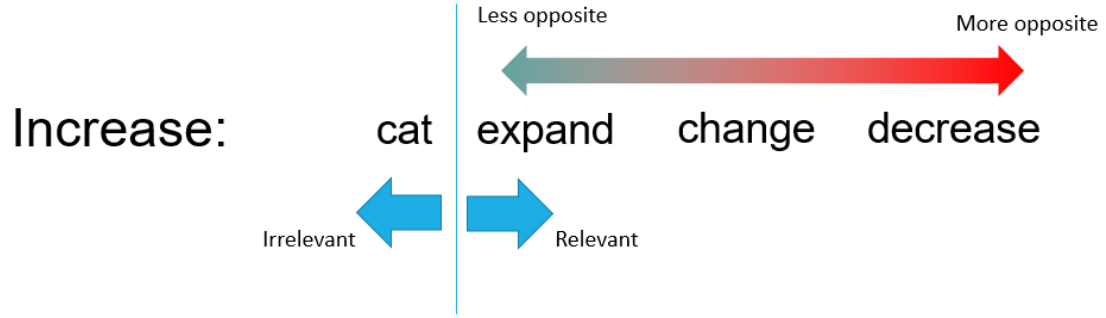


Figure 7. Expected optimal word order of *expand*, *decrease*, *change*, and *cat* in respect to the word *increase*

If one decides to use the naïve approach and take the inverse of the calculated similarities, one would get the result shown in Table 3.

Table 3. Naïve Method to Find Oppositeness

	expand	*decrease*	*change*	*cat*
Similarity to *increase*	0.80	0.75	0.46	0.25
1−**Similarity**	0.20	0.25	0.54	0.75

If the words are sorted by increasing difference, according to the calculated values in Table 3, the word order is *expand*, *decrease*, *change*, and *cat*. This is not the desired outcome. If this method is used, and a threshold is introduced to determine *decrease* as an opposite of *increase*, automatically *change* and *cat* also become

opposites of *increase*, rendering an erroneous result. Given this issue, instead of the naïve approach, we introduce the *Basic Oppositeness Measure* as shown is section 3.3.

3.3 Basic Oppositeness Measure

First, for each of the pair of words, the lemmas are extracted. The concrete lemma extraction process by means of Manning, Surdeanu, Bauer, Finkel et al. (2014) is discussed in Chapter V, with the first use-case. But in this chapter where we discuss the theory of the process, word-to-lemma conversion is taken as done by the function shown in the simple Equation 3.2 where W_i is the word and L_i is the obtained lemma.

$$L_i = lemma(W_i) \tag{3.2}$$

Following the convention in Equation 3.2, let us call the lemmas of the words W_1 and W_2, as L_1 and L_2 respectively. In the cases where the word does not yield a lemma, the word itself is used as its own lemma. For each lemma, all the *synsets* relevant for each of the word senses are extracted from English WordNet. We have discussed the structure of Synsets in Section 2.2. Given that a word might have many senses, this is a *one-to-many* mapping.

Next, for each synset, the list of antonym synsets is collected using the antonym feature of WordNet. Given that a word sense can have many antonyms in various contexts, this is, yet again, a *one-to-many* mapping. All the retrieved antonym synsets for one original lemma are put into a single list. Each of the words in each of the synsets in the said list are then taken out, to make a word list. Yet again, this is a *one-to-many* mapping, given that each synset has one or many words in it.

The resultant word list is then run through a duplicate-remover. This is the first reduction step in the antonym process so far. We name the antonym list of L_1 as a_1, and the antonym list of L_2 as a_2. The number of items in a_1 is n, while the

number of items in a_2 is m. Next, each antonym of L_1 is checked for similarity against the original L_2, and the maximum difference is extracted as dif_1 as shown in equation 3.10. Similarly, each antonym of L_2 is checked for similarity against the original L_1, and the maximum difference is extracted as dif_2, as shown in equation 3.4.

$$dif_1 = max(sim(L_2, a_1(1)), sim(L_2, a_1(2)), ..., sim(L_2, a_1(n))) \tag{3.3}$$

$$dif_2 = max(sim(L_1, a_2(1)), sim(L_1, a_2(2)), ..., sim(L_1, a_2(m))) \tag{3.4}$$

Once dif_1 and dif_2 are calculated, the overall difference, dif is calculated using Equation 3.5. Note here that c_1 and c_2 discussed in Equation 3.1 are present in Equation 3.5 for the purpose of consistency in cases where phrases are considered. But in any other application, as mentioned in Section 3.2, they would be set to the same constant, such as 1. Table 4 shows the results of the dif values for the same example as Table 3 by extending the latter.

$$dif = \frac{\frac{dif_1}{c_1} + \frac{dif_2}{c_1}}{2} \tag{3.5}$$

Table 4. Oppositeness With dif

	expand	*decrease*	*change*	*cat*
Similarity to *increase*	0.80	0.75	0.46	0.25
1−**Similarity**	0.20	0.25	0.54	0.75
dif **to** *increase*	0.63	1.00	0.72	0.25

If the words are sorted using dif in the increasing order, they would be *cat*, *expand*, *change*, *decrease*. We have reached the expected order, where first we have the irrelevant word, then the most similar word, next the neutral word, and finally the opposite word. However, still, the spread of words is not the optimum. This can

be seen from the gap between each pair of words in the above sorted order. It is 0.38, 0.09, 0.28 in order. What is needed is a way to magnify the difference values of the relevant opposite words, while shrinking the space taken up by irrelevant words, so that the threshold line can be comfortably drawn.

With both the dif and $simil$ values at hand, it is possible to calculate the oppositeness, fulfilling the above condition. Before moving on to the equation, it is prudent to look at the example on the first row of Table 4, once more. The words there are being compared to the word *increase*. As per the above discussion on the golden standard for this, the similarity measure correctly shows that *expand* and *decrease* are in the shared context of *increase*. Semantically, this implies that entities that can *increase* can also *expand* or *decrease*. They can also *change*; hence, the value for *change* comes next. But it is not as close as the previous two, because the word *change* can apply in a context that is very different from a context which is valid for *increase* just the same as it could appear in the same context. Finally, there is the value for *cat*, which is an irrelevant concept. What is observed from this is the fact that, as suggested in Section 2.1.1, the more semantically similar the two words are, the difference value has to be magnified proportional to that closeness. When the two words become less similar, the difference value has to be penalized. The equation 3.6 is introduced to calculate oppositeness where W_{no} and W_{yes} are hyper-parameters chosen to mitigate the inherent bias towards declaring that any existing two words have a considerable non-zero similarity, as observed in Section 2.3. As such, it should be noted here that the hyper-parameter W_{no} is always several degrees larger than the hyper-parameter W_{yes}. Figure 8 shows the 3D plot for the equation, and Figure 9 shows the Contour plot for the same. The power relationship between $simil_T$ and dif_T still upholds the *minimal difference with maximal similarity* principle which

was discussed in the literature in Section 2.1.1 while also supporting the *Irrelevancy* principle which was discussed in the literature in Section 2.1.2.

$$oppo = dif_T^{\left(0.5 * \frac{W_{no}}{W_{yes}} * simil_T + 1\right)} \tag{3.6}$$

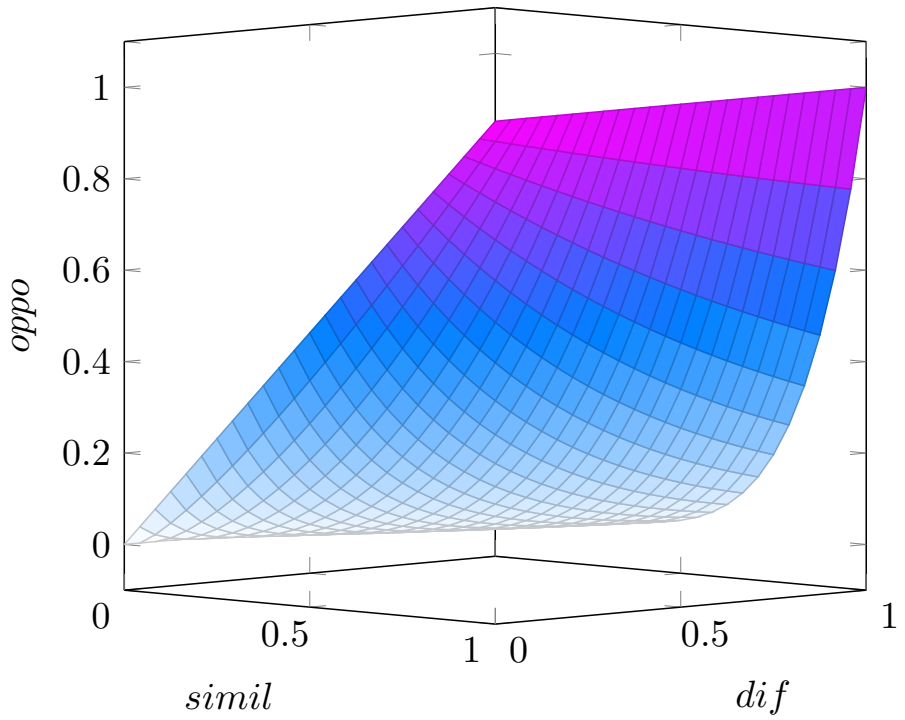


Figure 8. Oppositeness function: 3D plot

As evident by Fig 8 and Fig 9, in higher word similarities ($simil_T$), the differences (dif_T) also have to be very high for the final *oppo* value to be high. In lower $simil_T$ range, *oppo* becomes closer and closer to being directly proportional to dif_T and achieves it when $simil_T$ becomes zero. This quality, in this example, effectively pushes *decrease* farther away from *increase* than others. Values after this transformation are shown in Table 5.

Again, in Table 5, the word order, in increasing oppositeness, is; *cat*, *expand*, *change*, *decrease*, just as it was in Table 4. However, the scaled gaps between the

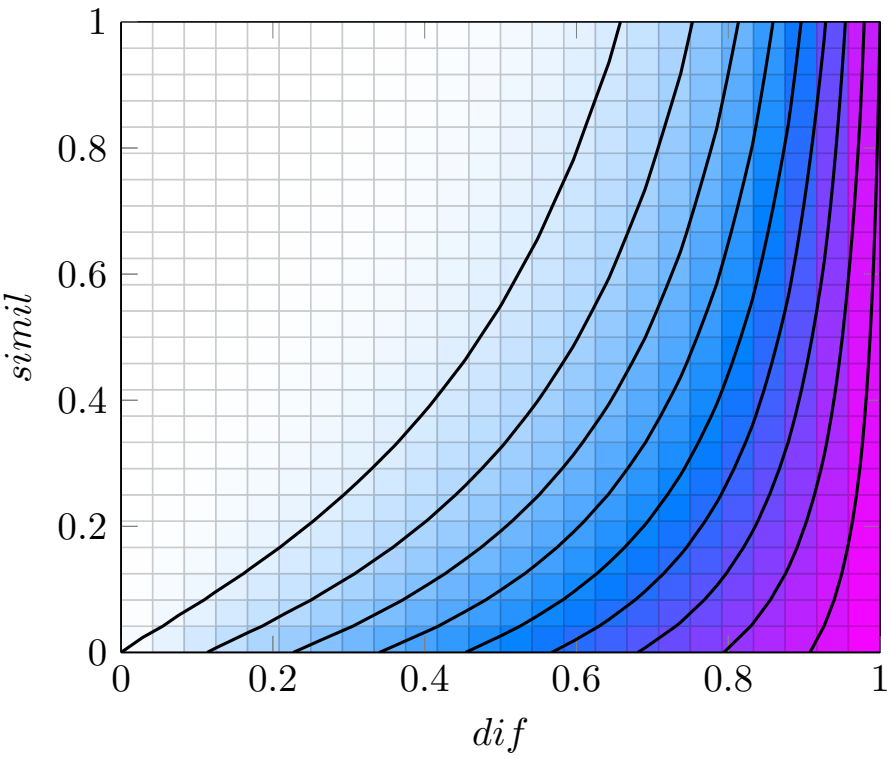


Figure 9. Oppositeness function: Contour plot

Table 5. Oppositeness With *oppo*

	expand	*decrease*	*change*	*cat*
oppo **to** *increase*	0.050	0.200	0.098	0.022
max scaled to 1	0.250	1.000	0.490	0.110

words are respectively 0.14, 0.24, 0.51. Thus, now, the antonym word is placed clearly apart from the rest of the words. The difference between the near-synonym *expand* and neutral word *change* is more prominent (distance 0.25 and 0.49 from *increase* compared to 0.63 and 0.72 in the case shown in Table 4). The irrelevant word *cat* has been pushed further downwards. The pushing of the irrelevant words serves well to further enhance the utility of the result by widening the gap between the relevant and irrelevant words, and thus making potential errors at a decided irrelevancy threshold to be scarce, as discussed above.

3.4 Improving the Calculating of Oppositeness

In this section we will discuss further enhancement of the oppositeness measure which was built in Section 3.2 and Section 3.3. The said oppositeness measure will here-onward be referred to as the *original oppositeness measure* or as *OOM*. In the following subsections, we will show how the *original oppositeness measure* can be enhanced to handle words which do not have direct antonyms to query and factor-in to the oppositeness calculation.

3.5 Alterations to the Original Measure

Before moving on to altering the *original oppositeness measure*, we introduce a few alterations to notation and equations established in Section 3.2 and Section 3.3. Further, we overlay the candidate words of the running example, *cat*, *expand*, *change*, *decrease* on the contour plot, for the benefit of the demonstration of subsequent alterations and comparisons. However, given that the intricacies of the functionality of *original oppositeness measure* have been adequately discussed in Section 3.2 and Section 3.3, in this subsection, we only discuss what components have been brought forward as they were and what components have been altered or redefined. The functionalities of these components are the same as they were in Section 3.2 and Section 3.3.

The first component, which is needed to calculate the oppositeness between two given words, w_1 and w_2, in the algorithm proposed by the *original oppositeness measure*, is the weighted semantic similarity as shown by Equation 3.1. Here, we introduce minor alterations to the notation to accommodate future extensions and define Equation 3.7. On the case of concrete functionality, among the various semantic similarity measures available as the *sim* function, the *original oppositeness measure* picks the method proposed by Wu and Palmer (1994), which gives the similarity

between two words in the 0 to 1 range. Here, we follow the same progression and use the Wu and Palmer method as the semantic similarity measure. The argument for collapsing the length constant values, c_1 and c_2 to the value 1, in equation for $simil_{w_1,w_2}$ and all subsequent equations, and there by trivially simplifying Equation 3.7 to $sim(w_1, w_2)$ of Wu and Palmer (1994) still holds true. As does the rationale of showing the said variables on the Equation for the purpose of utilities which extend beyond words.

$$simil_{w_1,w_2} = \frac{sim(w_1, w_2)}{c_1 + c_2} \tag{3.7}$$

For the *difference* component of the oppositeness calculation, it is needed to calculate the lemma of the given words and then obtain the antonym set of all the possible senses of the given word. We redefine Equation 3.2 to Equation 3.8, which reflects the obtaining of the lemma set better.

$$L_w = \Big\{ lemma(w) \Big\} \tag{3.8}$$

Further, we define Equation 3.9 to show obtaining of the antonym set, which was not mathematically expressed in the process of derivation of the *original oppositeness measure*. But since such representation is needed for the subsequent steps, we introduce it here.

$$A_w = \Big\{ antonyms(w) \Big\} \tag{3.9}$$

With the formal definitions of obtaining antonyms in equation 3.8 and equation 3.9, we redefine the Equation 3.5 for calculating difference used in the *original study*, as shown in Equation 3.10.

$$dif_{w_1,w_2} = \arg\max_{a_i \in A_1} \big(sim(L_2, a_i)\big) \tag{3.10}$$

The final relative difference equation we use is conceptually almost identical to the Equation 3.5 proposed in the *original oppositeness measure*. However, a few alterations are made, to accommodate the cumulative changes we have enacted up to the Equation 3.10. The final relative difference calculation is performed as shown in Equation 3.11, where $P = \{(w_1, w_2), (w_2, w_1)\}$.

$$reldif_{w_1,w_2} = \underset{(i,j)\in P}{avg}\Big[\arg\max_{a_k \in A_i} \big(sim(L_j, a_k)\big)\Big] \tag{3.11}$$

The *original oppositeness model* of de Silva et al. (2017) is built on the principle that the oppositeness value of two words that are highly similar should be more correlated with their difference value than that of two words that are less similar. This property is preserved by the Equation 3.6. In this step, we incorporate the alterations that we have have obtained from Equation 3.7 to Equation 3.11 into Equation 3.6 and obtain Equation 3.12. A notable difference at this point is the power scaling constant K, which is determined by the previous hyper-parameters W_{no} and W_{yes} along with the other constants that were in Equation 3.6. Note that the efforts taken in Section 3.3, to circumvent the bias observed in Section 2.3, are brought forward to this equation as well.

$$oppo\_ori_{w_1,w_2} = reldif_{w_1,w_2}^{\left(K * simil_{w_1,w_2}+1\right)} \tag{3.12}$$

Further, we bring forward the contour plot visualization as used by the *original oppositeness model* in Fig 9 and add an overlay of the placement of the four example words (*expand*, *decrease*, *change*, and *cat*) in relation to the word *increase* taken

from Table 5, for the ease of explanation of the subsequent alterations and additions we enact upon the basic algorithm. The altered contour plot visualization is shown in Fig 10.

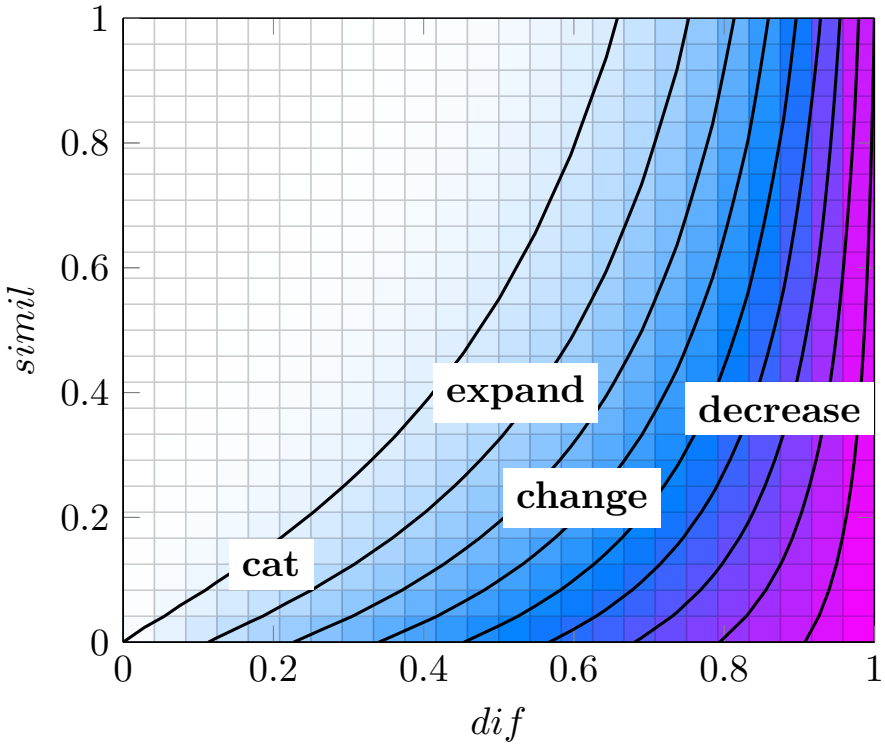


Figure 10. Oppositeness function: *OOM* Contour plot with example overlay

3.6 Antonym Dependency of the Original Oppositeness Measure

The weakness of the original model is the heavy dependence on the antonym property of the candidate words to calculate the difference. This weakness did not affect the performance of the application discussed by de Silva et al. (2017), because that study was comparing the oppositeness between the relationship component extracted from triples from medical abstracts as discussed in Chapter V. In that application, the relationship component always returns one or more action verbs. Coupled with the fact that the relevant use-case was comparing relationship strings

which contain more than one word, most of the instances translate to a high probability of encountering words with antonyms.

But in the generalized application of the algorithm, not only should the algorithm handle single-word instances; it also has to handle the possibility of that word not having an antonym. In such cases, where one or both considered words do not have antonyms, the difference value calculated by Equation 3.11 collapses to zero. This in turn further collapses the final oppositeness value calculated by Equation 3.12 to zero in cases where neither of them have antonyms; thus effectively rendering the particular data point obtained by the word pair in question, unusable. The methods we used to overcome this problem are discussed in section 3.7.

3.7 Naïve Oppositeness Measure

In the attempt to solve the problem of incalculable difference values, we turn to the naïve oppositeness measure that was replaced by the oppositeness measure proposed by the *original oppositeness model*. This naïve oppositeness measure is the simple operation of declaring the complement of similarity as oppositeness. The Equation 3.13 shows the definition of this measure. The Table 6 contains a comparative analysis of the *original oppositeness model* and the naïve method with the relevant rows brought forward from Table 4 and Table 5.

$$oppo\_nai_{w_1,w_2} = (1 - simil_{w_1,w_2}) \tag{3.13}$$

Table 6. $oppo\_ori_{w_1,w_2}$ and $oppo\_nai_{w_1,w_2}$ with $w_1 = increase$

	expand	*decrease*	*change*	*cat*
Similarity to *increase*	0.80	0.75	0.46	0.25
$oppo\_ori_{w_1,w_2}$	0.25	1.00	0.49	0.11
$oppo\_nai_{w_1,w_2}$	0.20	0.25	0.54	0.75

It is obvious from this data that the naïve method on its own does not achieve the desired properties of an oppositeness scale, as stipulated in the Section 3.2. While it is obvious that the naïve oppositeness measure is independent of the difference measure, we plot it on the same axis as the above oppositeness measure for the sake of comparison in Fig 11. The overlay of the example words in Fig 12 provides a clearer picture of the undesirability of this measure when used alone. However, at this point it should be noted that this model does not suffer from the weakness to words without antonyms that impacts the improved model proposed by the *original study*. This invulnerability stems from the similarity bias which was initially discussed in Section 2.3 and later pointed out to be a detrimental factor to the *original oppositeness model* in Section 3.3. But in the context of the naïve oppositeness measure, this bias proves to be a strength rather than a weakness.

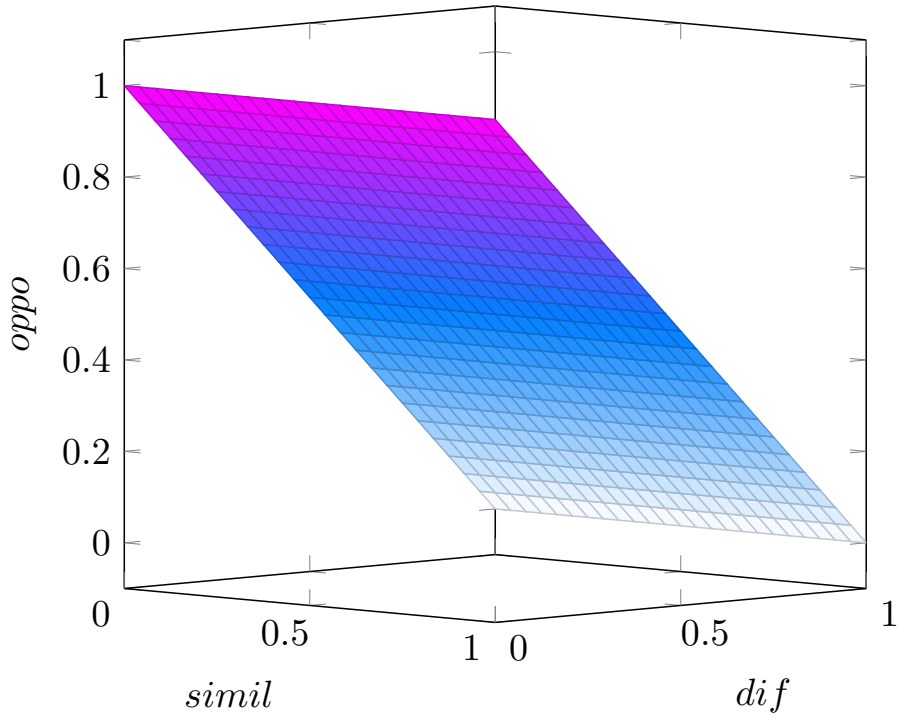


Figure 11. Naive Oppositeness function: 3D plot

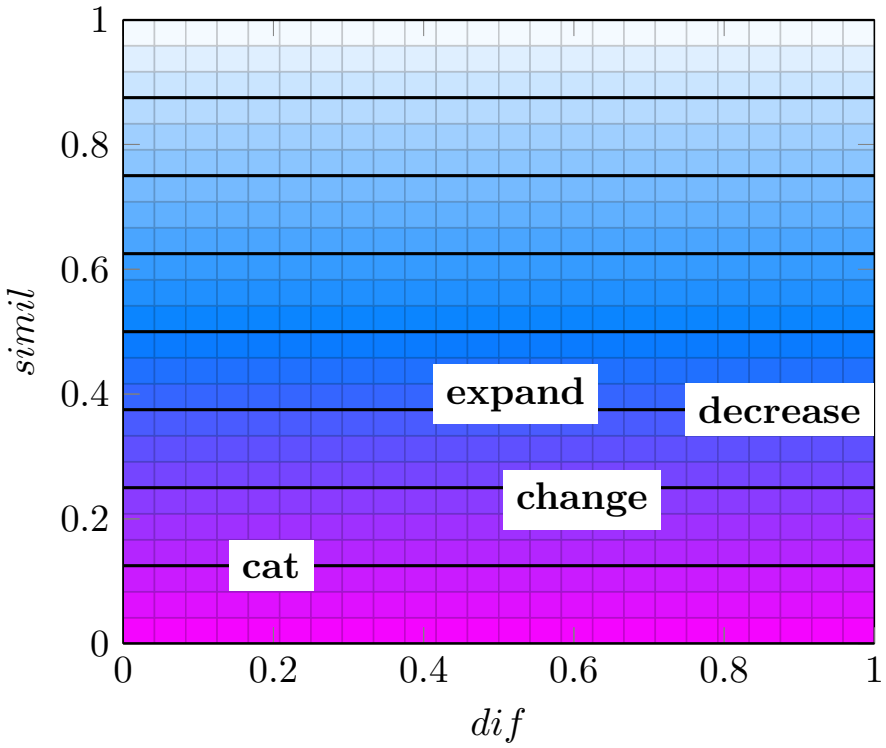


Figure 12. Naive Oppositeness function: Contour plot

3.8 Combined oppositeness model

In Section 3.7, we discussed how the naïve oppositeness model is invulnerable to the antonym dependency of the *Original Oppositeness Model* as raised in Section 3.6. However, given that the original oppositeness model is far superior to the naïve model, and that the original model is weak only at the specific instance where the difference measure is valued zero, it is vital that the two models are combined in a way that the naïve model would only take over at points where the original oppositeness model is weak. We achieved this by multiplying the naive oppositeness function with the term $(1 - reldif)$. Note here that there is no need to further multiply the original oppositeness measure with $reldif$, given that it is already positively correlated with $reldif$ as shown by Equation 3.12. If we do multiply the original oppositeness measure with $reldif$, the only alteration is the change of the constant 1 to 2. Given that the

only function of the constant 1 was to prevent powers of 0 and the oppositeness is always a relative measure, this change ultimately cancels itself. Thus, we do not perform this multiplication and unnecessarily complicate our equation.

To further fine-tune the balance between the original oppositeness measure and the naïve oppositeness measure, we introduced a hyper parameter, α. The final combined oppositeness measure is shown in Equation 3.14. Finally, we show the subtle alteration brought about by this improvement in the familiar visualization in Fig 13. In the example visualization, we have set α to 0.9. While it was needed to set α to this value for the purpose of showing a difference in the graphs discernible to the human eye, in practice, it was observed that the α value should be kept at 0.99 or higher, to prevent the naïve oppositeness measure from negatively affecting the overall calculation at points where the difference value is greater than zero. Note that the compliance to the *minimal difference with maximal similarity* principle and *Irrelevancy* Principle which was present in Equation 3.6 has been brought forward without harm to Equation 3.14. Thus, our methodology still conforms to the observations in the literature discussed in Sections 2.1.1 and 2.1.2.

$$oppo\big(w_1, w_2\big) = \alpha * oppo\_ori + (1-\alpha)(1-reldif) * oppo\_nai \tag{3.14}$$

At this point it should be noted that the reason for employing this continuous method to aggregate the two methods, rather than using a case-based approach, where the naïve oppositeness measure is only used at points where the difference measure is zero, is to make sure that the active surface of the oppositeness curve would be continuous and smooth at all points. This is important so that there would not be jarring differences in the produced oppositeness value in the comparative analysis of

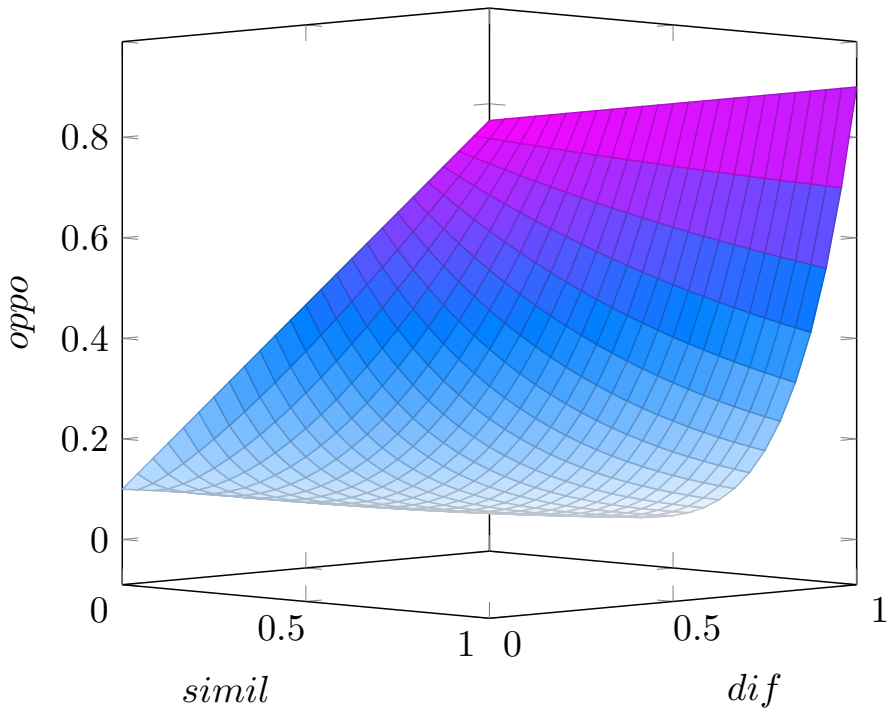


Figure 13. Oppositeness function with $\alpha = 0.9$: 3D plot

an edge case and a near edge case. On this idea, note the slight curvature present in Fig 14 in comparison with Fig 10, due to this addition.

3.9 Irrelevancy Threshold

We introduce the *irrelevancy threshold* to conform with the *irrelevancy principle* of oppositeness scale discussed in literature in Section 2.1.2. The *irrelevancy threshold* for w_1, I_{w_1} is defined as shown in Equation 3.15, where W is the set of words considered (which in most practical cases is equivalent to the entire vocabulary). $simil_{w_1,w_i}$ follows the definition in Equation 3.7, and $oppo(w_1, w_i)$ follows the definition in Equation 3.14. The idea is that any word that has smaller oppositeness against a given word w_1 than that of w_i, the most similar word to w_1, would be in the irrelevant range. This is because the relevance of oppositeness is defined only beyond the value of the most similar word.

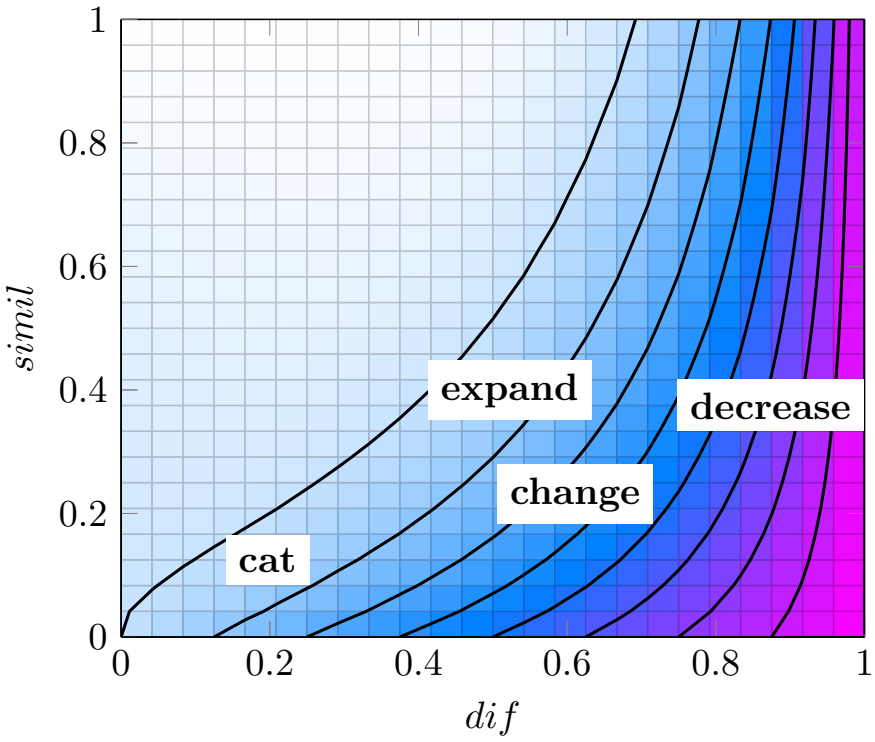


Figure 14. Oppositeness function with $\alpha = 0.9$: Contour plot

$$I_{w_1} = \underset{simil_{w_1,w_i}, w_i \in W}{\arg\max} \big(oppo(w_1, w_i)\big) \tag{3.15}$$

CHAPTER IV

SEMANTIC OPPOSITENESS EMBEDDING

4.1 Introduction

Despite being both innovative and useful, the oppositeness calculation algorithm we created in Chapter III is very computationally intensive. Therefore, in large text mining tasks it would significantly slow down the process. It is in an attempt to avoid such computational complexity that most Natural Language Processing (NLP) tasks involve the incorrect generalization of reducing semantic oppositeness to antonymy (Paradis et al., 1982) or inverse of semantic similarity (de Silva et al., 2017).

In the case of semantic similarity, this problem was overcome with the rise of the word embedding systems as discussed in Section 2.8. Tasks which used to be a complex set of word similarity calculations (Jiang & Conrath, 1997; Wu & Palmer, 1994) were reduced to simple K-NN look-ups in vector spaces (Mikolov et al., 2013b; Pennington et al., 2014). The objective of this chapter is to formulate obtaining such an embedding for semantic oppositeness, so that text mining applications that involve semantic oppositeness can become more efficient, akin to the transformation undergone by semantic similarity.

The method introduced in this chapter first autoencodes word vectors, then it transfer-learns the decoder half of the deep neural network by using values obtained by the oppositeness algorithm discussed in Chapter III as the target. Hence, at the end, the trained deep neural network can generate oppositeness values for given word pairs faster and using fewer resources than the original oppositeness algorithm, with the added benefit of the possibility of handling word pairs that could not be handled by the original oppositeness algorithm due to its limitations.

In addition to the main research contribution of introducing an embedding for the semantic oppositeness function, in this chapter we also introduce a novel unanchored vector embedding approach and a novel *inductive transfer learning* (Pan & Yang, 2010) process based on autoencoders (Goodfellow et al., 2016), which utilizes both the learnt embeddings and the learnt latent representation as discussed in Section 2.9.

The work of this chapter is adopted primarily from a collaborative paper that was published at a conference. The paper, de Silva and Dou (2019), was published at the *30th International Conference on Database and Expert Systems Applications* and was composed by myself and Dejing Dou. As lead author, I developed and implemented the contributed techniques, and I wrote the majority of the text contained in this chapter. Dejing Dou provided valuable guidance towards the motivation and application of this work. This chapter only discusses the semantic oppositeness embedding process introduced in the paper, along with the relevant experiments and results. The extensions and adaptations of the Semantic Oppositeness measure discussed in the study (de Silva & Dou, 2019), which lay the groundwork for the semantic oppositeness embedding process of this chapter, were covered in Section 3.4 to Section 3.8 of Chapter III.

4.2 Semantic Oppositeness Embedding Process

Once the algorithm described in Chapter III is used to calculate the oppositeness measures for word pairs, the next step of the process is to embed them in a vector space. Embedding the oppositeness gives applications the ability to do simple K-NN queries on the vector space, instead of running the costly algorithm *OOM* each time. This section discusses the process of embedding the said oppositeness values in a vector space.

4.2.1 Minimization constraint. In an embedding process, it is important to first define what the minimization constraint is. In almost all cases, it is defined as a function to calculate the distance between the vector currently obtained by the embedded object and the vector expected to be obtained by the embedded object. However, in this study our objective is novel in the sense that for this algorithm, it does not matter where the individual word vectors map to. All that matters is the difference between two given embedded word vectors approaching the oppositeness value calculated above. Therefore, in the learning process, instead of anchoring a vector (or a context) and trying to move the target vector close to it, we can employ an algorithm to push both vectors together with no contextual attachments. Thus the minimization constraint becomes a matching of two distance scalars, rather than a minimization of the distance between two vectors. The proposed minimization constraint is given in Equation 4.1, where $||a - b||$ denotes the Euclidean distance between vectors a and b, w_s and w_t are input words, y_t and y_s are the outputs of the neural network, and the *oppo* function returns the linguistically calculated oppositeness value, as proposed in Equation 3.14. Note that we need to preserve the sign of the difference to use the unanchored training. Thus, the absolute value function (abs) is deconstructed into three cases in later steps.

$$min\Big[abs\Big(oppo\big(w_t, w_s\big) - ||y_t - y_s||\Big)\Big] \tag{4.1}$$

4.2.2 Expected Vector Calculation. The range of the minimization constraint given in Equation 4.1 is unbound. Which means that it can arguably obtain values ranging from $-\infty$ to $+\infty$. In practice, this is bounded by the upper and lower limits of the values obtained by the embedded vectors. However, in either case, this large range is undesirable for the embedding task. Therefore, we define f as

shown in Equation 4.2, where f would be limited to a range of $+1$ and -1, depending on the placement of the embedded vectors in relation to the expected value, and σ indicates the standard Sigmoid function.

$$f = 2 * \sigma\big[oppo\big(w_t, w_s\big) - ||y_t - y_s||\big] - 1 \tag{4.2}$$

4.2.3 Target update rule. As mentioned in Section 4.2.2, the embedding in this study does not conform to the idea of anchoring one vector (or context) and pushing the candidate to match the expected vector. Instead, both the vectors in question are moved to make sure the oppositeness is defined by the distance between the said embedded vectors. It should be noted that to the best of our knowledge, this work was the first to utilize such an unanchored approach to word vector embedding. In this section we derive the update rule for each of the two vectors. For the simplicity of subsequent calculations, we define ΔY as $y_t - y_s$, while the expected shifts are $y_{t'} - y_t$ and $y_{s'} - y_s$.

There are three possible cases of vector placement as shown by Fig 15, Fig 16, and Fig 17. Each of these cases are uniquely identifiable by the f value calculated by equation 4.2. First, there is the case where the current embedding places the word vectors farther apart than the linguistic calculation of the oppositeness measure indicates is needed. These need to be pushed together. In these scenarios, the value of f would always evaluate as less than zero. This situation is visualized in fig 15. The notation in red shows the current embedding of the vectors, while the notation in blue shows the expected embedding.

The equation 4.3 shows the suitable update rule to calculate the embedding target $y_{t'}$, while the equation 4.4 shows the suitable update rule to calculate the embedding

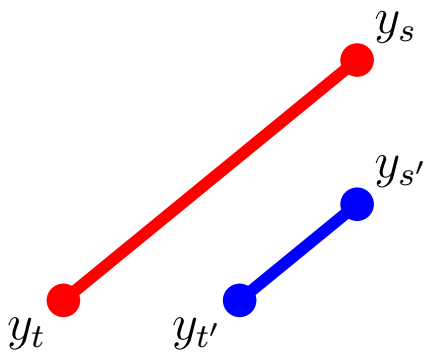


Figure 15. **Case:** $f < 0$

target $y_{s'}$. η_1 is the learning rate. Note how there are update rules for both vectors, rather than anchor one and push the other vector as embedding convention dictates.

$$y_{t'} = y_t - \eta_1 \Delta Y \tag{4.3}$$

$$y_{s'} = y_s + \eta_1 \Delta Y \tag{4.4}$$

The second case as shown by Fig 16 occurs when the current embedding of the vectors places them closer than the linguistic calculation of the oppositeness measure would indicate is expected. Similar to the above, the notation in red shows the current embedding of the vectors, while the notation in blue shows the expected embedding. These vectors need to be pushed apart. In these scenarios, the value of f would always evaluate as more than zero.

The equation 4.5 shows the suitable update rule to calculate the embedding target $y_{t'}$, while the equation 4.6 shows the suitable update rule to calculate the embedding target $y_{s'}$.

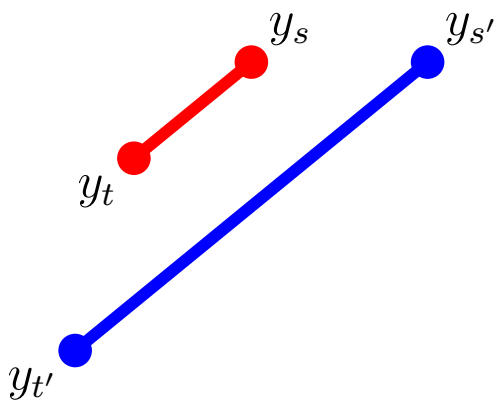


Figure 16. **Case:** $f > 0$

$$y_{t'} = y_t + \eta_2 \Delta Y \tag{4.5}$$

$$y_{s'} = y_s - \eta_2 \Delta Y \tag{4.6}$$

The final case covers the scenarios where f would evaluate to exactly zero. This case is shown by the fig 17. This occurs when the embedded vectors have obtained the expected outcome of being embedded with a distance equal to the value dictated by the linguistic calculation of the oppositeness measure, which was given by Equation 3.14.

None of the vectors need updating in this scenario. But for the sake of consistency and subsequent generalization, we declare the equation 4.7 to show the suitable update rule to calculate the embedding target $y_{t'}$ and the equation 4.8 to show the suitable update rule to calculate the embedding target $y_{s'}$.

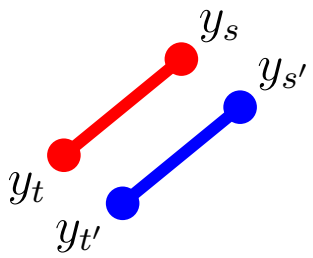


Figure 17. **Case:** $f = 0$

$$y_{t'} = y_t \tag{4.7}$$

$$y_{s'} = y_s \tag{4.8}$$

The Table 7 summarizes all update cases we have discussed with Fig 15, Fig 16, and Fig 17 for each of the vectors against the various value ranges of f.

Table 7. Case-based update rules

	$f < 0$	$f > 0$	$f = 0$
$y_{t'} - y_t$	$-\eta_1 \Delta Y$	$+\eta_2 \Delta Y$	0
$y_{s'} - y_s$	$+\eta_1 \Delta Y$	$-\eta_2 \Delta Y$	0

Finally, it is possible to combine all the above embedding target update rules together, based on the fact that they are uniquely mapped to the value of f. Thus, with consideration of the sign of f, it is possible to combine the Equation 4.3, Equation 4.5, and Equation 4.7 to obtain the generalized Equation 4.9. Similarly,

it is possible to combine the Equation 4.4, Equation 4.6, and Equation 4.8 to obtain the generalized Equation 4.10.

$$y_{t'} = y_t + f\eta\Delta Y \tag{4.9}$$

$$y_{s'} = y_s - f\eta\Delta Y \tag{4.10}$$

This translates to two forward propagation steps per each backpropagation of a word pair in the neural network. In the first forward pass, forward propagation is activated with x_t as the input, y_t as the output, and $y_{t'}$ as the expected output. In the second forward pass, forward propagation is activated with x_s as the input, y_s as the output, and $y_{s'}$ as the expected output. Then the difference between $(y_t - y_s)$ and $(y_{t'} - y_{s'})$ is calculated as the error and is used in the single backpropagation pass.

4.2.4 Autoencoder-based Transfer Learning. While the above algorithm is sound as a solution to embed words in a vector space guided by the oppositeness values, starting with fully empty or fully randomized word vectors would be counter-productive. In such an approach, our system will implicitly have to learn the word embeddings that are achieved by word embedding systems such as word2vec (Mikolov et al., 2013b) or GloVe (Pennington et al., 2014). Further, there is the initial hurdle of declaring the input (x_s, x_t) in an unambiguous manner. The solution to both of these problems is to involve an already trained word embedding model as the starting point. In this juncture, we decided to use word2vec (Mikolov et al., 2013b). This solves the second problem outright. The declaration of input (x_s, x_t) in an unambiguous manner is now a simple matter of querying the trained Word2vec model with the expectant word string.

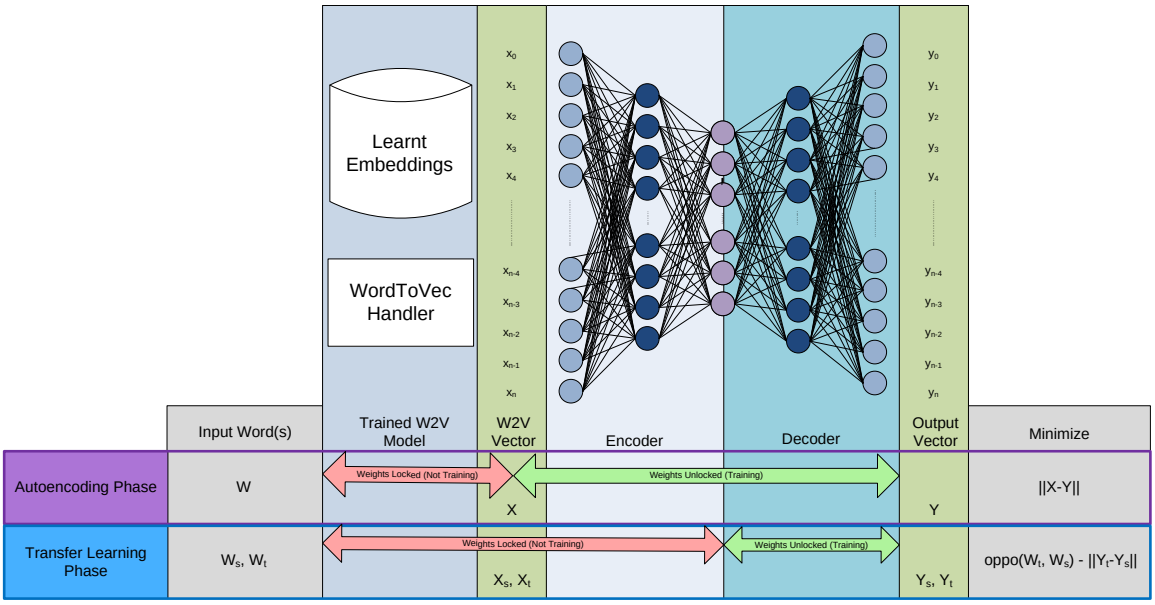


Figure 18. Overall Autoencoder-based Transfer Learning Model

Solving the first problem is not so straightforward. The objective of this step is to utilize the already existing embedding of words in word2vec to make oppositeness embedding faster. The rationale here is the fact that word2vec already clusters words by similarity; and thus, following the naïve method we discussed in above sections, it is reasonable to predict that the oppositeness embedding would be comparatively easier to achieve starting from a similarity embedding than by a random embedding or by a zero embedding. Here, note the fact that the naïve assumption was to assume that the similarity embedding trivially translates to the oppositeness embedding. We do not conform to that naïve assumption. We only claim that the similarity embedding would be *reasonably closer* to the expected oppositeness embedding, rather than to a zero or a random starting point. Therefore we propose the novel idea of applying transfer learning (Pan & Yang, 2010) on the decoder portion of the autoencoder, as introduced in Section 2.10. The proposed full learning model is shown in Fig 18.

First of all, to employ the proposed model, we should obtain a mapping from words to vectors. Among the various algorithms and models available to map words to vectors, such as Word2Vec (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014), we propose to use Word2Vec based on the wider support (especially the availability of large Google-trained data set[1]). This component of the ensemble would map a given word to a vector. Incidentally, this would become the input of our neural network. Thus, the *encoder* of the neural network model has as a similar number of input neurons to that of the feature length of the selected word embedding. Next the hidden layer of the *encoder* has a smaller number of nodes, while the latent representation layer of the *encoder* has yet an even small number of nodes.

This architecture is proposed to prevent the hidden layer and the latent representation layer from learning the identity function when the autoencoder is trained. Then by definition, the input layer size of the *decoder* is similar to the size of the latent representation layer of the *encoder*, and the hidden layer of the *decoder* has a similar number of nodes to that of the *encoder*. Finally, the output layer of the *decoder* has a number of nodes similar to that of the input layer of the *encoder* for the purpose of reconstructing the input.

The proposed model has two learning phases. The first phase of the proposed model is called the *Autoencoding Phase*. In this, we keep the word2vec model locked and the weights of the encoder and the decoder unlocked. The formal representation of an autoencoder is given in Equation 4.11, where the section $\sigma(W_1 x_i + b_1)$ correlates to the $encode(X)$ function where the W_1 and b_1 are weights and biases of the *encode* function. The $\sigma(W_1' l) + b_1'$ portion, where l represents $\sigma(W_1 x_i + b_1)$ discussed above, correlates to the $decode(l)$ function, where the W_1' and b_1' are weights and biases of the

[1]https://goo.gl/yV57W3

decode function and l is a latent representation output by the $encode(X)$ function. The learning objective of the autoencoder is to $minimize(||X - Y||)$ where X is the input vector of the encoder and Y is the output vector of the decoder. Note here that in the literature, Y is commonly referred to as X' to showcase the fact that it is supposed to be a reconstruction of the original X. However, in this work we opted to use Y for the sake of clarity of the subsequent steps where we use transfer learning instead of reconstruction (autoencoding).

$$Y_i = \left(\sigma\big(W_1'\sigma(W_1 X_i + b_1)\big) + b_1'\right) \tag{4.11}$$

In summary, during the *Autoencoding Phase* of the proposed model, the neural network learns to reconstruct a given word vector. As mentioned above, this is an attempt to utilize the learnt artifacts of a word embedding system, where related words are clustered together while unrelated words are embedded far apart.

The second phase of the proposed model is the *Transfer Learning Phase*. The transfer learning proposed in this work differs from prior work in the literature by three facts. Firstly, the transfer process applied on the autoencoder is applied, not to train yet another autoencoder model, but to map the same inputs to a different vector space. Thereby, at the end of the training process, the trained neural network is *not* an autoencoder. However, for the sake of readability of the chapter, we would continue to refer to the two components of the neural network as *encoder* and *decoder*. It is imperative that after the training, the output of the *decoder* no longer tries to reconstruct the input to the *encoder*. However, as mentioned above, given the linguistic properties, that vector will still be *reasonably close* to the input vector.

The second difference is the fact that we lock the weights of the *encoder* along with the word2vec model in the training process of this phase. While it is a given property

of transfer learning applications to lock a certain number of initial layers and train only a certain number of layers close to the output layer, usually, that choice is open-ended and unrestricted. In this study, however, we specifically lock all the layers that were previously in the *encoder* and keep the layers that were previously in the *decoder* unlocked. The rationale for this decision is as follows: The autoencoder has already learnt a latent representation of the word vectors, by using the autoencoding process, where the output is the input itself. Therefore, we can be sure of the accuracy of the learnt latent representation. The latent representation of a given vector need not be altered when the application is changed. To the best of our knowledge, this work is the first to propose this autoencoder-based *inductive transfer learning* (Pan & Yang, 2010) process to utilize both learnt embeddings and the learnt latent representation.

The third aspect that distinguishes this model from the traditional learning processes is the fact that this phase uses two forward passes to calculate the error for a single back-propagation pass. This is due to the fact that, as discussed in Section 4.2.1, the learning objective of the neural network is to achieve the minimization proposed in Equation 4.1. The mechanics of this process were further explained in Section 4.2.3.

4.2.5 Optimizations and Parallelization. We employed a number of optimization steps to minimize the word pair data set at hand to remove data points that would either slow down or hinder the oppositeness learning process. These optimizations are discussed in detail in Section 4.3.1. For the purpose of efficient training, and as a means for handling the over-fitting problem, we introduced a parallelized training model for the transfer learning portion of the methodology. This parallelization process is discussed in Section 4.3.3. The reason for discussing these components in the experiments and results section is the fact that these tweaks are introduced in more of a practicality perspective rather than a theoretical perspective.

Arguably, a person with infinite computing resources and infinite time would be able to duplicate the above discussed methodology without the proposed optimizations and the parallelization process.

4.3 Experiments and Results

4.3.1 Calculating the Oppositeness Data. For the purpose of obtaining an adequate collection of words for experimentation, we used the list available in the Linux dictionary[2]. The dictionary contained 72,186 total strings. However, it was observed that a certain portion of the strings were non-words. Further, for the sake of preserving the variety of the sample set, it was decided to replace words with their lemmas in cases where there are multiple morphological forms. This process was achieved by passing the potential word strings through the WordNet (Miller et al., 1990) lemmatizer. This yielded a reduced word list of 65,167.

Next the methodology discussed in Section 3.5 was used on the 65,167 words taken as pairs. Given that each word was considered against all other words, this resulted in 4,246,737,889 pairs of words. For each pair of words, the *similarity*, *difference*, and *oppositeness* were calculated. A few sample lines from the file of the word *increase* are shown in Example 4.1. Note the minimal value of the *similarity* slot for *advents*. This implies that the similarity measure could not give a similarity value to the pair (i.e., The pair is disjoint). Logically, this file, like all other files at this stage, has 65,167 lines.

Example 4.1 Sample Oppositeness Lines

```
increase,adrian:0.125,0.1,0.0313516
increase,adriatic:0.125,0.1,0.0313516
```

[2] /usr/share/dict/words

```
increase,advent:0.1875,0.2777778,0.088623986
increase,adventist:0.11764706,0.10526316,0.03561389
increase,advents:1.4E-45,1.4E-45,0.01
```

The above files were then processed by applying the *Irrelevancy Threshold* proposed in Section 3.9. Thus, following Equation 3.15, for each word, all pairs with oppositeness value *less than* the oppositeness value of the most similar pair were eliminated. After this reduction step, only 76,084,553 pairs out of the original 4,246,737,889 were left. This means, on average, each word contained 1168 pairs after this step, showing a reduction of 98.21%. As an example, the corresponding file of the word *increase* has only 1107 lines. This is a significant reduction in the case of potential computational load. Given that the file format stays the same as shown in Example 4.1, we do not provide a separate example here. Instead, we add the words *lion*, *zucchini*, *good*, and *bad* to the running example of *increase*, *decrease*, *expand*, *change*, and *cat* for the purpose of extended explanation. The mention map of this expanded word list after applying the irrelevancy threshold is shown in Fig 19. Here, note that some mentions are bidirectional and others are unidirectional. Obviously, before applying this threshold, it was a *complete graph*.

At this step, it should be noted that there may exist instances where it is discovered when considering the word pair (w_1, w_2), in the case of w_1, the oppositeness value of the pair clears the w_1 specific threshold, but in the case of w_2, the oppositeness value of the pair does not clear the w_2 specific threshold. This should effectively mean that the (w_1, w_2) pair should be invalidated. Thus, to further simplify the data set by reducing such redundant data points, the bidirectional filter was introduced. The bidirectional filter checked each word pair for the condition discussed above and

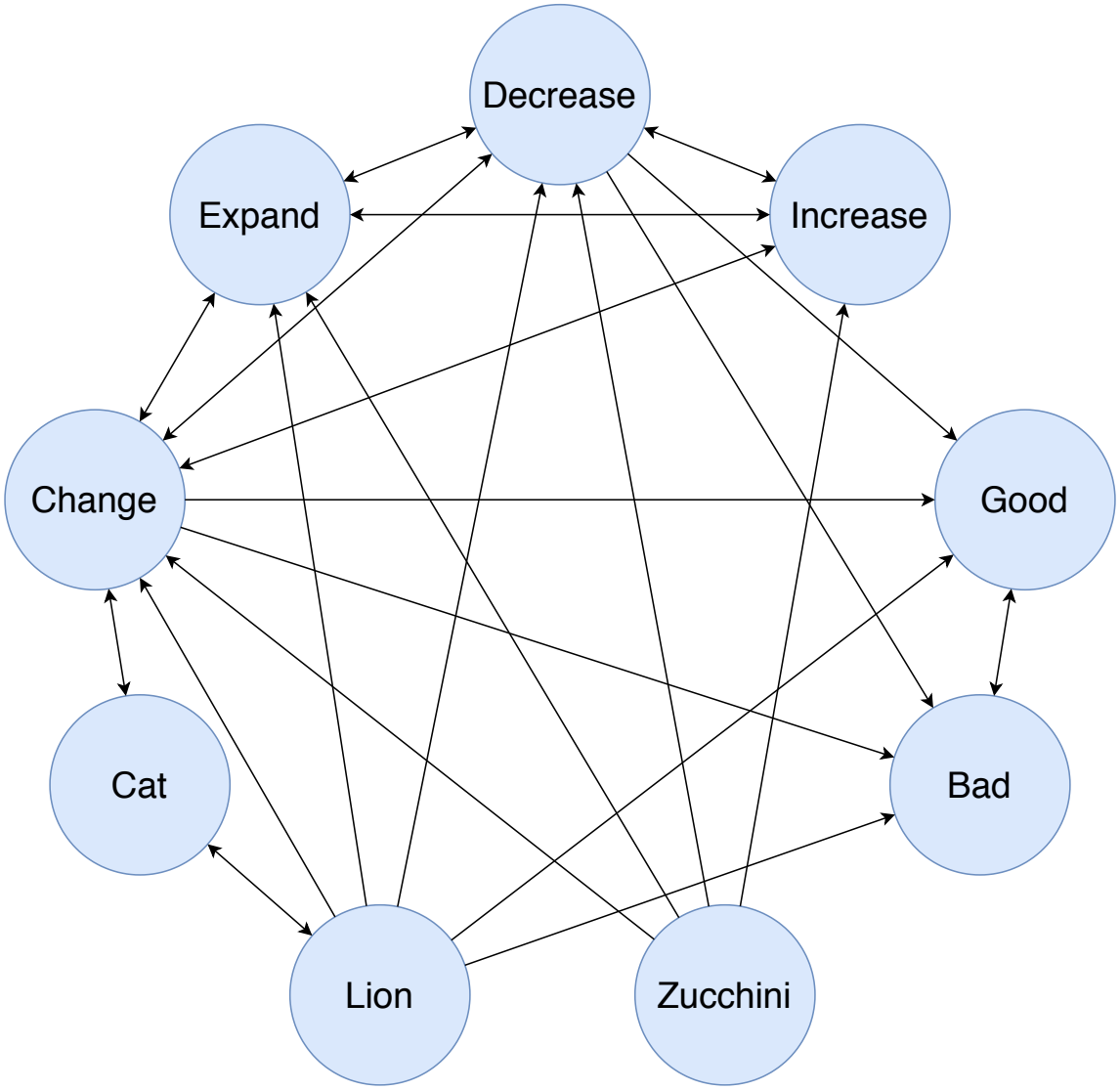


Figure 19. Mention map of the example set of words after applying the irrelevancy threshold.

removed orphaned members from the pair list of the remaining word. By the end of this reducing step, there were only 52,294,500 word pairs left. This reduced the average pair count per word to 802. In summary, this is a percentage reduction of 31.27% from the above step. For the purpose of comparison, here we report that the corresponding file of the word *increase* has only 1057 lines after this process. The

mention map of the expanded word list after applying the bi-directional threshold is shown in Fig 20.

It is evident how the words have started to break in to semi-cohesive clusters. Following that, note here that *zucchini* is no longer connected to any of the other words. This is an example of an obscure word that has lost all pairs but the self-referential pair at this point. The only entry that would remain for such a word, w, takes the constant form, as shown in Example 4.2.

Example 4.2 Only remaining line in the example obscure word w

```
w,w:1.0,0.0,0.0
```

As discussed in section 4.2.4, we used the Word2Vec (Mikolov et al., 2013b) model trained by Google[3]. This model contains 3,000,000 strings embedded to vectors with 300 features. However, these strings are not purely words; some of the strings embedded in this model are HTML tags. Further, it was observed that neither the word set mentioned above for calculating oppositeness nor the word set in the Google news Word2Vec model were a subset of the other. Hence, it was necessary to remove words that do not occur in one model from the other model for the sake of saving computational time and space. At the same time, it was decided to drop words that are only left with the self-referential pair as described above.

For the purposes of resource management, the Google-trained Word2Vec model was queried with the reduced word list of 65,167 words in 100 word batches. This is resulted in 652 batch files. Thus the word2vec embedding is reduced well below from the initial 3,000,000.

[3]https://goo.gl/yV57W3

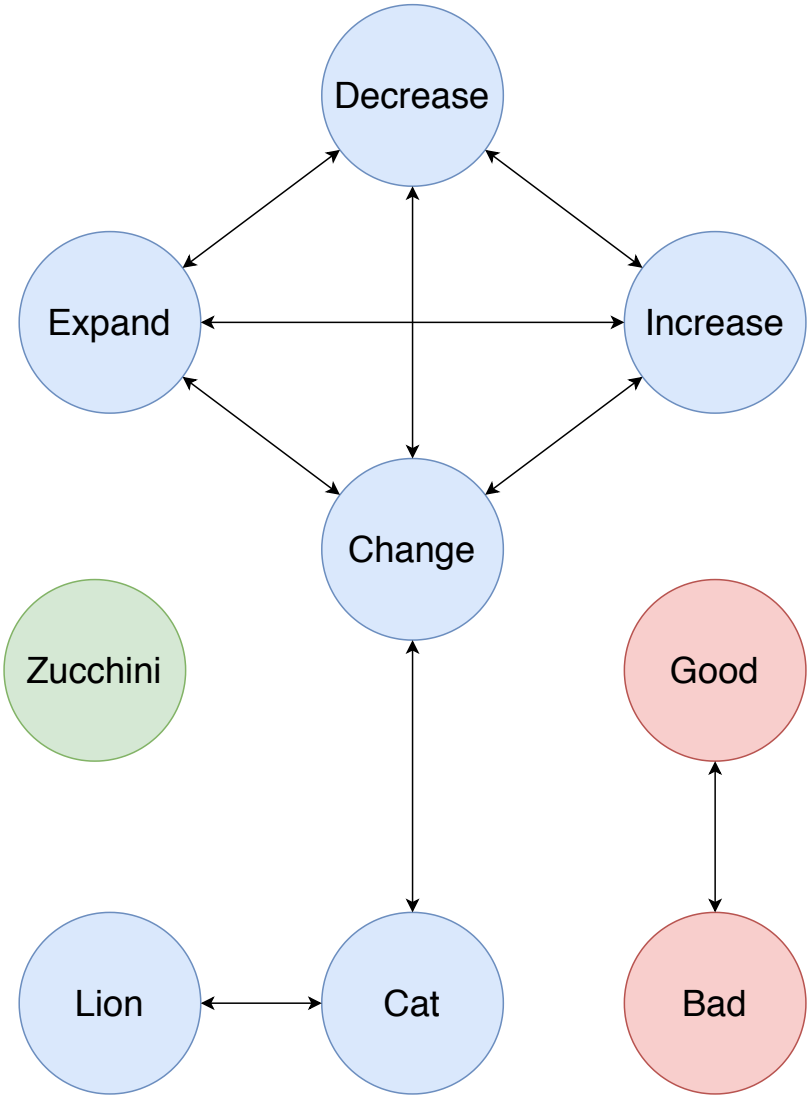


Figure 20. Mention map of the example set of words after applying the bi-directional threshold

Next, the words in these batch files which do not carry a vector were dropped from the word list. This reduction removes words that do not have a Google word2vec embedding from the system. If we were to use an untrained neural network to do the oppositeness embedding, instead of using transfer learning, we would not have had to drop these words from the experiment. But compared to the training time and resource benefits that are gained by using transfer learning, it was decided that this

is an acceptable compromise which only costs obscure words that were not captured by the Google word2vec model. As mentioned above, the purely self-referential pair words were also dropped from the word list. It is to be noted that these would be dropped regardless of whether the model is trained using transfer learning or started from scratch.

This step further reduced the word count to a more manageable count of 24,730. The total potential pair count at this point is 49,148,652, which implies that the average pair count per word has increased to 1987 from its previous value of 802. The reason for this increase of average pair count is the fact that this reduction step removed words which are disjointed or poorly connected to the oppositeness word clusters; and thus, the remaining words are the ones that are more cohesively connected to each other.

4.3.2 Autoencoding on Word2Vec Data. We used a TensorFlow (Abadi et al., 2016; Martın Abadi et al., 2015) based implementation of the two layer autoencoder proposed by Damien (2017) for the purpose of training on the MNIST Dataset (LeCun et al., 1998). The input layer was altered to have the size of 300 to match the trained model of the Google's word2vec embedding. The middle layer was of size 256, and the latent layer was of size 128. By definition, the decoder layer had an input size of 128 to match the latent layer of the encoder, then a middle layer of size 256 and finally an output layer of size 300 to match the input of the encoder layer. Following the precedent set by Damien (2017) for this particular configuration of autoencoder, we found 30,000 epochs to balance accuracy against the threat of over-fitting. By employing the multiple random restart method, we obtained a trained autoencoder with a validation accuracy of 94.83%. This is the model that we used for the next step of transfer learning.

4.3.3 Learning of Oppositeness Data. Here it was decided to use a 3 : 2 split for the training-validation set vs. test set for the transfer learning of oppositeness data. As such, the training-validation set contained 14,820 words, and the test set contained 9910 words. With the average oppositeness pair count calculated above, this yields 29,447,340 oppositeness pairs for the training-validation set and 19,691,170 oppositeness pairs to be in the test set. As mentioned in section 4.2.4, we locked the weights in the encoding layer for this step. The training was done only on the weights of the decoder layer. In this, we achieve two gains on the account of transfer learning. First and foremost, we already have the latent representation of the word2vec vectors learnt at the end of the encoder. This does not need to be re-learnt and can be transferred unaltered. Secondly, we have the decoder weights to take in a latent representation of a vector and re-construct the word2vec vector. As discussed in Section 4.2.4, it is reasonable to assume that the word2vec vector and the expected oppositeness vector will be close in the vector space. Thus, transferring the decoder weights to the system gives a more efficient starting point, compared to initiating with zero weights or random weights.

Given the large amount of training data, we decided to parallelize the training process. First the training-validation set was divided into 30 equal parts. The trained autoencoder was then cloned 30 times as well. Each of the cloned autoencoders was given the word2vec embedding as the input, along with the relevant portion of the training data to train on. This resulted in each cloned model producing a unique embedding. This parallelization architecture is shown in Fig 21.

This approach helped us in two ways. Firstly, as mentioned above, it helped us with managing the computational load. Secondly, it helped us overcome the problem of over-fitting by applying rigorous validation. In the common n-fold cross-validation

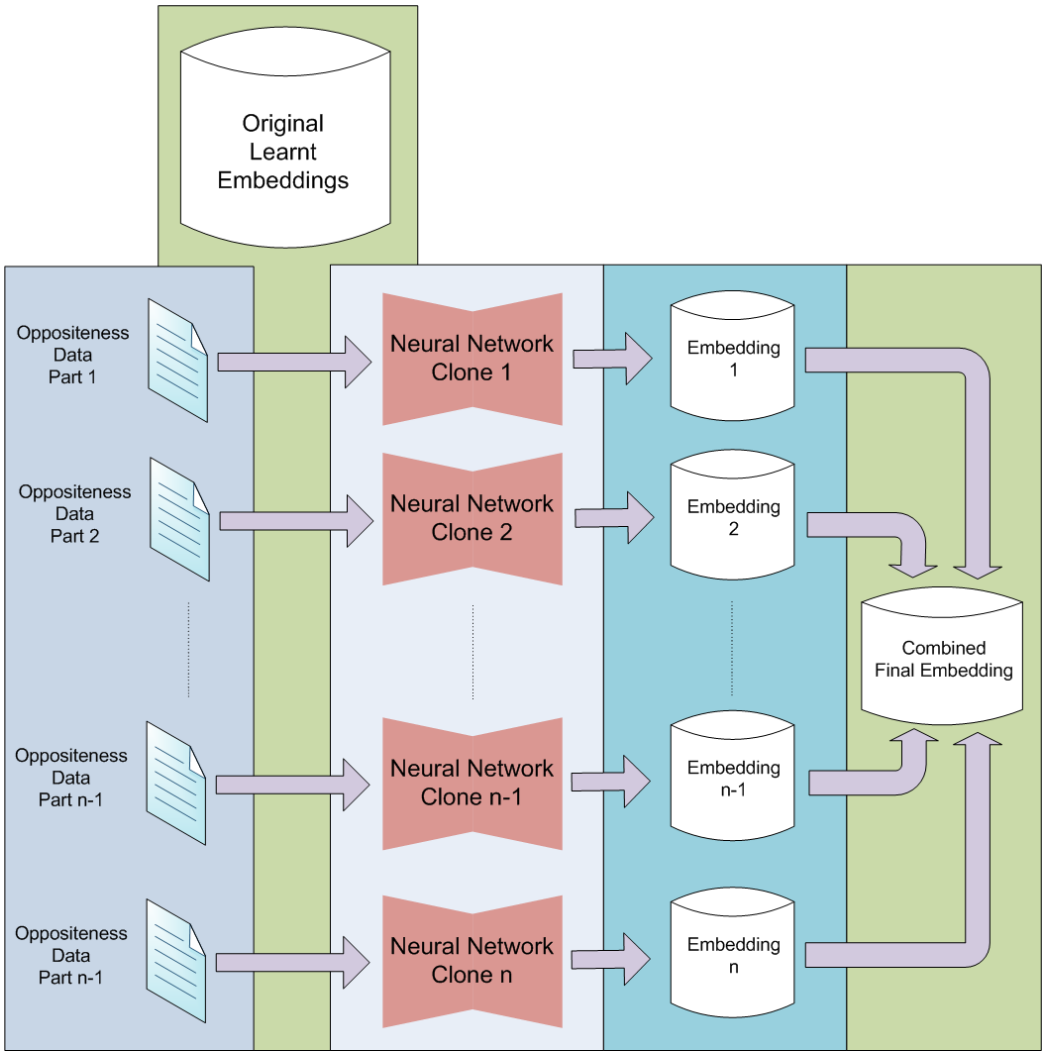


Figure 21. Parallelization architecture

method, the model is trained using $n-1$ portion of samples and is then validated using the remaining sample. But considering the fact that we intend to directly merge the results of the clones and the heavy computational workload, we opted to involve *inverse-n-fold cross validation*. As such, a single portion of data is used as the training data, and then the remaining $n-1$ portions of data are used for validation. This way, the validation process is both more rigorous and mutually independent. We report the accuracy of each of the separate clones in Fig 22. The Y axis (rows) of the Matrix corresponds to each transfer learning clone, and the x axis (columns)

corresponds to the portion of data-set. Hence, the 30 entries on the diagonal of the matrix correspond to the training accuracies, and the 870 entries on the remainder of the matrix correspond to the validation accuracies. It is observable that, while the diagonal is sightly distinguishable, some clones seem to be performing better than the others across the board. We claim that this is because of the linguistic property that some words are more central in a lexicon than others. These words might distinguish themselves by having more synonyms or by having polysemy (Apresjan, 1974; Tuggy, 1993). When the data set given to a clone has a majority of such words, it is possible to claim that the trained model would generalize better than in the case where the data set given to a clone has a minority of such words.

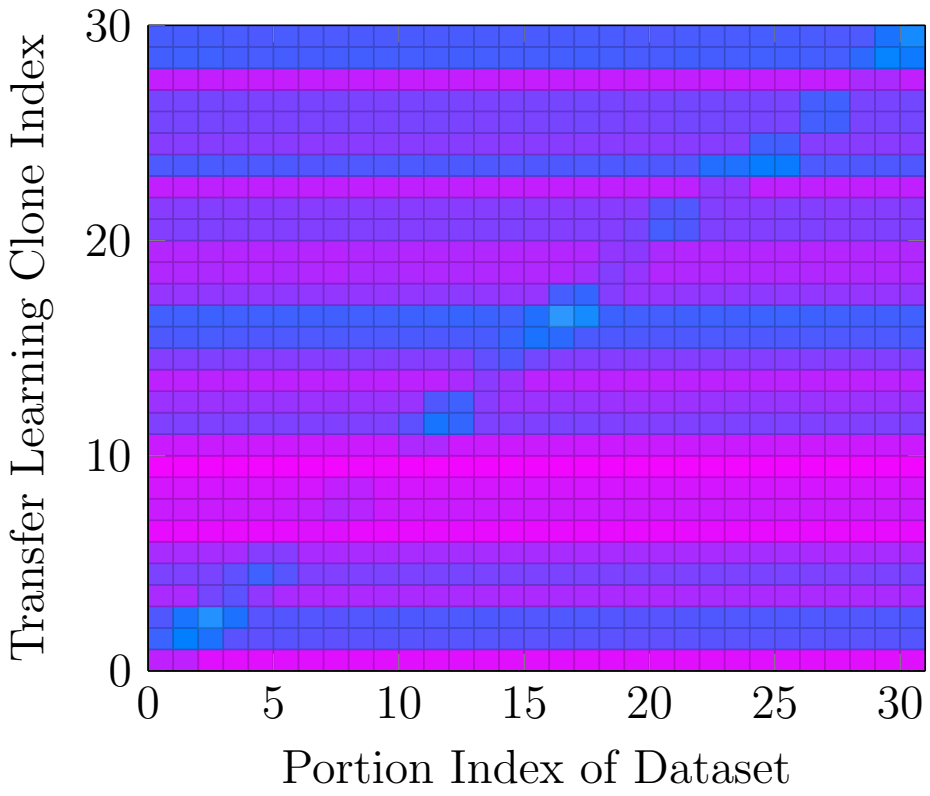


Figure 22. Training/Validation Matrix of the Clones

Finally, the relevant vectors from the 30 separate embeddings were averaged together to produce the final singular embedding. For that combined singular embedding, on the complete 14,820 training-word set (i.e., all 30 portions together), we obtained a mean training accuracy of **97.91%** with a standard deviation of 0.379092. The reason for this re-calculation is the fact that we anticipated that since we merged the trained models, the performance of merged model would not be the average of the separate components. The observable change in accuracy is proof that the said assumption is justified. Also, it is possible to note here that the above rigorous validation has cleared, in the case of the merged model, any possibility of over-fitting which may have threatened individual transfer learning clones. Further, note how the accuracy here is higher than the autoencoder accuracy. This is a result of the accuracy of oppositeness embedding being dependent on the distance between word embedding pairs regardless of where they map in vector space, while the autoencoder was attempting to recreate the exact vector given as input. Following that, on the 9910 test words, we obtained a mean test accuracy of **97.82%** with a standard deviation of 0.4316496. Yet again, we present the closeness of the training accuracy and test accuracy as proof that the system has not over-fitted to the data, despite obtaining very good training accuracy, which is higher than 97%. Note that all accuracy values at this point are calculated by taking the output of Equation 3.14 as the gold standard.

4.4 Conclusion

The main research contribution of this Chapter was the introduction of semantic oppositeness embedding. This Chapter successfully proposed and demonstrated an embedding methodology on 49,148,652 pairs of words, to obtain a training accuracy of 97.91% and a testing accuracy of 97.82%.

In addition to this main research contribution, this Chapter also introduced a novel, unanchored vector-embedding approach and a novel, *inductive transfer learning* process based on autoencoders, which utilizes both learnt embeddings and the learnt latent representation.

With experiments carried out with 30 instances of transfer learning, which yield 30 cases of training accuracy tests and 870 different cases of validation accuracy tests, it was observed that the linguistic property that some words may be more central in a lexicon than others, by having more synonyms or by having polysemy, is more crucial for the final overall accuracy on both training and validation sets, rather than an arbitrary stopping criterion for training.

Part B

Applications (Use Cases)

In this part of the dissertation, we apply the semantic oppositeness measurement and its embedding of Part A to two problems in the natural language processing domain. In particular, we introduce our new semantic oppositeness-based algorithms for the following three applications:

1. Inconsistency detection in PubMed abstracts in the MicroRNA domain. (Chapter V: Section 5.5)
2. Using semantic oppositeness in tandem with semantic similarity to propose new relationships for the OMIT ontology. (Chapter V: Section 5.6)
3. Disagreement detection in social media posts to discover rumours. (Chapter VI)

Each of these applications is unique and requires non-trivial alterations to how semantic oppositeness measurement is utilized.

CHAPTER V

INCONSISTENCY: DISCOVERING INCONSISTENCIES AND SIMILARITIES IN PUBMED ABSTRACTS

5.1 Introduction

Second only to cardiovascular diseases in the rates of mortality caused by noncommunicable diseases, cancers claim 8.2 million lives worldwide each year (World Health Organization [WHO], 2020). Thus, research that could contribute to preventing or curing cancer is imperative. As the growth of cancer involves abnormal cell division, it is important to look at the agents that get involved in that process. MicroRNA (miRNA) is a small, non-coding RNA molecule that plays a complementary role to mRNAs (messenger RNAs) in the gene regulation step of cell division (Exiqon, 2016). It is possible to observe the presence of miRNA in plants, animals, and some viruses. Mainly, they are involved in RNA silencing and post-transcriptional regulation of gene expression. This vital role played by miRNAs in gene expression is what makes them relevant and interesting for the pursuit of a cure for cancer.

The work of this chapter is adopted primarily from a collaborative conference paper (de Silva et al., 2017) that was published at the *8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* and a collaborative journal paper currently under review. The journal paper is an extension of the conference paper. Both papers were composed by myself, Dejing Dou, and Jingshan Huang. The contributions of the conference paper are covered in Section 3.2 and Section 3.3 of this chapter while the unique contributions of the journal paper are covered in Section 5.6 and Section 5.8.2. As lead author, I developed and implemented the contributed techniques, and I wrote the majority of

the text contained in this chapter. Dejing Dou provided valuable guidance towards the motivation and application of this work. Jingshan Huang helped verify the results obtained using the proposed Semantic Oppositeness model on the OMIT project (Huang et al., 2014).

In light of the potential importance of miRNA, an increasing quantity of research is being engaged upon its domain, albeit that not all studies are confirming studies. As such, some of the new studies about miRNA might either alter, or even completely disprove, some of the prior knowledge. Recognizing how this knowledge evolves over the course of time is important for various analytical tasks. It is also vital to note these changes, so that forthcoming studies would not mistakenly base their assumptions and their start conditions on conclusions in a prior body of work that has since been disproved. Logically, when there are changes in the foundations upon which any later research is based, and from which any later research draws its conclusions, that later research will also need to be re-evaluated, especially if the conclusions of the foundational work(s) were found to be no longer valid.

The first objective of this chapter is to find such alterations in knowledge in the miRNA domain. For this, we need to have a source from which we obtain details of research about miRNA. The best source for details about scientific research is through research papers, which are the common means used in all sciences to publish new findings. In a research paper, the abstract is a fair summarization of both the area of focus and the conclusions driven by the research described therein. Given that miRNA falls in the medical domain, an ideal source for searchable medical abstracts is PubMed (National Center for Biotechnology Information [NCBI], 2020), a free search engine, accessing primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics. It is maintained by the United States National

Library of Medicine (NLM) at the National Institutes of Health (NIH) as part of the Entrez system of information retrieval. As of 31st October 2020, PubMed has over 30 million records going back to 1966. Of those, 13.1 million of PubMed's records are listed with their abstracts.

In addition to keeping records of research papers, PubMed also provides free access to a Medical Subject Headings (MeSH) (U.S. National Library of Medicine [NLM], 2020) database. MeSH is a comprehensive, controlled vocabulary for the purpose of indexing journal articles and books in the life sciences. Thus, it facilitates searching for a particular subject within the medical domain. MeSH is created and updated by the United States National Library of Medicine (NLM).

Based on the above observations and resources, we propose an ontology-based information extraction model to discover inconsistencies in PubMed abstracts as discussed in Section 2.11. These inconsistencies are found when the knowledge extracted from one abstract disagrees with the knowledge extracted from another abstract. Thus, the inconsistencies our model discovers are an indication of the aforementioned shifts and improvements in the study of the focal subject, which for our present purposes will be miRNAs. While important, research about extracting information from the abstracts of biomedical papers is limited to a very narrow area of topics (de Silva, 2017b). An example is the seminal work by Kulick, Bies, Liberman, Mandel et al. (2004) that extracted information on drug development and cancer genomics. As introduced in Section 2.5.2, Ontology-Based Information Extraction (OBIE) is a subfield of information extraction, in which an ontology is used to guide the information extraction process (Wimalasuriya & Dou, 2010). Given that this study is focusing on the miRNA domain, we used the Ontology for MIcroRNA Targets (OMIT) (Huang et al., 2016b) as the guiding ontology for the

OBIE process as discussed in Section 2.5.1. Because OMIT lacks relationship data, such that traditional OBIE methods were not applicable, we used the Open Language Learning for Information Extraction (OLLIE) (Mausam et al., 2012) discussed in Section 2.6. OLLIE is unique in utilizing tree-like representations of the dependencies of the sentence, such that it is able to capture long-range relations. Once relationship information is extracted from the abstracts in the form of triples, we use our novel method to calculate the oppositeness between the said relationships on the basis of the semantic similarity measure of Wu and Palmer (1994) as discussed in Chapter III.

The key idea of our methodology is that the information in the PubMed abstracts in the miRNA domain is expressed in terms of (a) concepts, and (b) relationships that exist between those concepts. An inconsistency would arise if the relationship that was extracted between two given concepts in a certain abstract is opposite to the relationship that was extracted between the same two concepts in a different abstract. As mentioned above, we use OBIE methods, utilizing the OMIT ontology, to extract the said concepts from the abstracts. In order to discover the relationships between the extracted concepts, we use the OLLIE information extraction system.

The second objective of this chapter stems from the aforementioned lack of relationship data in OLLIE. We take the triples that were created in the previous step and send them through an optimized comparison scheme with dynamic thresholds. The resultant relationships are then output as suggestions, to be added to OMIT after domain expert approval.

Our main contributions are as follows:

– We introduce an ontology-based information extraction model to discover inconsistencies in PubMed abstracts.

– We propose a new methodology to incorporate open information extraction into the ontology-based information extraction process, in order to compensate for the lack of relationships in the domain ontology.
– We propose a semantic oppositeness measure, to be used to calculate the oppositeness between two relationships. We illustrate how this novel semantic oppositeness measure is superior, both to the antonym method and to the naïve similarity inverse method.
– We propose an ontology-based information extraction model to discover similarities in PubMed abstracts, which then leads to compiling possible updates to OMIT, in order to fix its problem of lack of relationships.

This chapter provides an in-depth explanation of the algorithms that we used, created, and adapted. All the code for the implementations of this chapter[1] and collected data[2] are available to download.

The rest of the chapter is organized as follows: We describe our methodology from Section 5.2 to Section 5.6. The configuration of the computer on which we did our experiments is introduced in Section 5.7. Then the Results and discussion follow that in Section 5.8. The work is concluded in Section 5.9.

5.2 Data Preparation

5.2.1 Obtaining PubMed Abstracts. The first step was to obtain a list of relevant PubMedIDs. This was done by querying the on-line PubMed site[3] with the header, "miRNA". The PubMedIDs were then processed to remove duplicates, and they were then separated into easily manageable files, with a maximum of 1000 IDs each.

[1] https://github.com/OMIT-PubMed-Project

[2] http://aimlab.cs.uoregon.edu/obie/OMIT/PubMedInconsistencies/01_Abstracts.tar.gz

[3] https://pubmed.ncbi.nlm.nih.gov/

These IDs are then used to extract the abstracts out of the PubMed system. One important thing to note here is the fact that, even though PubMed has an option to query its system with an ID to supposedly return the relevant abstract, we found it to be inefficient for this study. More often than not, the formatting of the free text was done in different ways, as shown in Fig. A.34a (Kuzmenko, Smirnova, Ivanov, Starodubova & Karpov, 2016) and Fig. A.34b (Yang, Yi, Wang & Wang, 2016) in Appendix A. Thus, it proved that extracting the pure abstract out of this output would require some unnecessary effort. Instead, it was decided to use the XML interface provided by PubMed and extract the abstracts locally. This step corresponds to the "preprocessor" component of OBIE (Wimalasuriya & Dou, 2010).

5.2.2 Creating OLLIE triples. The downloaded free text is then subjected to the open information extraction system introduced by Mausam et al. (2012), which was described in Section 2.6 by the name OLLIE. This process extracts triples, in the form of binary relations, from the free text and creates a set of possible triples as shown in Example 5.1 with a sentence taken from Hu et al. (2015). From this point onward, this chapter will refer to these triples as "OLLIE triples".

Example 5.1 Open Information Extraction Example

```
Nevertheless, we found that miR-31 was particularly up-
   regulated in HSCs but not in hepatocytes during
   fibrogenesis.
0.689: (miR-31; was particularly; up-regulated)
0.661: (miR-31; was particularly up-regulated in; HSCs)
```

The first line of the example shows the original sentence itself. Then each line has an extracted triple. The number leading the triple is the confidence that the OLLIE algorithm has of the triple being valid.

The remainder of the triple is of the format $(A; R; B)$ where A is the *subject* of the relation R, and B is the *object* of the relation R. Typically, in regular information extraction processes, which were explained in the leading paragraphs of Section 2.4, these relations (R) are fairly simple and would contain one to a few words. Similarly, the Subject (A) and Object (B) are set out to be clear-cut singular concepts. However, due to the openness of this methodology, which does not depend on any *subject context* specific rule but the grammar rules of the language itself, the output of this step does not have those properties. Typically, the relation name is just the text linking the subject and the object. Subject and object, themselves, are more often phrases, rather than the coherent concepts expected. This is an issue that we rectify in a later step.

5.2.3 Creating Stanford XML files. The same free text obtained in Section 5.2.1 is sent through a system to extract other linguistic information. In this case, we are using the methodology developed by Manning et al. (2014). The objective of this step is to extract the parse tree, get the lemmatized forms of each word, and get each sentence element separated. The parse tree of the same sentence (Hu et al., 2015) shown in Example 5.1 ("*Nevertheless, we found that miR-31 was particularly up-regulated in HSCs but not in hepatocytes during fibrogenesis.*") is given in Fig 23. The result of lemmatization and PoS tagging of three words of the same sentence is given in Example A.1 and Example A.2 respectively in Appendix A. From this point onward, this paper will refer to these outputs for each abstract as "Stanford XML".

5.2.4 Creating medical term dictionary. Before moving on to the next part of this study, some background data have to be generated, pertaining to the

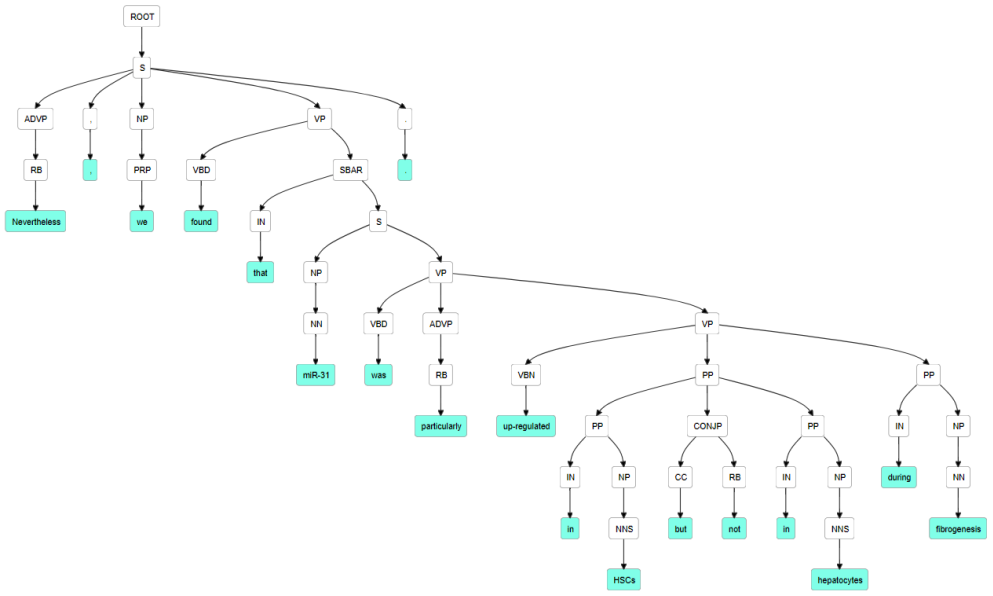


Figure 23. Parse tree of the sentence

abstracts. A very important part in an ontology-based information extraction system is the semantic lexicon (Wimalasuriya & Dou, 2010). WordNet is the primary lexicon in this system, as discussed in Section 2.2. But due to medical domain language being specific, a general lexicon such as WordNet is not enough to serve as the *Semantic Lexicon* for this system (Sugathadasa et al., 2017). Thus, a complementary lexicon must be created, with information specific to the medical domain. That is what is done in this step.

A good indication of how important a given term is in a certain domain is the frequency in which it is used within the domain. Therefore, the semantic information of term usage is vital to the following information extraction task, and it is not something that can be obtained via a generic lexicon, such as WordNet. Given that the semantic information that is to be extracted is of the format of term frequencies, it was decided to follow the structure of the famous information retrieval algorithm, TF-IDF (Leskovec et al., 2014) discussed in Section 2.7.

Each abstract is considered a separate document, and the term frequency of each term in the abstract is calculated. Then the inverse document frequency is calculated across abstracts. These two statistics are combined to calculate a semantic weight for each of the terms. Using the Stanford XML, the lemma of each term is extracted. Next, a triple consisting of the term (word), the lemma of the term, and its semantic weight is created for each term. Finally, the triples for each term (word) are output into a dictionary file as an intermediate output. Example 5.2 shows some typical lines from the dictionary file.

Example 5.2 Dictionary lines example

```
illuminators illuminator 0.045435406
twisting twist 0.0238714
lowering-drugs lowering-drug 0.049106136
mir-362 mir-362 0.03663714
mir-374 mir-374 0.07645514
mir-373 mir-373 0.1492043
mir-372 mir-372 0.13382968
ellas ellas 0.025369484
scavenges scavenge 0.013151284
architectures architecture 0.050796155
```

5.3 Creating Final Triples

With the above intermediary outputs ready, we move on to the next step of creating triples. Triples are created based on separate abstracts. Each of the OLLIE triple sets for a given abstract is read alongside the corresponding Stanford XML. Each triple carries the triple information (*Subject*;*Relationship*;*Object*), the confidence value, the relevant original sentence from the text abstract, and the sentence ID.

5.3.1 Triple building. The first information extraction step is a gazetteer list approach, as described by Wimalasuriya and Dou (2010). In this stage, a gazetteer list of MESH terms is derived from the OMIT ontology by extracting the concept tree

rooted at the *MESH term* concept, and adding all the individuals present in that tree to the gazetteer list. One important thing to note here is the fact that some of the strings in the OMIT ontology are not in the same format that one would use in a text. An example would be *Technology, Pharmaceutical*. Entries such as this were changed to the normalized form; for example, *Pharmaceutical Technology*. Next, the subject and the object of the triple are tested for occurrences of an individual now present in the gazetteer list. If any were present, the node list corresponding to the relevant subject or object is updated by appending the returned OMIT concept node to the end of the said list.

Next, Regular Expression (REGEX)-based information extraction is used. A base REGEX is built on the common usages of miRNA in abstracts and is matched to the counterparts in OMIT, as per the descriptions by Wimalasuriya and Dou (2010). The base REGEX is then expanded to cover all common forms of mentions of miRNA in literature. This is further enhanced by adding other pairings of REGEX and OMIT concepts. All of these REGEXes are then used to find the corresponding OMIT concept nodes for each of the words that exist in the subject or the object of the triple (depending on which one is being examined at the time.) These results, too, are then added to the node list as explained above.

The relationship in the OLLIE triple is then analyzed against the corresponding elements in the Stanford XML. In the case of the relationship being a single word, the lemmatized form of the said word is extracted from the Stanford XML, and the relationship is replaced with that lemmatized form. This simplification is not done when the relationship is a phrase.

The above steps are reduction steps, in the sense that out of all the concepts in the English language, only the ones that are directly relevant to the miRNA domain

are present in the OMIT ontology. Thus, the subject and/or object of some of the OLLIE triples will have empty node lists.

Next, a triple of each is created, using every node in the object list, for every node in subject list, utilizing the reduced or pure relationship from the original OLLIE triple. (As mentioned above, the relationship is only reduced when it is comprised of a single word.) This is an increment step, given the fact that the resulting number of triples is the multiplication of the number of elements in the subject list and the object list of the original OLLIE triple. Thus, this also means that any OLLIE triple that was reduced to have an empty subject list or an empty object list will produce no triples in this step.

5.3.2 Triple simplification. Newly created triples are then sent through two simplification processes. An important point to note is the fact that these simplifications happen on a sentence-by-sentence basis here. In this step, triples corresponding to one sentence have no effect on the triples corresponding to a different sentence.

The first simplification step goes through all the given triples and analyses the subject, the object, and the relationship. In the case where all three of them are equal for two given triples, a new merged triple is created with the same subject, object, and relationship, along with the average value for the confidence.

The second simplification uses the concept hierarchical information from OMIT. Thus, it belongs to the ideas of Ontology-Based Information Extraction discussed in 2.5.2. Here, the triple list is simplified, on the fact that some triples in the list are ancestors of other triples in the list, as defined in Definition 5.3.1.

Definition 5.3.1 (Triple Ancestor)**.** *A triple X is defined as the ancestor of another triple Y if and only if the following two conditions are satisfied: both triples have*

the same relationship; and the subject node and the object node of X are respectively ancestors of the subject node and object node of Y, as defined by Definition 5.3.2.

Definition 5.3.2 (Node Ancestor)**.** *The ancestor relationship for nodes W and Z are defined as follows: A node W is the ancestor of a node Z if and only if, the node W is the same as node Z, or the OMIT node of W is an ancestor of OMIT node of Z in the concept hierarchy of the OMIT ontology.*

First, the triple list is scanned from left to right, to see if any triple would be the ancestor of one that is listed left of it. In the case where an ancestor is found, the ancestor is discarded, and the descendant's confidence is set to the average of that of the original confidence value of the descendant and the confidence value of the ancestor. Then, the triple list is scanned from right to left, to see if any triple would be the ancestor of one that is listed right of it. The same simplification process used in the left-to-right scan is applied on the ancestors and descendants that are found.

The rationale of this process is the following: In the step in which we created the new triples out of OLLIE triples, we were doing string REGEX matching on the subjects and objects of the OLLIE triples and assigning nodes that correspond to a concept in OMIT. There are many cases in OMIT ontology where the name of an ancestor node is a substring of a descendant node. An example is shown in Fig. 24, where the concept node with the name *"Cells"* has descendants with names such as *"Goblet Cells"* and *"Dendritic Cells"*. Thus a sentence that mentioned *"Goblet Cells,"* such as *"The goblet cells are found in the intestinal tract,"* which is expected to produce the triple *(Goblet Cells ; are found in ; Intestinal Tract)*, will also produce the triple *(Cells ; are found in; Intestinal Tract).* From definition 5.3.1, it is evident that the latter triple is an ancestor of the former triple. Thus, by the simplification process discussed above, the latter triple is removed, and the confidence of the former

triple is updated, using the current confidence values of the former and latter triples. This makes sense, because such sentences are always relevant to the concept with the smaller granularity, as shown in the above example.

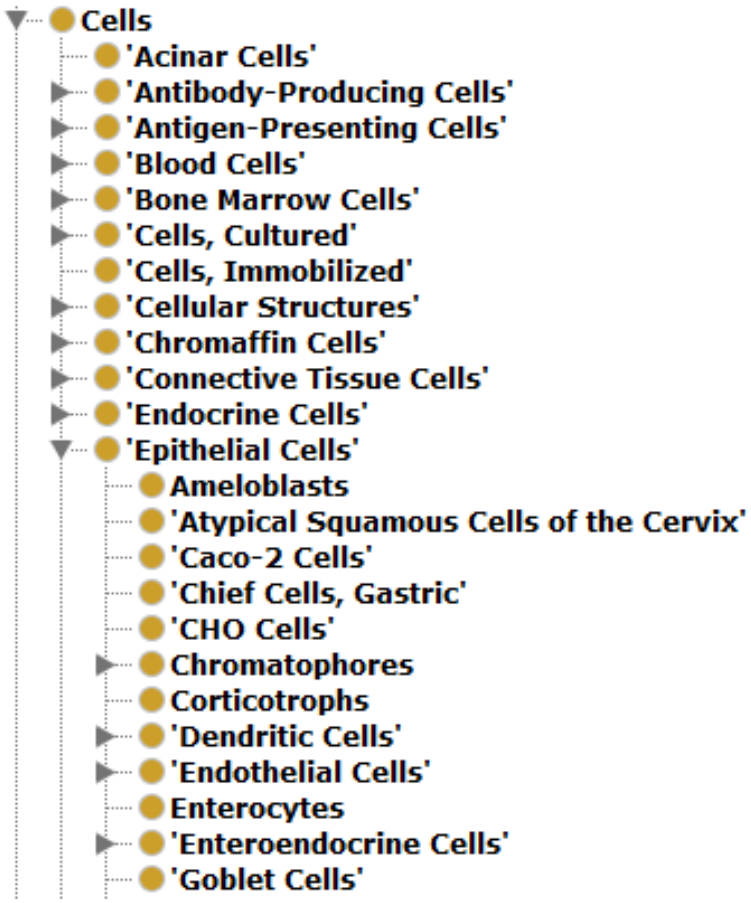


Figure 24. Part of OMIT hierarchy

Once the simplification process is finished for each sentence, all the resultant triples are added to a single list. Then that list is passed to a simplification process similar to that of the first step, but with a slight change. Just like in the per-sentence simplification, the process goes through all the given triples and analyses the subject, the object, and the relationship; but this time, it is done over the entire abstract. It should be noted that the second simplification, i.e., ancestor-based simplification, is not done here. This is because of the possibility of losing a generalized claim when

it exists in an abstract that also makes a specific claim. In the case where all three – subject, object and relationship – are equal for given triples, a new, merged triple is created. But this time, the new triple will carry both sentences (if they are different), and the confidence value is updated to the new value C_{new} according to Equation 5.1, where: the confidence in triple i is given by C_i (Such that $0 < C_i < 1$); the sentence count in triple i is given by S_i. Sentence count is never zero.

$$C_{new} = \frac{\sum_i C_i * S_i}{\sum_i S_i} \tag{5.1}$$

The resultant triples of the above process are put into a list. These are the final triples. The final triples are then written to a set of files as an intermediate output. A separate file is written for each separate abstract. By this point, some abstracts will have empty lists, because none of the OLLIE triples of those abstracts have survived the conversion to the final triples form, if the OLLIE triples from those abstracts lacked any information relevant to be extracted using the OMIT ontology. These abstracts will have empty files in their name.

5.4 Facilitating the Inconsistency Detection

From here onward, we discuss the methodology used to find inconsistencies using the final triples, other resources, and intermediate files created in the previous sections.

5.4.1 Preparing to Compare Relationship Strings. The first order of business for finding inconsistencies is to load the intermediate files created at 5.3 and 5.2.4, for new triples and the dictionary, respectively. Abstracts are read, and data are loaded, next. But instead of storing data with the distinct unit *per abstract*, as we have been doing so far, a new minimum unit is introduced, which has a unique

entry *for each triple*. Which means a sentence with multiple candidate triples will be represented in corresponding multiple entries.

All the triple entries are loaded to a list. Each triple entry i is compared with each triple entry j, such that i goes from 1 to the length of triple entry list, while for each i, j goes from $i+1$ to the length of triple entry list. This way, the triple entries are compared with the triple entries that follow them; thus, each pair of triple entries only gets compared once.

5.4.2 Initial filtering. Before the analysis begins, a couple of filters are applied. The first filter makes sure that triple entries of the same abstract are not compared to each other, because finding inconsistencies within the same abstract is not the objective of this work. The second filter is applied to handle the cases in which a redacted article is found to have the exact same content as another legitimate article. In this case, one is dropped from the consistency checking. For the purpose of this study, it does not matter which one is dropped, for the simple reason that if the legitimate article is dropped, and the system ends up finding an inconsistency with the redacted article against some third article, it is a simple matter of re-consulting the PubMed database to find the relevant legitimate article by way of the redacted article.

5.4.3 Cleaning the strings. The relation value of triple entry pairs that pass the filtering process is then put through a cleaning process. Special contractions such as "can't", "won't" are explicitly handled, and simple contractions such as "don't", "hadn't" are scripturally handled. Next, the relationship is split to the terms, and when there exists a "not", it is handled as the negation of the following term. Following that, all the stop words are removed from the list; and finally, using

the lemmatization results loaded from the dictionary created at section 5.2.4, all words are stemmed to their basic lemma.

5.4.4 Calculating oppositeness of relationships. The two lists of cleaned strings that were created from the triple relationships are then evaluated against each other, word-by-word. We define the item count of these lists as c_1 and c_2. Before going into the oppositeness function, some simple comparisons are made to lighten the computing load.

When both the comparing words are exactly the same, the weight of the word is extracted from the dictionary that was created at section 5.2.4 and loaded at the beginning of section 5.4.1. This is raised to the power of two, then multiplied by the constant "yes weight" (W_{yes}). The resultant value is added to the similarity amount ($simil_T$); the similarity number counter (s_n) is increased by one.

When either of the words is the direct simple negation of the other by the keyword "not", (i.e.: "increased"-"not increased", "found"-"not found"), again the weight of the non-negated word is extracted from the dictionary and raised to the power of two. The resultant value is then multiplied by the constant "no weight" (W_{no}). This value is added to the difference amount (dif_T); the difference number counter (d_n) is increased by one.

For every other pair of words, the lemma is extracted using the dictionary created at section 5.2.4. Let us call them L_1 and L_2. When the word does not exist in the dictionary, the word itself is used as its own lemma. Next, the *oppo* value for each pair is calculated using the methodology described in Section 3.2 to Section 3.3 in Chapter III by means of Equation 3.6. Note that for the use-case in this chapter we used the *oppo* measure of Equation 3.6 rather than that of Equation 3.14 given that the

research for this chapter was conducted in de Silva et al. (2017) prior to the changes introduced in de Silva and Dou (2019) which included the updated Equation 3.14.

The final *oppo* value, after the threshold is applied, is multiplied by -1 and is returned as the oppositeness measure of the two words. The returned value is then multiplied by the weights of the two words extracted from the dictionary. If the value is greater than zero, the value is multiplied by the constant "yes weight" (W_{yes}). The resultant value is added to the similarity amount ($simil_T$); the similarity number counter (s_n) is increased by one.

If it is less than zero, the value is then multiplied by the constant "no weight" (W_{no}) and -1. This value is added to the difference amount (dif_T); the difference number counter (d_n) is increased by one. Thus, when the value is zero, no change happens to any similarity/difference values or counters.

5.4.5 Finalizing the oppositeness of relationship strings. Once all the words in the two relationship strings have finished going through the above steps, both $simil_T$ and dif_T are normalized using a small constant ϵ with s_n and d_n as shown in equations 5.2 and 5.3.

$$simil_T = \frac{simil_T * (d_n + \epsilon) * W_{yes}}{s_n + d_n + 2 * \epsilon} \tag{5.2}$$

$$dif_T = \frac{dif_T * (s_n + \epsilon) * W_{no}}{s_n + d_n + 2 * \epsilon} \tag{5.3}$$

Finally, if $simil_T$ is greater than dif_T, $simil_T$ is returned as the similarity value of the two relationship strings. Otherwise dif_T multiplied by -1 is returned as the difference value of the two relationship strings.

5.5 Discovering inconsistencies

5.5.1 Registering inconsistencies.

The value returned by the above step for a given pair of relationship strings is then multiplied by -1 and put through a threshold test. If it passes the threshold, it is registered as an inconsistency.

For each abstract that gets involved in a potential inconsistency, PubMed is queried again, to obtain the publication date and other relevant details. The reason for doing this at this stage is the fact that only a small portion of all abstracts are relevant for this stage; and thus, we can do a lesser amount of processing and data storage for the bearable cost trade-off of a few instances of XML fetching over the Internet.

Each of the inconsistencies that are found are written to an intermediate result file, where a line holds; confidence (the difference value returned); PubMedIds of the contradicting abstracts, along with the publication dates; subject and object of the relevant triple; the relationship present in the triple in the first abstract; the relevant sentence ID from the first abstract; the relationship present in the triple in the second abstract; and the relevant sentence ID from the second abstract. An example of some lines from the said intermediate result file is shown at Example 5.3.

Example 5.3 Intermediate inconsistency result example

```
0.8333333;24969691;2014/9/1;27601936;2016/9/7;Cells;Vimentin;increase;3;
    decrease;7
0.8333333;25435961;2015/1/1;26632856;2015/12/1;DNA;Cells;promote;7;
    breaks in;12
0.625;25004396;2014/6/15;26257392;2015/11/1;MIR152;Cells;were decreased
    in;3;be Interestingly increased in;10
```

5.5.2 Preparing inconsistency for analysis.

This is the final stage of the methodology for finding inconsistencies. First, the intermediate result file written in the previous step is read. Then the Subject and Object of the inconsistent triples are checked against OMIT, to see if either or both of them are of the type miRNA. The reason we pushed this check to this final step is that, this way, the intermediate file created before this step can potentially be used for other research on inconsistencies in the medical abstracts, in domains other than miRNA, as well.

If either or both the subject and the object are, indeed, of the type of miRNA, then for each such inconsistency, the relevant OLLIE files are read, and the contributing actual sentences are extracted using the sentence IDs. Then, the information gained from the intermediate result file and extracted sentences is reformatted to be more readable by humans. Here, finally, the original OLLIE confidences are used. The final confidence Con_{fin} is calculated using the inconsistency confidence Con_{cont} calculated above, OLLIE confidence of triple 1 Con_1, OLLIE confidence of triple 2 Con_2, and the constant C as shown in Equation 5.4. C is selected $C > 1$.

$$Con_{fin} = C * Con_{cont} * Con_1 * Con_2 \tag{5.4}$$

The reformatted inconsistencies are then written to the final result file, to be read and analyzed by human experts. An example of some lines from the final result file is shown in Example 5.4. The lines in the example are from Wang, Lv, Liu, Zhu and Qiu (2015) and Liu et al. (2016). All the results up to this point can be downloaded from the project website[4].

[4]http://aimlab.cs.uoregon.edu/obie/OMIT/

Example 5.4 Final result file example

```
.................................................

0.056045435

25738546
2015/5/1
( MIR214 ; was significantly increased in ; Tissues )
4
Our results revealed that miR-214 expression was significantly increased
    in the BC tissues compared with the adjacent benign tissues, and
   that the upregulation of miR-214 was significantly associated with
   the invasion ability of the BC cells.

27109339
2016/6/1
( MIR214 ; were significantly decreased in ; Tissues )
4
Our results revealed that the expression of miR-214 and miR-218 were
   significantly decreased in breast cancer tissues compared with
   adjacent tissues.

.................................................
```

5.6 Discovering Potential Relationships for OMIT

The second application of the relationship data that were generated in Section 5.4 is suggesting relationships for OMIT. As mentioned in the Section 2.5.2, the lack of relationships is the greatest weakness of OMIT. Thus, it was suggested that this research attempt to remedy this problem. Before getting on to the relationship-finding algorithm from Section 5.6.2 onward, it is important to discuss the optimized relationship comparison scheme that was used in this task. The rationale and intricacies of the said scheme are described in Section 5.6.1.

5.6.1 Optimized Relationship Comparing Scheme. Given that the objective of this step is to find the best relationship to suggest for the given pair of entities, the relationship-comparing scheme of this step is fundamentally different from that of Section 5.5.1. In Section 5.5.1, the inconsistencies could be obtained by

comparing each relationship once. The reason for this is the fact that the inconsistency threshold was global. Thus, if the relationship R_1 was deemed inconsistent against relationship R_2, there was no need to check and see if relationship R_2 is inconsistent against relationship R_1 or not. The inconsistency property was symmetric around the common global threshold. But here, the relationships are measured over a relative threshold, solely dictated by the semantic properties of R_1. This asymmetrical behaviour of oppositeness is further discussed by Mikołajczak-Matyja (2018) with Polish use cases (Bednarek & Grochowski, 1993; Grochowski, 1982). In the case of computational complexity, this does not alter anything, given that both the processes are of $O(n^2)$ complexity. However, in actual execution, the comparison step described here takes twice the time that it takes for the comparisons in Section 5.5.1.

Consider the case of six relationships R_1, R_2, R_3, R_4, R_5, and R_6. The instances where each relationship has to calculate the semantic similarity-oppositeness using the algorithm described in Section 5.4 are given in matrices in Fig 25. If a relationship named in a row label has to be calculated using the semantic similarity-oppositeness against a relationship named in a column label, that particular cell is shaded and carries the number 1. Otherwise, the particular cell is not shaded, and it carries the number 0. The Fig 25a shows the instances where similarity-oppositeness calculation had to be done in the case of Section 5.5.1. The Fig 25b shows the instances where similarity-oppositeness calculation has to be done in the case of this section. As mentioned above, it is obvious that the computational workload has now doubled from what it was before *(15 instances versus 30 instances)*.

To solve this issue, we introduced a testing step of lexical similarity before the semantic similarity test is carried out. The idea is that before two relationships R_1 and R_2 are compared using the semantic similarity measures described in the

	R_1	R_2	R_3	R_4	R_5	R_6
R_1	0	1	1	1	1	1
R_2	0	0	1	1	1	1
R_3	0	0	0	1	1	1
R_4	0	0	0	0	1	1
R_5	0	0	0	0	0	1
R_6	0	0	0	0	0	0

(a) Matrix A

	R_1	R_2	R_3	R_4	R_5	R_6
R_1	0	1	1	1	1	1
R_2	1	0	1	1	1	1
R_3	1	1	0	1	1	1
R_4	1	1	1	0	1	1
R_5	1	1	1	1	0	1
R_6	1	1	1	1	1	0

(b) Matrix B

Figure 25. Matrix A and B

algorithm described in Section 5.4, we do a simple lexical similarity test between relationships R_1 and R_2. If the lexical similarity test is deemed successful, we forgo the calculation of semantic similarity. Otherwise, the semantic similarity calculation happens as described above.

Consider the case of the same six relationships we mentioned above: R_1, R_2, R_3, R_4, R_5, and R_6 where $(R_1, R_3)_{lexical}$, $(R_1, R_6)_{lexical}$, and $(R_2, R_5)_{lexical}$, all equal to 1 $(TRUE)$[5]. Now, with dropping of lexically similar pairs from having to be compared for semantic similarity, we can obtain what is shown in Fig 26a. This reduces the number of needed semantic comparisons to *22 instances*. Even though this is a step up from the *30 instances* in Fig 25b, it is still behind the situation in Fig 25a which only had *15 instances*.

The introduction of the lexical similarity section allowed us to further reduce the computation cost by having the ability to prevent any relationship, which was in a successful lexical similarity test in the role of the sink against any other relationship, from being considered as the source for future comparisons. The reason for this is the fact that all the acts of comparison on them will be mirroring what happened with the relevant previous source relationship. The Fig 26b shows this situation. Now the

[5]By transitive property of the lexical similarity, we can trivially claim that $(R_3, R_6)_{lexical} = 1$ as well

number of comparisons needed for our example case has been reduced to *12 instances*. Note here that this result is not only better than the result shown in Fig 26a, but also better than the situation shown in Fig 25a. Further, it can be claimed that when the relationship count is n, and the unique relationship count is m, where uniqueness is defined as the property of being dissimilar under the lexical similarity property. The number of needed semantic comparisons have changed from $n(n-1)$ to $n(m-1)$. Given that $m \leq n$, we can claim that in practical applications, the algorithm runs faster now.

	R_1	R_2	R_3	R_4	R_5	R_6
R_1	0	1	0	1	1	0
R_2	1	0	1	1	0	1
R_3	0	1	0	1	1	0
R_4	1	1	1	0	1	1
R_5	1	0	1	1	0	1
R_6	0	1	0	1	1	0

(a) Matrix C

	R_1	R_2	R_3	R_4	R_5	R_6
R_1	0	1	0	1	1	0
R_2	1	0	1	1	0	1
R_3	0	0	0	0	0	0
R_4	1	1	1	0	1	1
R_5	0	0	0	0	0	0
R_6	0	0	0	0	0	0

(b) Matrix D

	R_1	R_2	R_3	R_4	R_5	R_6
R_1	0	1	0	1	0	0
R_2	1	0	0	1	0	0
R_3	0	0	0	0	0	0
R_4	1	1	0	0	0	0
R_5	0	0	0	0	0	0
R_6	0	0	0	0	0	0

(c) Matrix E

Figure 26. Matrix C, D, and E

The above measures paved the way to further reduce computation by eliminating comparisons such as $(R_x, R_z)_{semantic}$ when we already know the value of $(R_y, R_z)_{semantic}$ and the fact that $(R_x, R_y)_{lexical} = 1$. Instead of doing a recalculation, we use a hash map to store the already calculated data, allowing us to apply dynamic computing principals to the system. Thus, we reduce the number of costly semantic comparisons drastically. The Fig 26c shows this arrangement for our running example.

The need to do semantic similarity calculations has been reduced to a staggering *6 instances*. This further enhancement alters the needed semantic comparisons to be $m(m-1)$. Again, by the same previous fact $m \leq n$, we can claim that in practical applications, the algorithm runs even faster than it did when the needed semantic comparisons were at $n(m-1)$.

5.6.2 Calculating Relationship Value. The basics on registering relationships is the same as what we used to find inconsistencies above. For each pair of entities, we have an initial set of relationships (P). These relationships need to be compared within each other. As mentioned in Section 5.6.1, if the lexical similarity test is deemed successful between two relationships, we increase a lexical similarity counter ($N_{l,i}$) and forgo the calculation of semantic similarity. Otherwise, the semantic similarity calculation happens.

The semantic similarity calculation is yielded by the steps in Section 5.4. This, as described above, is a dual test that analyses both semantic similarity and semantic oppositeness. Here the relationship pair is selected as an initial candidate only if the result of the above proves to be partial to similarity rather than oppositeness. The membership condition for the sink relationship j in respect to the initial candidate set C_i for a source relationship i is given by Equation 5.5.

$$C_i = \{j | simil_T(i,j) > dif_T(i,j)\} \tag{5.5}$$

The most important point to note here is the fact that this works as a dynamic threshold to filter candidate membership. This proves to be a much more effective methodology than using a simple direct threshold on similarity, for the simple reason that this method allows for a relative threshold unique to each pair of relationships, rather than a common global threshold. Each time a semantic similarity calculation

happens that yields a non-zero result after applying the threshold, we increase a semantic similarity counter ($N_{s,i}$) and add semantic similarity value to a cumulative sum (Sem_i).

The reason to maintain separate counters for the lexical similarity and semantic similarity is to make sure the two values can be separately normalized. Each time a lexical similarity counter or the semantic similarity counter is updated, the corresponding abstract ID and sentence number are added to lists. The elements that were added along with lexical similarity are later used to build the *Matches* list, and the elements that were added along with semantic similarity are later used to build the *Close Matches* list. That process is described in Section 5.6.3. Each relationship pairing that passes either through lexical similarity or semantic similarity contributes to the confidence value $SimCon_{R_i}$ following the Equation 5.6 where Con_{R_i} is the confidence of R_i which is the source relationship, Con_{R_j} is the confidence of R_j which is the sink relationship, and C is a scaling constant.

$$SimCon_{R_i} = \sum_{j \in C_i} C * Con_{R_i} * Con_{R_j} \tag{5.6}$$

The final confidence value $finCon_i$ of relationship R_i is calculated using Equation 5.7, where $N_{l,i}$ is the lexical similarity counter value, $N_{s,i}$ is the semantic similarity counter value, and $SimCon_{R_i}$ is the relevant value calculated in Equation 5.6.

$$finCon_i = \frac{SimCon_{R_i}}{N_{l,i} + N_{s,i}} \tag{5.7}$$

Finally, the overall value V_i of the relationship R_i is calculated using Equation 5.8, where $finCon_i$ is the confidence calculated in Equation 5.7, $N_{l,i}$ is the lexical

similarity counter value, $N_{s,i}$ is the semantic similarity counter value, and Sem_i is the cumulative semantic similarity value calculated above.

$$V_i = \frac{finCon_i * (N_{l,i} + Sem_i)}{N_{l,i} + N_{s,i}} \tag{5.8}$$

At this point it is important to understand what exactly the value V_i of a relationship R_i indicates. It is an evaluation of R_i's placement within the set P. Assume we were to place all relationship elements in P on an arbitrary vector space such that elements that are:

1. Similar to each other with high confidence are placed very close to each other.
2. Similar to each other with low confidence are placed moderately close to each other.
3. Dissimilar to each other with low confidence are placed moderately far from each other.
4. Dissimilar to each other with high confidence are placed very far from each other.

In such a vector space, the elements with high V_i values will be placed closer to the center of the cluster. Thus, a high V_i value is an indication of the fact that the said relationship R_i is representative of the considered relationship set P.

5.6.3 Registering Relationships. As mentioned in the Section 5.6.2, the calculated V_i values indicate how representative of the set P the relationship R_i is. Thus, the next step was to sort all relationships according to their V_i values. At this point it was decided to retain at least the top two suggested relationships, when data allowed such a selection. The reason for this decision was the fact that the final objective of this part of the research was to suggest relationships to be

added to the OMIT. Thus, the results are only added at the discretion of the domain experts. It is not a black-box process of automatically adding the relationship with the highest score. This human component implied that it is better to give the human experts some freedom of choice, by providing them with multiple choices for relationships when possible. Therefore, in the cases where there were tied V_i values at the top, relationships were permitted until we arrive at a situation where we have at most two unique V_i values. This further implies that when there were ties, the total number of accepted relationships is bounded above only by the cardinality of set P. The extracted data is set in the format shown in Example A.3. Each $< Relationship\ Data\ i >$ item is further represented as shown in Example A.4. Both $< Matches\ List >$ and $< Close\ Matches\ List >$ follow the list data format shown in Example A.5 where $< ID\ j >$ refers to the abstract id of the jth abstract and $< Sentence\ Number\ j >$ refers to the sentence index of the relevant sentence in the context of abstract j. An example of a data line that got by executing this step is shown in Example A.6. An intermediate output of such data lines was created. The intermediate results for this section are available to download[6].

5.6.4 Preparing relationships for analysis. This is the final stage of the methodology for suggesting relationships for OMIT. First, the intermediate result file written in the previous step is read. Then the Subject and Object of the inconsistent triples are checked against OMIT, to see if either or both of them are of the type miRNA, as in Section 5.5.2. Yet again, the reason we pushed this check to this final step is to preserve the potential of the created intermediate file being used for other research activities that may or may not align with the task of suggesting relationships for OMIT.

[6]http://aimlab.cs.uoregon.edu/obie/OMIT/PubMedInconsistencies/results_Sim.txt

Yet again, similar to the process in Section 5.5.2, if either or both the subject and the object are, indeed, of the type of miRNA, then for each such suggested relationship, the relevant OLLIE files are read, and the contributing actual sentences for the *Matches* list and the *Close Matches* list are extracted using the abstract IDs and sentence IDs. The subject and the object in the original line and the suggested relationship are put together to re-create the potential triple. The sentences in the *Matches* list and the *Close Matches* list are added, to help the expert make an informed decision. Section breaks are introduced to increase the readability of the output for the benefit of the experts. The reformatted suggested relationships are then written to the final result file, to be read and analyzed by human experts. An example of some lines from the final result file is shown in Example A.7 (Sentences are from Sun et al. (2012) and Chen et al. (2015)). Note how the sentence under *matches* of one relationship is in the *close matches* of the other relationship and visa-versa. This is because the top relationships that are extracted for the subject-object pair we have selected for the example in this paper are highly co-related. Further, it is possible to make a case that the second suggested triple *(MIR320A;inhibit;Cell Proliferation)* seems more suitable than the first suggested triple *(MIR320A;Moreover inhibits;Cell Proliferation)*, in the aspects of brevity and accuracy. We showcase this as supporting proof for our earlier decision in Section 5.6.3 to provide the experts with the top few options instead of just returning the top scoring relationship. The final results for this section are available to download[7].

[7]http://aimlab.cs.uoregon.edu/obie/OMIT/PubMedInconsistencies/simReduResults.txt

5.7 Configuration

All steps of the methodology were implemented in Java (jdk1.8) and were run on a computer with Windows 10 Home 64-bit, Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz, 16GB (15.9GB Usable) RAM.

5.8 Results and Discussion

In the PMID-extraction step, we obtained 39,149 relevant abstract IDs, from which 36,877 were processed and downloaded as text files containing abstracts. Around 5.8% of extracted PubMed entries did not have an abstract section, and there were three possible situations:

1. When an entry had some graphs instead of an entire research paper, e.g., PMIDs 24324220, 24318653, 24311611, and 24303553.
2. When there was only a comment about the entry rather than a complete entry, e.g., PMIDs 24311611 and 24303553.
3. When the entry was empty except for the entry name, author names, and other metadata, e.g., PMID 24313780.

Other than these three situations, each and every abstract from the remaining 94.2% of relevant IDs were downloaded for analysis. All 36,877 downloaded abstracts were processed to yield OLLIE triple files and Stanford XML files. These intermediate files were used to create the intermediate result file, where a total of 67,481 unique subject-object pairs were detected.

5.8.1 Inconsistency Detection Results. 503 total inconsistencies were discovered from these subject-object pairs, involving 224 out of 36,877 abstracts. This observation indicated that the percentage of abstracts that contributed to inconsistencies was only 0.61% out of all considered.

After the reduction step (detailed in Section 5.5.2) was performed to keep only the inconsistencies that involved at least one miRNA entry, we ended up with 102 inconsistencies involving 95 abstracts. This outcome revealed that, out of 503 total inconsistencies, only 20.28% were relevant to miRNA. Abstracts participating in inconsistencies involving miRNA consisted of 0.26% of all downloaded abstracts, and 42.41% of those abstracts that were found to be involved in inconsistencies of any kind.

The very miniature nature of these numbers is the best justification of this research. When contradictions are at low percentages, such as 0.26%, asking experts to find them by manually reading 36,877 downloaded abstracts is an extremely unreasonable demand. But by using our methodology, we not only pinpoint the 95 abstracts that are relevant; we also extract the exact sentences that contribute to the inconsistencies.

5.8.2 OMIT relationship extraction Results. 4443 subject-object pairs yielded potential relationships in Section 5.6.3. Thus, in comparison with the total 67,481 unique subject-object pairs, it can be claimed that 6.58% of the subject-object pairs have displayed strong enough potential to be considered to be added as relationships. After the miRNA filtering done in Section 5.6.4, 1636 subject-object pair base relationships were deemed to be relevant to the miRNA domain. This is, in fact, 36.82% of the total subject-object pairs with relationship potential found above. In accordance to the same, this is related to 2.42% of the total subject-object pairs.

After the relationship preparation of Section 5.6.4, 3065 relationship suggestion blocks, such as the one shown in Example A.7 were created. Thus, in comparison to the 1636 subject-object pair base relationships that were found, this implies an average of 1.87 blocks per subject-object pair. That this result is less than 2 proves

that our methodology, by which we handle value ties, in Section 5.6.3, by listing them, has not adversely affected the result set by listing exponentially large counts of suggestions. Further, the fact that the number 1.87 is close enough to 2 proves that, in most cases, the objective of providing the experts with two suggestions, whenever there is sufficient data, has been successful.

5.9 Conclusion

One of the primary research contributions of this chapter was to use ontology-based information extraction to observe how inconsistencies rise in the literature in relation to previously established knowledge in a scientific field. This study successfully proposed a method by which to conduct that observation, and we succeeded in finding 503 such inconsistencies in a corpus of 39,149 research paper abstracts. Since these inconsistencies are rooted in very domain-specific medical jargon, they need to be analyzed by medical experts before getting incorporated into future studies.

This study had to face the problem that the ontology that was being used did not have the relationship rules that most of the established OBIE systems use. Thus, this study came up with a novel way to solve this problem, by involving open information extraction systems to extract the relationships and then using the conventional OBIE systems to do the information extraction. This methodology can be considered as a new way of doing OBIE, in addition to the traditional and established methods discussed in Wimalasuriya and Dou (2010).

Another primary contribution of this study was the use of the above mixture of ontology-based information extraction and open information extraction, along with a semantic similarity and oppositeness measuring algorithm, to suggest relationships for an ontology that lacks the relationships. In this particular case, it was focused

on OMIT, itself, given that we discussed how OMIT lacks this key component. We found 4443 relationship groups, and we were able to suggest 3065 relationship options for the further development of the OMIT ontology.

For future work, one most basic thing that can be improved is in the preprocessing stage, to include common medical acronyms that are used but are not defined in the first use. It is also possible to investigate the redacted articles mentioned in Section 5.4.2 to see if the redaction was a result of an inconsistency. It is also possible to extend the *cleaning the strings* step (Section 5.4.3) and the *creation of final triples* step (Section 5.3) using the already generated Stanford XML.

Funding for this research was provided by the National Cancer Institute (NCI) at the National Institutes of Health (NIH), under the Award Number U01CA180982.

CHAPTER VI

DISAGREEMENT: RUMOUR DETECTION IN SOCIAL NETWORKS

6.1 Introduction

Social media changed the ecosystem of the World Wide Web by making it possible for any individual, regardless of their level of knowledge of web technologies, to create and maintain *profiles* online. At the same time, various social media provided these individuals with means to tap into the information disseminated by others (e.g., Facebook by *adding friends*, Twitter by *following*). By virtue of other mechanisms, such as *Facebook pages* and *Twitter lists*, the reach of each individual was then extended to the range of thousands-to-millions of users. New content, in the form of *posts*, is created on social media sites each passing second. The rapidity of this post creation is such, that it is possible to claim that social media reflect a near real-time view of the events in the real world (Veyseh et al., 2019). While it was, indeed, beneficial in terms of *volume* of data, to have private individuals be content creators and propagators of information, this created significant issues, from the perspective of the *veracity* of the data. This gave rise to a challenge of detecting *fake news* and *rumours* (which, in this chapter, we refer to as the task of *rumour detection* as discussed in Section 2.12). The need for rumour detection has come to the forefront, in light of its momentous impacts on political events (Jin et al., 2017) and social (Jin, Cao, Jiang & Zhang, 2014) or economic (Domm, 2013) trends. Manual intervention on this task would require extensive analysis of and reasoning about various sources of information, resulting in long response times, which are intolerable, given the impact of these rumours, and the rate at which they spread. Thus, *automatic rumour detection*, toward which we contribute in this chapter, has become an important area of contemporary research.

The work of this chapter is adopted primarily from a collaborative paper that is under review and was composed by myself and Dejing Dou. As lead author, I developed and implemented the contributed techniques, and I wrote the majority of the text contained in this chapter. Dejing Dou provided valuable guidance towards the motivation and application of this work. This chapter discusses how the Semantic Oppositeness measure can be used in the use-case of *automatic rumour detection in social networks* where the Semantic Oppositeness is used to *detect disagreements* in reply threads, rooted at a potential rumour or non-rumour tweet, paving way to identify rumours.

Cao et al. (2018) define "*any piece of information, of which the veracity status was questionable at the time of posting*", as a rumour. They further claim that a rumour may later be verified to be true or false by other authorized sources. We follow their definition in this work; thus, we also define the task of rumour detection as: "*Given a piece of information from a social network, predict whether the piece of information is a rumour or not using the conversations which were induced by the said piece of information*". The initial piece of information could be a tweet or a user post, and the induced conversation would be the replies from other users (which we use as contextual information). Following the conventions in the literature, in this work, we refer to a main post and its replies as a *thread*.

In this Chapter, we utilize the semantic oppositeness embeddings created in Chapter IV to improve the rumour detection task, which has so far been restricted to only considering semantic similarity. We further prove that semantic oppositeness is well-suited to be applied to this domain, under the observation that rumour threads are more discordant than those of non-rumours. We further observe that, within rumour threads, false rumour threads continue to be clamorous; while true rumour

threads settle into inevitable acquiescence. We claim that semantic oppositeness can help in distinguishing this behavior as well. We propose word-level self-attention mechanism for the semantic oppositeness to augment the tweet level self-attention mechanism for the semantic similarity. We model the explicit and implicit connections within a thread, using a relevancy matrix. Unlike a regular adjacency matrix, our relevancy matrix recognizes the coherence of each sub-tree of conversation rooted at the main post, while acknowledging that, by definition, for this task, the main tweet must be directly related to all the rest of the tweets, regardless of the degrees of separation that may exist between them. We conduct extensive experiments to compare our proposed model with the state-of-the-art studies conducted on the same topic. To the best of our knowledge, this work is the first to utilize semantic oppositeness in rumour detection. In summary, our contributions in this Chapter include:

- We introduce a novel method for rumour detection, based on both semantic similarity and semantic oppositeness, utilizing the main post and the contextual replies.
- We model the explicit and implicit connections within a thread, using a relevancy matrix, which is then used to balance the impact semantic similarity and semantic oppositeness have on the overall prediction.
- We conduct experiments on recent rumour detection data sets and compare with numerous state-of-the-art baseline models to show that we achieve superior performance.

The remainder of this chapter is organized as follows: Section 6.2 provides a formal definition of the problem, along with our solution. It is followed by Section 6.3 discussing experiments and results. Finally the Section 6.4 concludes the chapter.

6.2 Methodology

We use a recent work (Veyseh et al., 2019) on rumour detection as our baseline. Their work, in turn, was heavily influenced by the earlier work on rumour detection in Twitter (Ma et al., 2018b). A tweet set I is defined as shown in Equation 6.1, where R_0 is the initial tweet and $R_1, R_2, \ldots, R_T$ are replies, such that T is the count of replies. Each tweet R_i is a sequence of words $W_1, W_2, \ldots, W_n$, such that n is the count of words. We tokenize the tweets; and in this work, *tokens* and *words* are used interchangeably. We also define the relevance matrix M, which carries the information of the tree structure of the tweet tree in Equation 6.2, where $A \star B$ denotes that A and B belong to the same tree in the forest obtained by eliminating the initial tweet. We show the process in Fig 27 as well. Our input is the pair $P = (I, M)$, which differs from Veyseh et al. (2019), where only I was used as the input. The entire data set is represented by D. The steps shown in Fig 27 are as follows:

1. Original tweet reply tree.
2. Obtain the forest by temporarily removing the root (main tweet).
3. Consider each tree in the forest to be fully connected graphs, and obtain the relevance matrices.
4. Obtain the full Relevance matrix by putting together the matrices from the previous step and considering the main tweet to be connected to all the other tweets.

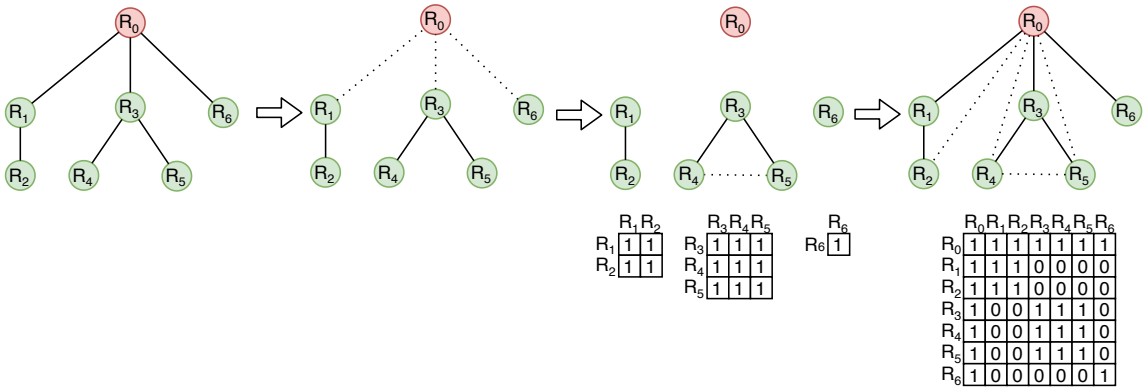


Figure 27. Relevance matrix building

$$I = (R_0, R_1, R_2, \ldots, R_T) \tag{6.1}$$

$$m_{i,j} = \begin{cases} 1 & \text{if } R_i = R_0 \vee R_j = R_0 \\ 1 & \text{if } R_i \star R_j \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

Following the convention of Veyseh et al. (2019) which is our baseline, we classify each pair (I, M) into four labels:

1. Not a Rumour (NR)
2. False Rumour (FR)
3. True Rumour (TR)
4. Unrecognizable (UR)

It should be noted that the distinction between "*False Rumour*" and "*True Rumour*" is drawn from the truthfulness of R_0.

Baseline works prior to Veyseh et al. (2019) have used RvNN to utilize the structural information in social networks, which bars the network from finding new relations among replies which are not explicitly connected. They have relaxed this constraint entirely, in hopes that the model would learn the relation among different

replies. In this work, we alter the dense adjacency matrix used for the self-attention based model to have the the constraint released in sub-trees but held among the aforementioned sub-trees.

This renders the work in this Chapter more general and able to capture different aspects of data than the baselines (Ma et al., 2018b), while being more efficient in computing than Veyseh et al. (2019). The computational speedup S of this work is given by the simple Equation 6.3, where n is the number of nodes in the thread, k is the sub-tree count, and m_i is the number of nodes in the i-th sub-tree excluding the global root.

$$S = \frac{n^2}{\left(\sum_{i=1}^{k} m_i^2\right) + 2n - 1} \tag{6.3}$$

The overall rumour detection model configuration is shown in Fig 28, where red vectors and node represent the main (root) tweet, and green vectors and nodes represent replies. Pooling operations are shown in boxes with dashed lines. The conventions and functionality of the model and its various components will be discussed in Section 6.2.1 to Section 6.2.5.

6.2.1 Formal Definition of Tweet Representation. Each tweet will have a different number of words n; thus, we pad the short tweets with a special token, until all the tweets have the same word length N as defined by 6.4.

$$N = \underset{P_i \in D}{\operatorname{argmax}}(n_i) \tag{6.4}$$

We build the representative oppositeness list O using the semantic oppositeness embeddings created in Chapter IV such that, for the i-th tweet R_i, with words $W_{i1}, W_{i2}, ..., W_{iN}$, the oppositeness vector O_i is created as $o_{i1}, o_{i2}, ..., o_{im}$ where o_{ij}

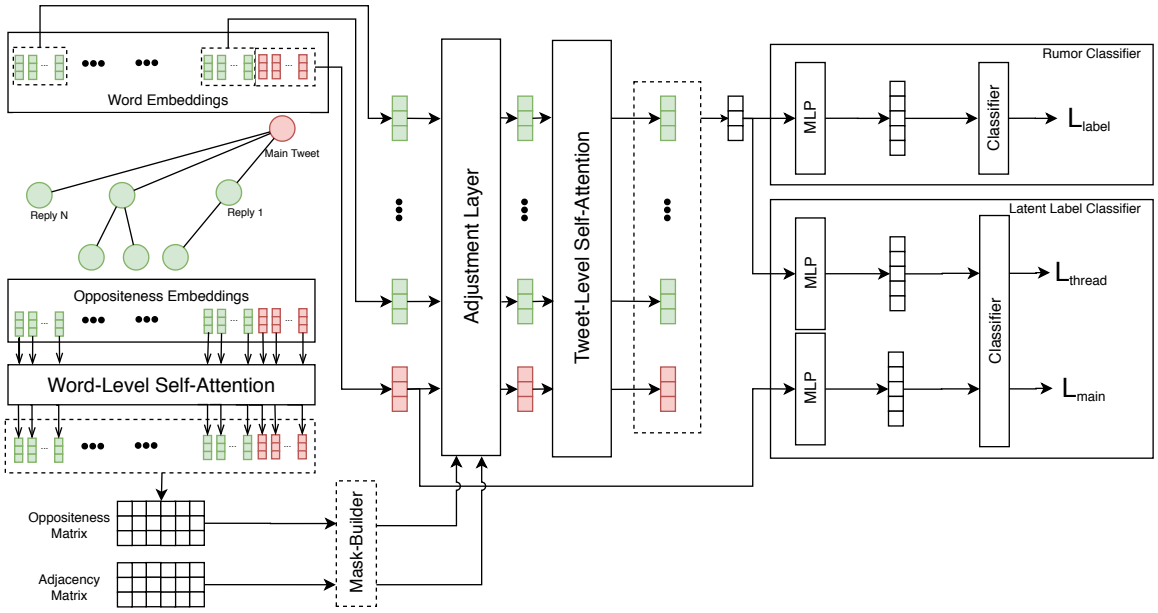


Figure 28. The overall rumour detection model configuration

is the embedding of W_{ij}. Note that $m \leq N$ where all tokens might not have corresponding oppositeness embeddings.

Each word in each tweet is then converted to a representative vector by means of a set of pre-trained word embeddings, such that for the i-th tweet R_i, with words $W_{i1}, W_{i2}, ..., W_{iN}$ is converted $e_{i1}, e_{i2}, ..., e_{iN}$. We then apply a max-pooling operation over the word embeddings along each dimension, resulting in a representative vector h_i coupled to R_i, as shown in Equation 6.5. At this point, note that the tweet set I of each pair P, which used to be $I = (R_0, R_1, R_2, \ldots, R_T)$, has been replaced by $I = (h_0, h_1, h_2, \ldots, h_T)$. It is this new representation which is passed to the following steps.

$$h_i = Elementwise\_Max(e_{i1}, e_{i2}, ..., e_{iN}) \tag{6.5}$$

6.2.2 Similarity-Based Contextualization. As discussed earlier, the Twitter data is organized as a tree rooted at the main tweet R_0 in each instance.

The earlier work by Ma et al. (2018b) proved that, in rumour detection, it is helpful to capture these relations among the main tweet and the replies. The subsequent work by Veyseh et al. (2019) noted that only considering explicit reply relation between the main tweet and other tweets neglects the the implicit relations among the tweets, arising from their semantic similarities (i.e., by the virtue of discussing the same topic, tweets in two separate branches may carry mutually useful information). Following this hypothesis, they exploited such implicit semantic relations for the purpose of improving the performance of the rumour detection task. However, in doing so, they abandoned the information garnered from the tree structure. In this work we propose to continue to use the implicit information, but to augment it with the information derived from the tree structure.

We initially follow the self-attention mechanism of Veyseh et al. (2019), which was inspired by the transformer architecture in Vaswani et al. (2017), to learn the pairwise similarities among tweets for capturing the semantic relations between the tweets. The process starts with calculating the key (k_i) and query (q_i) vectors for each tweet, based on its representation h_i, as shown in Equation 6.6.

$$\begin{aligned} k_i &= W_k * h_i + b_k \\ q_i &= W_q * h_i + b_q \end{aligned} \tag{6.6}$$

With the key and query vectors, we calculate the similarity a_{ij} between i-th and j-th tweets, using the dot product as shown in Equation 6.7, where γ is a normalization factor.

$$a_{i,j} = k_i \cdot q_j / \gamma \tag{6.7}$$

6.2.3 Oppositeness-Based Contextualization. Unlike in the case of similarity vectors, which were reduced to a single dimension at this point, the oppositeness representations are still at two dimensions. Thus the self-attention of oppositeness between tweets is handled at a word level, rather than at the tweet level. We build key ($k_i^{'}$) and query ($q_i^{'}$) vectors for each word based on its representation o_i, as shown in Equation 6.8.

$$\begin{aligned} k_i^{'} &= W_k * o_i + b_k \\ q_i^{'} &= W_q * o_i + b_q \end{aligned} \tag{6.8}$$

Since the oppositeness embedding of Chapter IV is based on Euclidean distance, with the key and query vectors, we calculate the oppositeness op_{i_x,j_y} between x-th word of i-th tweet and y-th word of j-th tweet using the Euclidean distance, as shown in Equation 6.9 where $k_{i_x}^{'}$ is the key vector for x-th word of i-th tweet, $q_{j_y}^{'}$ is the query vector for y-th word of j-th tweet, and Euclidean distance $d(,)$ is calculated across the size of the oppositeness embedding vector.

$$op_{i_x,j_y} = d(k_{i_x}^{'}, q_{j_y}^{'}) \tag{6.9}$$

To obtain the abstract tweet-level oppositeness, we apply element-wise average-pooling on the $OP_{i,j}$ matrix, as shown in Equation 6.10, to create the oppositeness matrix $O^{"}$, where δ is the oppositeness vector count of the i-th tweet, and ϱ is the oppositeness vector count of the j-th tweet. Note that the dimensions of the oppositeness matrix $O^{"}$ are the same as the relevance matrix M.

$$o^{"}_{i,j} = Elementwise\_Average\left(\begin{bmatrix} op_{i_0,j_0} & op_{i_1,j_0} & \dots & op_{i_\delta,j_0} \\ op_{i_0,j_1} & op_{i_1,j_1} & \dots & op_{i_\delta,j_1} \\ \dots & \dots & \dots & \dots \\ op_{i_0,j_\varrho} & op_{i_1,j_\varrho} & \dots & op_{i_\delta,j_\varrho} \end{bmatrix} \right) \tag{6.10}$$

Next we create the oppositeness mask Ω by average-pooling $O^{"}$ along rows and columns, as shown in Equation 6.11, where similar to Equation 6.4, n_i and n_j are natural lengths of the i-th and j-th tweets respectively.

$$\begin{aligned} \omega_{i,j} = 1 - Elementwise\_Average(o^{"}_{i,0}, o^{"}_{i,1}, ..., o^{"}_{i,n_j}) \\ -Elementwise\_Average(o^{"}_{0,j}, o^{"}_{1,j}, ..., o^{"}_{n_i,j}) \end{aligned} \tag{6.11}$$

6.2.4 Deriving Overall Thread Representations. Similar to the oppositeness mask Ω, we create the relevance mask Ψ by sum-pooling M along rows and columns, as shown in Equation 6.12, where similar to Equation 6.4, n_i and n_j are natural lengths of the i-th and j-th tweets respectively.

$$\begin{aligned} \psi_{i,j} = Elementwise\_Sum(m_{i,0}, m_{i,1}, ..., m_{i,n_j}) \\ +Elementwise\_Sum(m_{0,j}, m_{1,j}, ..., m_{n_i,j}) \end{aligned} \tag{6.12}$$

At this point we diverge from Veyseh et al. (2019) in two ways, and we utilize the related relevance mask M as a weighting mechanism, with proportion constant α (where $0 < \alpha < 1$), as well as the oppositeness mask OM, to obtain augmented attention $a^{'}_{i,j}$ as shown in Equation 6.13.

$$a^{'}_{i,j} = a_{i,j}\omega_{i,j}\Big[(\psi_{i,j} - \alpha)^2 + \alpha\psi_{i,j}\Big] \tag{6.13}$$

We utilize the augmented similarity values $a'_{i,j}$ for each tweet pair in the thread to compute abstract representations for the tweets based on the weighted sums, as shown in Equation 6.14.

$$h'_i = \Sigma_j a'_{i,j} * h_j \tag{6.14}$$

Next, we apply the max-pooling operation over the processed tweet representation vectors h'_i to obtain the overall representation vector h' for the input pair P as shown in Equation 6.15.

$$h' = Elementwise\_Max(h'_0, h'_1, h'_2, ..., h'_T) \tag{6.15}$$

Finally, the result is sent through a 2-layer feed-forward neural network capped with a softmax layer, producing the probability distribution $P(y|R_0, R_1, R_2, \ldots, R_T; \theta)$ over the four possible labels, where θ is the model parameter. On this, we optimize the negative log-likelihood function, in order to train the model, as shown in Equation 6.16, where y^* is the expected (correct) label for I.

$$L_{label} = -\log P(y^*|R_0, R_1, R_2, \ldots, R_T; \theta) \tag{6.16}$$

6.2.5 Main Tweet Information Preservation. The Veyseh et al. (2019) study noted that the model by Ma et al. (2018b) treats all tweets equally. This was deemed undesirable, given that the main tweet of each thread incites the conversation, and thus, arguably, carries the most important content in the conversation, which should be emphasized, to produce good performance. To achieve this end, it was proposed to bring forward the information in the main tweet independently of and

separately from that of the collective Twitter thread, in order to provide a check. We, in this work, also provide this sanctity check, to enhance the obtained results.

The basic idea is that, by virtue of definition, if a main tweet is a rumour (or not), unique trait and information pertaining to that class should be in the main tweet itself. Thus, the latent label (L_{thread}) obtained by processing the thread representation h' above should be the same as a potential latent label (L_{main}) obtained by processing the representation of the main tweet h_0. To calculate L_{main}, we use a 2-layer feed-forward neural network with a softmax layer in the end, where it assigns the latent labels drawn from K possible latent labels. Next, we use another 2-layer feed-forward neural network with a softmax layer in the end, assigning the same K number of possible latent labels as shown in the negative log-likelihood function to match it with the thread.

$$L_{main} = argmax_L P(L|R_0) \tag{6.17}$$

$$L_{thread} = -\log P'(L_{main}|R_0, R_1, R_2, \ldots, R_T) \tag{6.18}$$

Finally, the loss function to train the entire model is defined as in Equation 6.19, where the L_{label} is obtained from Equation 6.16, and β is a hyper-parameter which controls the contribution of the main tweet information preservation loss to final loss.

$$Loss = L_{label} + \beta L_{thread} \tag{6.19}$$

6.3 Experiments

We use the *Twitter 15* and *Twitter 16* data sets introduced by Ma, Gao and Wong (2018a) for the task of rumour detection. Respectively, there are 1381 and 1118 tweet threads in each data set. We use Glove (Pennington et al., 2014) embedding to

initialize the word vectors and oppositeness embedding from Chapter IV to initialize the oppositeness vectors. Both embedding vectors are of size 300. Key and query vectors in Equations 6.6 and Equations 6.8 employ 300 hidden units. The rumour classifier feed-forward network has two layers of 200 hidden units. The feed-forward layer in the main tweet information preservation component has two layers, each with 100 hidden units, and it maps to three latent labels. The proportion constant α, which balances the explicit and implicit information, is set at 0.1. The loss function uses a trade-off parameter of $\beta = 1$, with an initial learning rate of 0.3 on the *Adagrad* optimizer. For the purpose of fair results comparison, we follow the convention of using 5-fold cross validation procedure to tune the parameters set by Ma et al. (2018b).

6.3.1 Comparison to the State-of-the-Art Models. We compare the proposed model against the state-of-the-art models on the same data sets. The performance is compared by means of overall accuracy and F1 score per class. We observe that there are two types of models against which we compare. The first type are the *feature-based models*, which used feature engineering to extract features for Decision Trees (Castillo et al., 2011; Zhao, Resnick & Mei, 2015), Random Forest (Kwon, Cha, Jung, Chen & Wang, 2013), and SVM (Ma, Gao, Wei, Lu & Wong, 2015; Ma et al., 2017; Wu, Yang & Zhu, 2015). The second type of models are *deep learning models*, which used Recurrent Neural Networks or Recursive Neural Networks to learn features for rumour detection. We compare our model to GRU-RNN proposed by Ma et al. (2016), BU-RvNN and TD-RvNN proposed by Ma et al. (2018b), and Semantic Graph proposed by Veyseh et al. (2019). Results for Twitter 15 and Twitter 16 are shown in Tables 8 and 9, respectively.

It is evident from these tables that, in the rumour detection task, the deep learning models outperform feature-based models, proving that automatically

Table 8. Model Performance on Twitter 15.

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
DTR (Zhao, Resnick & Mei, 2015)	0.409	0.501	0.311	0.364	0.473
DTC (Castillo, Mendoza & Poblete, 2011)	0.454	0.733	0.355	0.317	0.415
RFC (Kwon, Cha, Jung, Chen & Wang, 2013)	0.565	0.810	0.422	0.401	0.543
SVM-TS (Ma, Gao, Wei, Lu & Wong, 2015)	0.544	0.796	0.472	0.404	0.483
SVM-BOW (Ma, Gao & Wong, 2018b)	0.548	0.564	0.524	0.582	0.512
SVM-HK (Wu, Yang & Zhu, 2015)	0.493	0.650	0.439	0.342	0.336
SVM-TK (Ma, Gao & Wong, 2017)	0.667	0.619	0.669	0.772	0.645
GRU-RNN (Ma et al., 2016)	0.641	0.684	0.634	0.688	0.571
BU-RvNN (Ma, Gao & Wong, 2018b)	0.708	0.695	0.728	0.759	0.653
TD-RvNN (Ma, Gao & Wong, 2018b)	0.723	0.682	0.758	0.821	0.654
SG (Veyseh, Thai, Nguyen & Dou, 2019)	0.770	0.814	0.764	0.775	**0.743**
Semantic Oppositeness Graph (SOG)	**0.796**	**0.825**	**0.820**	**0.814**	0.742

Table 9. Model Performance on Twitter 16.

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
DTR (Zhao, Resnick & Mei, 2015)	0.414	0.394	0.273	0.630	0.344
DTC (Castillo, Mendoza & Poblete, 2011)	0.465	0.643	0.393	0.419	0.403
RFC (Kwon, Cha, Jung, Chen & Wang, 2013)	0.585	0.752	0.415	0.547	0.563
SVM-TS (Ma, Gao, Wei, Lu & Wong, 2015)	0.574	0.755	0.420	0.571	0.526
SVM-BOW (Ma, Gao & Wong, 2018b)	0.585	0.553	0.655	0.582	0.578
SVM-HK (Wu, Yang & Zhu, 2015)	0.511	0.648	0.434	0.473	0.451
SVM-TK (Ma, Gao & Wong, 2017)	0.662	0.643	0.623	0.783	0.655
GRU-RNN (Ma et al., 2016)	0.633	0.617	0.715	0.577	0.527
BU-RvNN (Ma, Gao & Wong, 2018b)	0.718	0.723	0.712	0.779	0.659
TD-RvNN (Ma, Gao & Wong, 2018b)	0.737	0.662	0.743	0.835	0.708
SG (Veyseh, Thai, Nguyen & Dou, 2019)	0.768	0.825	0.751	0.768	**0.789**
Semantic Oppositeness Graph (SOG)	**0.826**	**0.843**	**0.843**	**0.878**	0.774

learning effective features from data is superior to hand-crafting features. We also note that the *Semantic Oppositeness Graph*, along with the Semantic Graph, and other RvNN models with GRU-RNN, generally do well, which attests to the utility of structural information, be it in the form of reply structure or be it in the form of semantic relations, in helping to improve performance. We further notice that Veyseh et al. (2019), which uses implicit information, outperforms TD-RvNN Ma et al. (2018b), which only uses explicit information. *Semantic Oppositeness Graph*, which uses explicit information, implicit information, and semantic oppositeness, outperforms all the other models in accuracy, while outperforming all the other

models in three out of four classes, in terms of F1 Score. The one class in which *Semantic Oppositeness Graph* loses out to Veyseh et al. (2019) is in the case of the *Unrecognizable* (UR) class. We argue that this is not an issue, given that the unrecognizable class consists of tweets which were too ambiguous for human annotators to tag as one of: not a rumour (NR), false rumour (FR), or true rumour (TR). We assert that Tables 8 and 9 clearly demonstrate the effectiveness of the proposed *Semantic Oppositeness Graph* method in the task of rumour detection.

6.3.2 Model Stability Analysis. While comparing our system with Veyseh et al. (2019), which we use as our main baseline, we noticed that their system has a high variance in results, depending on the random weight initialization. This was impactful in such a way that in some random weight initializations, the accuracy of their system could fall as low as 24% from the reported high 70% results in their paper. Given that we use their system as our baseline and the basis for our model, we decided to do a stability analysis between their system and ours. For this purpose, we created 100 random seeds and trained four models with each seed, resulting in a total of 400 models. The models were:

1. Veyseh et al. (2019) on *Twitter 15*
2. Veyseh et al. (2019) on *Twitter 16*
3. *Semantic Oppositeness Graph* on *Twitter 15*
4. *Semantic Oppositeness Graph* on *Twitter 16*

Then we normalized the results of the Veyseh et al. (2019) models to the values reported their paper (also shown in the relevant row on Tables 8 and 9). Each result is reported in the format of $\frac{\mu}{\sigma}$.

From the results in Tables 10 and 11, it is evident that our *Semantic Oppositeness Graph* has higher mean values for accuracy, not a rumour (NR), false rumour (FR),

Table 10. Model Variance Performance on Twitter 15.

Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
Semantic Graph	0.770	0.814	0.764	0.775	0.743
(Veyseh, Thai, Nguyen & Dou, 2019)	0.138	0.133	0.198	0.118	0.129
Semantic Oppositeness Graph	0.796	0.825	0.820	0.814	0.742
(SOG)	0.089	0.080	0.109	0.093	0.100

Table 11. Model Variance Performance on Twitter 16.

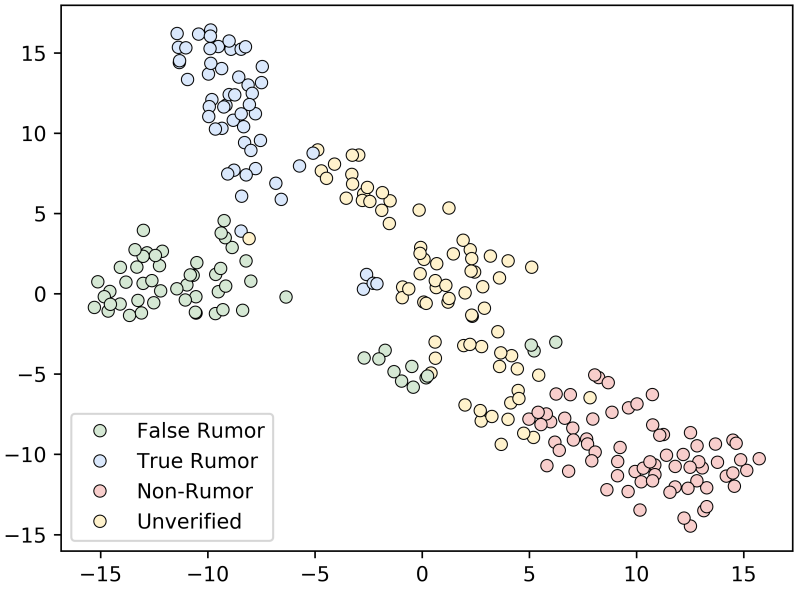
Model	Accuracy	F1 NR	F1 FR	F1 TR	F1 UR
Semantic Graph	0.768	0.825	0.751	0.768	0.789
(Veyseh, Thai, Nguyen & Dou, 2019)	0.103	0.226	0.103	0.096	0.184
Semantic Oppositeness Graph	0.826	0.843	0.843	0.878	0.774
(SOG)	0.082	0.153	0.091	0.074	0.114

and true rumour (TR), while having comparably reasonable values for Unrecognizable (UR) class. But more interesting are the standard deviation values. It is evident that in all cases, our model has smaller standard deviation values than that of Veyseh et al. (2019). This is proof that our system is comparatively more stable in the face of random weight initialization. We argue that this stability comes from the introduction of the oppositeness component, which augments the decision-making process with the oppositeness information, as a counterpart for the already existing similarity information, preventing the predictions from having a swinging bias.

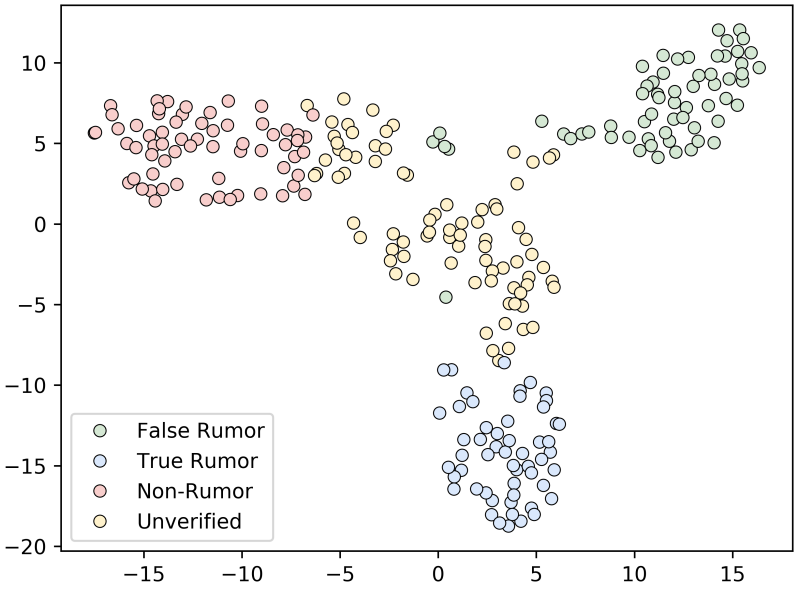
For a demonstration, consider the subset of three words *increase*, *decrease*, and *expand* from the example used in Chapter III. If the main tweet (R_0) were to say "*A will increase B*", R_1 replied with "*A will decrease B*", and R_2 replied with "*A will expand B*", then the purely semantic similarity based model will position R_0 and R_1 closer than R_0 and R_2, given that the word contexts in which *increase* and *decrease* are found are more similar than the word contexts in which *increase* and *expand* are found. This would result in the neural network having to learn the opposite semantics between *increase* and *expand* by itself, during the training, making

it more vulnerable to issues of bad initial weight selection. This, in turn, will result in greater variance among the trained models. However, a system with an oppositeness component will already have the opposite semantics between *increase* and *decrease*, as well as *increase* and *expand* already calculated. Thus, such a system would have pre-knowledge that the word pair *increase* and *decrease*, despite being used in more common contexts, is more semantically opposite than the word pair *increase* and *expand*, which is used in less common contexts. Hence the neural network does not have to learn that information from scratch during the training, resulting in it being less vulnerable to issues of bad initial weight selection. Analogously, this, in turn, will result in lesser variance among the trained models; hence, explaining the better stability demonstrated by *Semantic Oppositeness Graph* in comparison to Veyseh et al. (2019) in Tables 10 and 11.

6.3.3 Impact of the Oppositeness Component. Finally, to emphasize the effect the oppositeness component has on the model, we draw the t-SNE diagrams (Linderman & Steinerberger, 2019; Maaten & Hinton, 2008) for the final representations of the threads. Figure 29a shows the data points clustering when the model is trained without the oppositeness component, and Fig. 29b shows the data points clustering when the model is trained with the oppositeness component. Note that all other variables, including the seed for the weight initializer, are the same in the two models. These diagrams prove that the oppositeness component helps improve the separability of the classes. Specially note how the *False Rumour* and *True Rumour* classes are now more clearly separated. We postulate that this derives from the fact that the oppositeness component would help in distinguishing the continuous discord happening in a *False Rumour* thread from the subsequent general agreement in a *True Rumour* thread.



(a) Without the Oppositeness Component



(b) With the Oppositeness Component

Figure 29. t-SNE diagrams for thread representations.

6.4 Conclusion

Rumours and fake news are a significant problem in social networks, due to their intrinsic nature of connecting users to millions of others and giving any individual the power to post anything. We introduced a novel method for rumour detection, based on semantic oppositeness, in this chapter. We demonstrated the effectiveness of our method using data sets from Twitter. Compared to previous work, which only used explicit structures in the reply relations or implicit relations for semantic similarity, our model learns both explicit and implicit relations between a main tweet and its replies, by utilizing both semantic similarity and semantic oppositeness. We proved, with extensive experiments, that our proposed model achieves state-of-the-art performance, while being more resistant to the variances in performance introduced by randomness.

Part C

Summary and Future Work

In this part of the dissertation, we discover, synthesize, recommend best practices on, and make observations on the usage of semantic oppositeness measurement of Parts A and B.

CHAPTER VII

SUMMARY

The overarching goal of this dissertation work is to investigate and address the dissertation question: *How can we convert the linguistic concept of semantic oppositeness to the computing domain?* This summarising chapter addresses the dissertation question by synthesizing the findings and best practices of the previous chapters.

Under the dissertation question, we investigated four problems related to semantic oppositeness. First, given the linguistic definition of semantic oppositeness and the two principles under which it works, (*Minimal difference with maximal similarity principle* and *Irrelevancy principle*), how can we convert the idea into a computational metric? Second, how can we overcome, using embedding, the limitations of the semantic oppositeness measure that we inherited from the restrictions of antonymy present in language and reflected in WordNet? Third, we applied our knowledge to the target of PubMed abstracts to answer the question, how can we find inconsistencies that arise on the path to scientific progress when new knowledge supplants old knowledge? In this step, we also explored how the same inconsistency finding can be leveraged against semantic similarity to suggest relationships to an ontology. Fourth, we applied our knowledge to the target of social media posts to answer the question, how can we find disagreement in social media discussion threads, and thereby classify which posts constitute a rumour and which do not? Further, at this point we discovered that the nature of semantic oppositeness grants us the ability to specifically distinguish between true-rumours and false-rumours, by the virtue that true-rumour threads settle into acquiescence, while false-rumor threads continue ad infinitum, to the clamour of disagreement.

This dissertation consisted of six chapters, each based primarily on original research by myself and various co-authors, either previously published, or under review. The content of each chapter is summarized as follows. Chapter I introduced and described the motivation for the dissertation question, as well as outlining the contents of the dissertation. Chapter II reviewed fundamental concepts and related work for the linguistic base of semantic oppositeness, as well as the computing resources and techniques that we employ to convert semantic oppositeness to the computational linguistics domain and to test on subsequent use cases. Chapter III introduced the new semantic oppositeness measure in a formal, mathematical setting, with computerized implementations following the linguistic observations and principals of Chapter II. Chapter IV took the formalized computational model of semantic oppositeness from Chapter III and embedded it in a vector space to attain better generalizability and efficiency. Chapter V applied the basic semantic oppositeness measure of Chapter III to find inconsistencies among research paper abstracts, as well as to find potential strong relationships to be added to an ontology in a relevant domain. Chapter VI applied the embedded semantic oppositeness measure of Chapter IV to detect rumours in social networks by means of discovering disagreements.

The main contributions of the dissertation are summarized as follows:

1. We introduce a novel metric by which semantic oppositeness can be calculated, adhering to the *Minimal difference with maximal similarity principle* and *Irrelevancy principle* in linguistics. For this purpose, we utilize the semantic similarity measures, as well as antonymy relationships.
2. We introduce a semantic oppositeness embedding which takes the above metric and embeds it in a vector space to obtain better generalizability and

efficiency. In the process of this, we also introduce a novel, unanchored vector-embedding approach and a novel, *inductive transfer learning* process based on auto encoders, which utilizes both learnt embeddings and the learnt latent representation.

3. We utilize our semantic oppositeness measure on the question of finding inconsistencies with PubMed abstracts as a use case, where we show the effectiveness of our methodology in the given question. We illustrate how this novel semantic oppositeness measure is superior to the antonym method and to the naïve similarity inverse method. While working on this use case, we produce the following sub-contributions:

   (a) We introduce an ontology-based information extraction model to discover inconsistencies in PubMed abstracts.
   (b) We propose a new methodology to incorporate open information extraction into an ontology-based information extraction process, in order to compensate for the lack of relationships in the domain ontology.
   (c) We propose an ontology-based information extraction model to discover similarities in PubMed abstracts, which then leads to compiling possible updates to OMIT in order to fix its problem of lack of relationships.

4. We utilize our embedded semantic oppositeness measure on the question of finding disagreement in social media threads as a use case, where we show the effectiveness of our methodology in the given question under the following aspects:

   (a) We introduce a novel method for rumour detection, based on both semantic similarity and semantic oppositeness, utilizing the main post and the contextual replies.

(b) We model the explicit and implicit connections within a thread, using a relevancy matrix, which is then used to balance the impact semantic similarity and semantic oppositeness have on the overall prediction.

(c) We conduct experiments on recent rumour detection data sets and compare with numerous state-of-the-art baseline models to show that we achieve superior performance.

The experiments validate the effectiveness of the semantic oppositeness measure for these problems.

CHAPTER VIII

FUTURE WORK

This concluding chapter presents our recommendations for future research that would build upon and further substantiate the work presented in this dissertation. Before getting on to the recommendations, we can make observations on the current usage of our semantic oppositeness model. From the citations to our papers, we observe that our semantic oppositeness has been considered in a number of domains so far:

1. *Biomedical domain*: Given that our initial use case publication was in this domain, it is understandable that relevant work in this area would try to utilize the metric. Under this, we observe topics such as: pattern discovery (Li et al., 2018; Wang, Zhang, Li, Chen & Han, 2018b), textual entailment (Tawfik & Spruit, 2019a; Tawfik & Spruit, 2019b), contradiction detection (Tawfik & Spruit, 2018), automated labeling (Teng, Bai & Li, 2019).
2. *Legal domain*: The legal domain works with arguments and disagreements. Therefore, NLP tasks in the legal domain benefit from working with semantic oppositeness. In the legal domain, we see research referring to semantic oppositeness on: domain-specific semantic similarity (Sugathadasa et al., 2017), and shift-of-perspective identification (Ratnayaka et al., 2019b).
3. *Knowledge/Ontology domain*: Ontology representation (Jayawardana et al., 2017b) and population (Jayawardana et al., 2017a; Jayawardana et al., 2017c) are two tasks in which our semantic oppositeness has been cited.

We provide a number of recommendations for future studies in Sections 8.1 through 8.6. They range from altering the fundamental method of oppositeness

calculation discussed in Section 8.1, to extending the oppositeness calculation and embedding process discussed in Sections 8.2, 8.3, and 8.4, to some further use cases where oppositeness measure can be utilised, as discussed in Sections 8.5 and 8.6.

8.1 Alternate Oppositeness Calculations

Future studies may explore the possibility of modelling the *Minimal difference with maximal similarity principle* (Section 2.1.1) and the *Irrelevancy principle* (Section 2.1.2) differently from to how we have done in Chapter III and compare. We have released code, intermediate outputs, and final outputs of our work, as mentioned throughout the dissertation. Consider the Equation 3.12, and consider the variant in Equation 8.1 where the result has been scaled back to 0 to 1 range. This would yield Fig. 30 and Fig. 31. As it can be observed, this setting reinforces *Minimal Difference with Maximal Similarity Principle* of Section 2.1.1, but it does not improve our ability to support the *Irrelevancy Principle* of Section 2.1.2 as much as our Equation 3.12 does. But for a future study with that flavour of leaning, this variant may be valid.

$$oppo\_ori_{w_1,w_2} = \frac{(reldif_{w_1,w_2} + 1)^{\left(K * simil_{w_1,w_2} + 1\right)}}{2^{K+1}} \tag{8.1}$$

By replacing the constant 1 in Equation 8.1 with a very small constant ϵ, it is possible to obtain the Equation 8.2. As shown in Fig 32 and Fig 33, this approximates the combined oppositeness of Equation 3.14 but without the fine control granted by α. In our work, we considered this option first, but we later opted for Equation 3.14, for the flexibility of α control. But a future implementation willing to reduce the computational complexity at the cost of fine control can utilize this version. It would perhaps be suitable for a situation where a large number of approximate oppositeness values needs to be calculated with low resources and then go into a few short-listed instances to obtain more accurate results. In that case, we can recommend

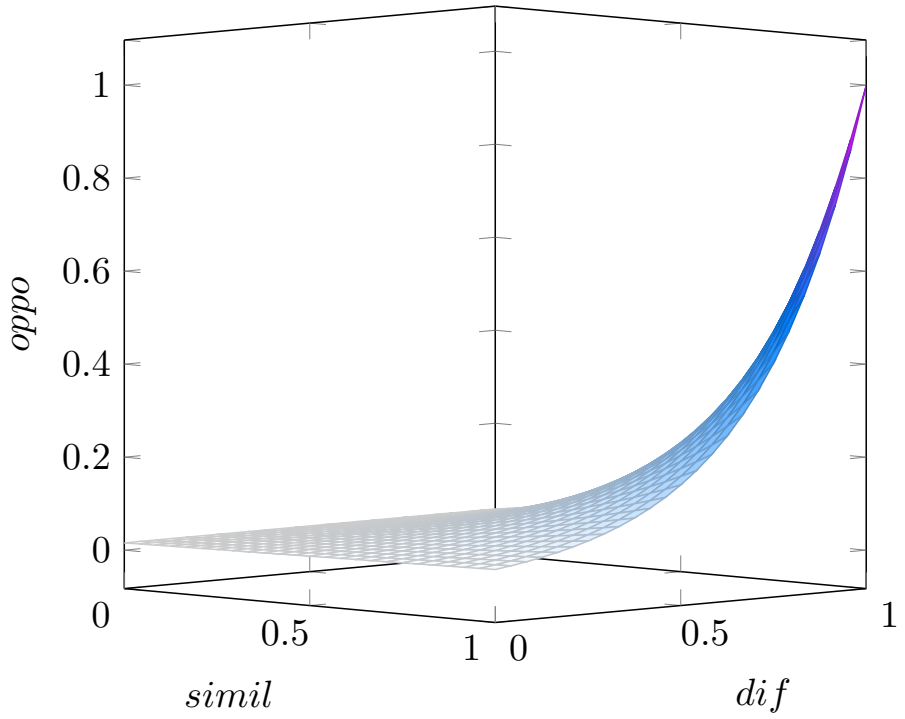


Figure 30. Variant Oppositeness function: 3D plot

Equation 8.2 for the mass scale approximation step and Equation 3.14 for the short-listed precise step.

$$oppo\_ori_{w_1,w_2} = \frac{(reldif_{w_1,w_2} + \epsilon)^{\left(K * simil_{w_1,w_2} + \epsilon\right)}}{2^{K+\epsilon}} \tag{8.2}$$

Another useful suggestion with regard to this is to use the semantic similarity values obtained from a word embedding system to alter or replace Equation 3.7. Given that *antonymy* would stay as a static one-to-many relationship in the generic domain, there is no need to replace the functionality of Equation 3.9. However, the latter assessment becomes invalid in the situation discussed in Section 8.4, given that the *antonymy* property being valid for a word pair in the generic domain does not necessarily imply that it is valid in a specialized or specific domain. In that situation, a large corpus of words annotated by a human expert of the domain would be needed, in

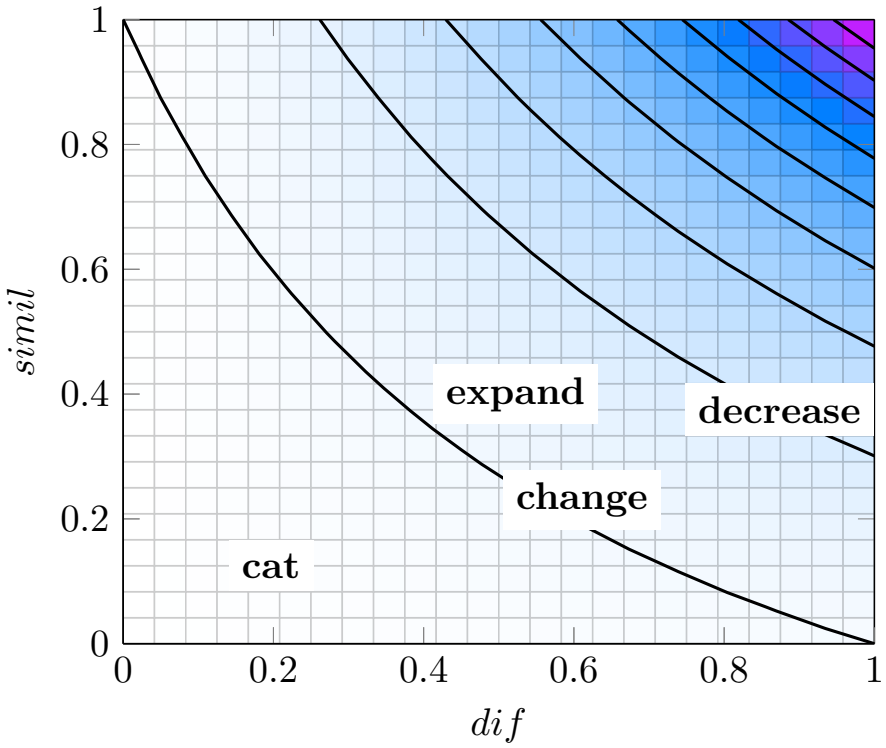


Figure 31. Variant Oppositeness function: Contour plot

order to proceed. This is because antonyms do not necessarily carry a shared linguistic root. Regardless of the fact that the considered pair of antonyms are *canonical* (Cruse, 1986) or otherwise (Jones, 2003), this situation may arise. (Naturally, a majority of *canonical* antonyms do share linguistic roots, while diversity is more apparent in *non-canonical* antonyms.) This burden on the process of learning required antonyms to be used in the oppositeness measure may be lightened or augmented by corpus-driven methods, as suggested by Lobanova (2012); but for the sake of precision, a significant amount of manual tagging has to be done by an expert.

8.2 Phrase and Sentence Level Oppositeness

Our work has been focused mainly on word-level oppositeness, other than in the case of Chapter V, where relationships of triples were compared. But even at that implementation, the minute comparison was among words. Therefore, a future study

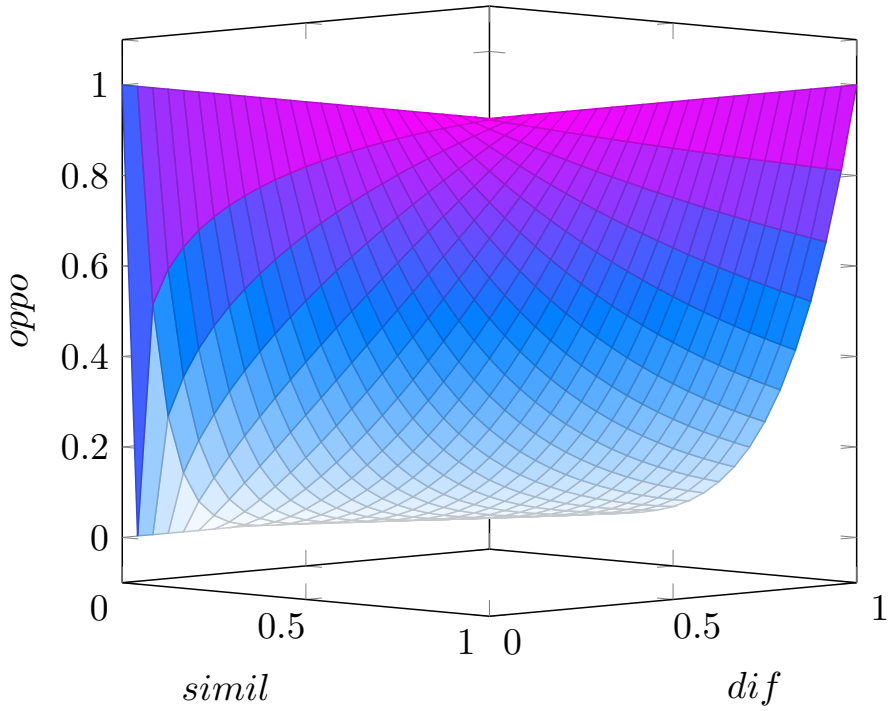


Figure 32. Oppositeness function with small ϵ: 3D plot

may look into calculating semantic oppositeness at the level of phrases and sentences. This may also reveal whether *the whole* has emergent properties that go beyond *the sum of its parts*. However, this would exponentially increase the number of instances to be regarded, while decreasing the support for each unique entry. That would result in a demand for larger corpi, as well as possibly causing a deficiency in areas where the phrases or sentences tend to run long. Nevertheless, for a reasonably large corpus consisting of short sentences and phrases, this application should be viable. However, the greatest hurdle of such an application would be the translation of Equation 3.9 to the higher level. A naïve solution would be a pairwise comparison of each word in the two phrases (or sentences). However, this would run into the issue of most word pairs not being natural antonyms of each other and thus producing a very sparsely populated matrix as the calculated values. This, in turn, can deteriorate the quality of the final oppositeness value. A solution for this would come from either

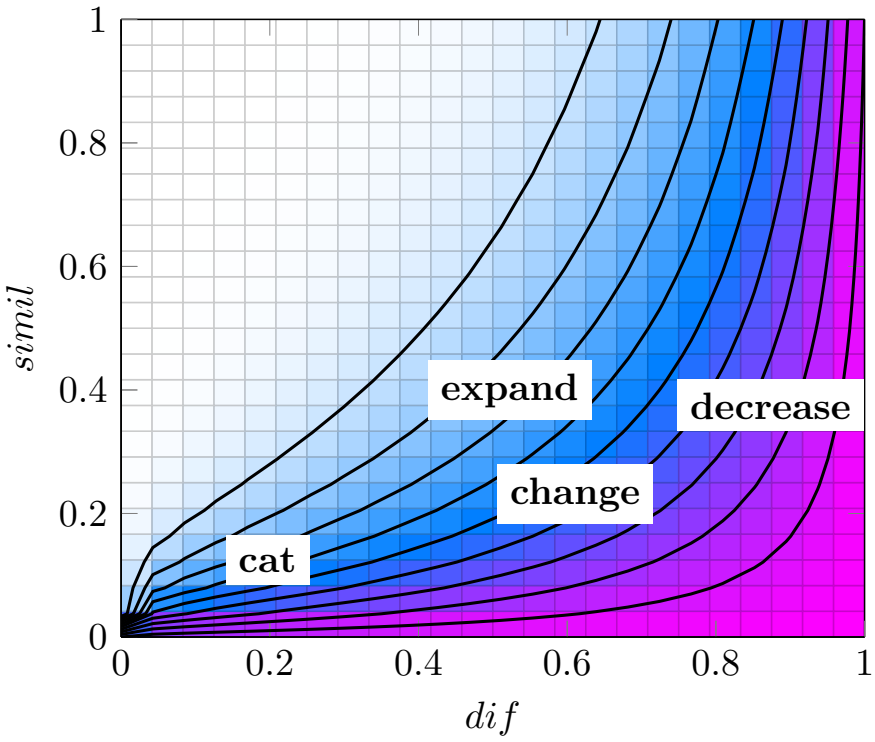


Figure 33. Oppositeness function with small ϵ: Contour plot

an attention approach, which learns the relative importance of each word, or a parse tree-based approach, where the trees are aligned before word-level comparisons, which are restricted by the local connections of words and phrases discovered along the parse tree. Of course, it is also possible to merge the above two solutions and attempt a hybrid approach. Regardless of the approach selected, future work on this direction will heavily depend on obtaining or creating large human-annotated data sets.

8.3 Higher-Level Oppositeness Embedding

Related to the direction in Section 8.2 and with the advent of embedding systems for phrases (Mikolov et al., 2013a; Wu, Zhao & Li, 2020; Yin & Schütze, 2016), sentences (Lin et al., 2017; Palangi et al., 2016), paragraphs (Kawamura, Watanabe, Matsumoto, Egami & Jibu, 2018; Wang, Zhang, Ding & Zou, 2017), and documents (Dai, Olah & Le, 2015; Lau & Baldwin, 2016) it is possible to extend

our word-level oppositeness embedding to higher levels of text such as phrases or sentences. This may even shed a light on the illusive sarcasm problem in NLP (Cai, Cai & Wan, 2019; Castro et al., 2019; Pan, Lin, Fu, Qi & Wang, 2020). However, as discussed in Section 8.2, the issue of the thinning-out of support would come into play here, as well. When the phrases and sentences become longer, they become more unique; and thus, the probability of them reoccurring in a given corpus becomes low. This would result embeddings that are, at best, imperfect, and at worst, incorrect. As a solution to this, it might be possible to incorporate a layered embedding, where the embedding of phrases is derived from that of words, and the embedding of sentences is derived from that of phrases. It may also be possible to obtain already-embedded phrase or sentence models and use a transfer learning method similar to what we have discussed in Section 4.2.4. However, given that this step is largely dependant on the successful implantation of the methodologies we have discussed in Section 8.2, by the virtue of transitive property, this extension too will be heavily dependant on human-annotated large corpi.

8.4 Domain Dependent Oppositeness Embedding

Word embedding is moving towards higher context sensitivity, with implementations such as BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019), which embed words with consideration to various senses that they carry. We also observed proof of such word senses altering depending on domain in the case of Section 5.2.4 for the medical domain. Further proof of this phenomenon in the legal domain is discussed by Sugathadasa et al. (2017). Thus it is arguable that, if semantic similarity, which can be derived by word embeddings, is capable of being altered on the basis of domain, the same should be true for oppositeness. The same linguistic forces which have *robbery* and *burglary* play synonym roles in common vernacular while being drastically

different in the legal domain will play the mirrored role in the case of oppositeness. As such, we postulate that semantic oppositeness embedding with context clues from the domain would prove to be better in the context of that particular domain. However, this task is not currently achievable with the oppositeness calculation methodology proposed in this dissertation, and it would need a massive corpus of expert annotated data, for any domain on which one wishes to implement this. Once such a system is in place, it could be utilised for applications such as: domain specific sentence classification using discourse and argumentative properties (Ratnayaka et al., 2019a), question retrieval (Wang et al., 2018a), domain specific sentiment analysis (Gamage et al., 2018; Mudalige et al., 2020; Rajapaksha et al., 2020; Ratnayaka, de Silva, Perera & Pathirana, 2020).

8.5 Use Case: Hate Speech Detection

Extending on our rumour detection use case in Chapter VI, we suggest that the semantic oppositeness measure can be effectively utilized in detecting and suppressing hate speech, given that hate speech, by nature, goes opposite to the regular accepted speech on social media. However, this extension heavily leans on an assumption of online morality (Shin, 2008) having a benevolent bias, given that the expected clamour of the hate speech threads will rise from a shared distaste of such statements. This sweeping assumption is not always held true. For example one could point out the alleged toxicity of platforms such as *4chan* (Bernstein et al., 2011; Nagle, 2017). In such a scenario, hate speech might not raise an opposite response but instead raise approval. This problem in the analysis will have to be overcome by collaborating with anthropologists and social scientists. On platforms which are not such edge cases, using the oppositeness measure to detect hate speech should work under the same principles as rumour detection. A second obstacle that might need to be overcome

in this use case, as well as in any future extension of the rumour detection use case in Chapter VI, would be the handling of slang (Mapa et al., 2012). Given the purely academic and theoretical root of our methodology, the basis of our system does not cover the modern slang. The fact that new slang words would appear from time to time, as well as the fact that slang words may change meaning as the time goes, introduces unique challenges for an implementation of an oppositeness measure. Then there is the issue of slang slowly becoming accepted into mainstream usage. Consider the following examples:

1. The original definition of the word "*awesome*" was *inspiring awe*. However, in current usage, it is used to mean *terrific* or *extraordinary*[1].

2. The case of the word "*literally*" is even more extreme. It has the original meaning of *literal sense or manner*. However, in current usage, it is also used to indicate the very opposite of its original meaning: *virtually* or *figuratively*[2].

Such shifts and changes in opposite behaviour will have a great impact on both the designing as well as the effectiveness of the oppositeness measure on this use case. The solution to this problem may lie in the discussion we presented in Section 8.4 where it is possible to register varying oppositeness values without conflict, akin to how word senses are being handled by models such as BERT (Devlin et al., 2018) or XLNet (Yang et al., 2019). It would also be important to see how this can be applied to languages other than English (de Silva, 2019; Wijeratne & de Silva, 2020; Wijeratne, de Silva & Shanmugarajah, 2019).

[1]https://www.merriam-webster.com/dictionary/awesome

[2]https://www.merriam-webster.com/dictionary/literally

8.6 Use Case: Paronomasia Detection

In my work towards this dissertation, I briefly explored (de Silva, 2017a) the possibility of using semantic oppositeness to detect paronomasia (puns) in text. However, due to time constraints and scoping, this use case was not explored further. I believe that this use case can yield good results, given that puns, similar to sarcasm (Pan et al., 2020), are heavily dependent on incongruity, which is related to oppositeness. As per Valitutti, Strapparava and Stock (2008), the *incongruity resolution model* claims that a punning text induces two different interpretations, where one is incoherent and the other is coherent. Thus, by this model, the coherent interpretation is expected to be funny. The *varied word* in the pun is what causes the incongruity in the first interpretation. At the same time, it is also the *trigger of coherence* which does the restoration in the second interpretation. Delabastita (2016) observed that the features of a pun are mutually independent, as such a pun can be *homographic*, *homophonic*, *both*, or *neither*. But in all types of puns, the common thread is the fact that there needs to be some form of incongruity. That incongruity is where the oppositeness measure can be used for detection. For example, in the case of *homographic* puns, incongruity exists with different linguistic morphs of two roots ending up with the same string of graphemes. In such a situation, measuring the contextual oppositeness between the two lemmas (or other similar standardized forms) would give a score to the shift in sense between the two competing interpretations. Similarly, semantic oppositeness can be used to detect incongruity triggers in other types of puns, as well.

APPENDIX

SUPPLEMENTARY FIGURES AND EXAMPLES

1. Intervirology. 2016 Nov 23;59(2):111-117. [Epub ahead of print]

Nonstructural Protein 1 of Tick-Borne Encephalitis Virus Induces Oxidative Stress and Activates Antioxidant Defense by the Nrf2/ARE Pathway.

Kuzmenko YV(1), Smirnova OA, Ivanov AV, Starodubova ES, Karpov VL.

Author information:
(1)Engelhardt Institute of Molecular Biology, Russian Academy of Sciences, Moscow, Russia.

BACKGROUND: Infection with tick-borne encephalitis virus (TBEV) causes pathological changes in the central nervous system. However, the possible redox alterations in the infected cells that can contribute to the virus pathogenicity remain unknown.
OBJECTIVE: In the current study we explored the ability of TBEV nonstructural protein 1 (NS1) to induce oxidative stress and activate antioxidant defense via the nuclear factor (erythroid-derived-2)-like 2/antioxidant response element (Nrf2/ARE) pathway.
METHODS: HEK 293T cells were transfected with plasmid encoding NS1 protein, and the production of reactive oxygen species (ROS) was measured using oxidation-sensitive dyes, the activation of the ARE promoter was estimated using a reporter plasmid, and the expression of phase II detoxifying enzymes was quantified by measuring their mRNA levels using RT-qPCR.
RESULTS: A high level of ROS production was detected in cells transfected with NS1-expressing plasmid. In addition, this protein activated the promoter with an ARE and upregulated the transcription of ARE-dependent genes that encode phase II enzymes.
CONCLUSION: TBEV NS1 protein both triggers ROS production and activates a defense Nrf2/ARE pathway. These data suggest that a role of redox-mediated processes in TBEV-induced damage of the central nervous system should also be explored. These data can contribute to a better understanding of TBEV pathogenicity, further improvement of TBE treatment, and the development of vaccine candidates against this infection.

DOI: 10.1159/000452160
PMID: 27875810 [PubMed - as supplied by publisher]

(a) Sample abstract 1

1. Sci Rep. 2016 Nov 22;6:37370. doi: 10.1038/srep37370.

Effects of light-emitting diode irradiation on the osteogenesis of human umbilical cord mesenchymal stem cells in vitro.

Yang D(1), Yi W(1), Wang E(1), Wang M(1).

Author information:
(1)Department of Orthopaedics, Nanshan Hospital, Guangdong MedicalCollege, Shenzhen Guangdong, 518052, China.

The aim of this study was to examine the effects of light-emitting diode (LED) photobiomodulation therapy on the proliferation and differentiation of human umbilical cord mesenchymal stem cells (hUMSCs) cultured in osteogenic differentiation medium. HUMSCs were irradiated with an LED light at 620 nm and 2 J/cm(2) and monitored for cell proliferation and osteogenic differentiation activity. The experiment involved four groups of cells: the control group; the osteogenic group (osteo group); the LED group; the osteogenic + LED group (LED + osteo group). HUMSC proliferation was detected by performing a3-(4,5-dimethylthiazol-2yl)-2,5 diphenyltetrazolium bromide(MTT) assay. Osteogenic activity was evaluated by performing alkaline phosphatase (ALP) and Von Kossa staining, and osteopontin (OPN) gene mRNA expression was evaluated byreverse transcription polymerase chain reaction (RT-PCR). The hUMSCs in the LED + osteo group exhibited a significantly higher proliferation rate than the other subgroups. Additionally, there were greater numbers of ALP-positive cells and Von Kossa nodules in the LED + osteo group. OPN mRNA expression in the LED + osteo group was higher than other subgroups. In conclusion, low levels of LED light at a wavelength of 620 nm enhance the proliferation and osteogenic differentiation of hUMSCs during a long culture period.

DOI: 10.1038/srep37370
PMID: 27874039 [PubMed - in process]

(b) Sample abstract 2

Figure A.34. Sample PubMed text abstracts

Example A.1 Lemmatization example

```
<token id="4">
    <word>found</word>
    <lemma>find</lemma>
    <CharacterOffsetBegin>504</CharacterOffsetBegin>
    <CharacterOffsetEnd>509</CharacterOffsetEnd>
    <POS>VBD</POS>
    <NER>O</NER>
    <Speaker>PER0</Speaker>
</token>
<token id="7">
    <word>was</word>
    <lemma>be</lemma>
    <CharacterOffsetBegin>522</CharacterOffsetBegin>
    <CharacterOffsetEnd>525</CharacterOffsetEnd>
    <POS>VBD</POS>
    <NER>O</NER>
    <Speaker>PER0</Speaker>
</token>
<token id="12">
    <word>but</word>
    <lemma>but</lemma>
    <CharacterOffsetBegin>560</CharacterOffsetBegin>
    <CharacterOffsetEnd>563</CharacterOffsetEnd>
    <POS>CC</POS>
    <NER>O</NER>
    <Speaker>PER0</Speaker>
</token>
```

Example A.2 Parse tree example

```
(ROOT
(S
(ADVP (RB Nevertheless))
(, ,)
(NP (PRP we))
(VP (VBD found)
(SBAR (IN that)
(S
(NP (NN miR-31))
(VP (VBD was)
(ADVP (RB particularly))
(VP (VBN up-regulated)
(PP
(PP (IN in)
(NP (NNS HSCs)))
(CONJP (CC but)
(RB not))
(PP (IN in)
(NP (NNS hepatocytes))))
(PP (IN during)
(NP (NN fibrogenesis)))))))))
(. .)))
```

Example A.3 High level data format

```
<entity 1>;<entity 2>;<Relationship Data 1>;<Relationship Data 2>;...;<
    Relationship Data N>
```

Example A.4 Relationship data format

```
<Relationship String>:<Value>:<Matches List>:<Close Matches List>
```

Example A.5 List data format

```
<ID 1>,<Sentence Number 1>.<ID 2>,<Sentence Number 2>.....<ID M>,<
    Sentence Number M>
```

Example A.6 Example data line

```
MIR320A;Cell Proliferation;
Moreover inhibits:1.7151437:25728840,6:22459450,6:;
inhibit:1.7057567:22459450,6:25728840,6:
```

Example A.7 Final result file example

```
**********************************************************
**********************************************************

(MIR320A;Moreover inhibits;Cell Proliferation)

Matches
Moreover, miR-320a expression inhibits human-derived endothelium cell proliferation
    and induces apoptosis.

Close Matches
And we demonstrated that miR-320a restoration inhibited colon cancer cell
    proliferation and β-catenin, a functionally oncogenic molecule was a direct
    target gene of miR-320a.

**********************************************************
```

```
(MIR320A;inhibit;Cell Proliferation)

Matches
And we demonstrated that miR-320a restoration inhibited colon cancer cell
    proliferation and β-catenin, a functionally oncogenic molecule was a direct
    target gene of miR-320a.

Close Matches
Moreover, miR-320a expression inhibits human-derived endothelium cell proliferation
    and induces apoptosis.

**********
**********
```

REFERENCES CITED

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Isard, M. et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation* (Vol. 16, pp. 265–283).

Alsheikh, M. A., Niyato, D., Lin, S., Tan, H.-P. & Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE Network*, *30*(3), 22–29.

Apresjan, J. D. (1974). Regular polysemy. *Linguistics*, *12*(142), 5–32.

Arvidsson, F. & Flycht-Eriksson, A. (2016). Ontologies I. Retrieved from http://tomgruber.org/writing/ontolingua-kaj-1993.pdf

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D. et al. (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics*, *25*(1), 25–29.

Bednarek, A. & Grochowski, M. (1993). *Zadania z semantyki jezykoznawczej*. Uniwersytet Mikołaja Kopernika.

Bernstein, M., Monroy-Hernández, A., Harry, D., André, P., Panovich, K. & Vargas, G. (2011). 4chan and/b: An Analysis of Anonymity and Ephemerality in a Large Online Community.

Blitzer, J., Dredze, M. & Pereira, F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th annual meeting of the Association of Computational Linguistics* (pp. 440–447).

Bollegala, D. & Bao, C. (2018). Learning Word Meta-Embeddings by Autoencoding. In *Proceedings of the 27th International Conference on Computational Linguistics* (pp. 1650–1661).

Brent, E., Atkisson, C. & Green, N. (2010). Time-shifted Online Collaboration: Creating Teachable Moments Through Automated Grading. In *Proceedings of Monitoring and Assessment in Online Collaborative Environments: Emergent Computational Technologies for E-learning Support* (pp. 55–73). IGI Global.

Bruckschen, M., Northfleet, C., Silva, D., Bridi, P., Granada, R., Vieira, R., ... Sander, T. (2010). Named entity recognition in the legal domain for ontology population. In *Proceedings of the 3rd Workshop on Semantic Processing of Legal Texts (SPLeT 2010)* (pp. 16–21).

Cai, Y., Cai, H. & Wan, X. (2019). Multi-Modal Sarcasm Detection in Twitter with Hierarchical Fusion Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 2506–2515).

Cao, J., Guo, J., Li, X., Jin, Z., Guo, H. & Li, J. (2018). Automatic Rumor Detection on Microblogs: A Survey. *arXiv preprint arXiv:1807.03505*.

Castillo, C., Mendoza, M. & Poblete, B. (2011). Information Credibility on Twitter. In *International World Wide Web Conference* (pp. 675–684). ACM. doi:10.1145/1963405.1963500

Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R. & Poria, S. (2019). Towards Multimodal Sarcasm Detection (An _Obviously_ Perfect Paper). *arXiv preprint arXiv:1906.01815*.

Chandar, S., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C. & Saha, A. (2014). An Autoencoder Approach to Learning Bilingual Word Representations. In *Advances in Neural Information Processing Systems* (pp. 1853–1861).

Charles, W. G. & Miller, G. A. (1989). Contexts of Antonymous Adjectives. *Applied Psycholinguistics*, *10*(3), 357–375.

Chen, C., Wang, Y., Yang, S., Li, H., Zhao, G., Wang, F., ... Wang, D. W. (2015). MiR-320a contributes to atherogenesis by augmenting multiple risk factors and down-regulating SRF. *Journal of Cellular and Molecular Medicine*, *19*(5), 970–985.

Clark, E. V. (1972). On the child's acquisition of antonyms in two semantic fields. *Journal of Verbal Learning and Verbal Behavior*, *11*(6), 750–758.

Clark, H. H. (1970a). The primitive nature of children's relational concepts. *Cognition and the Development of Language*, 269–278.

Clark, H. H. (1970b). Word Associations and Linguistic Theory. *New Horizons in Linguistics*, *1*, 271–286.

Cruse, A. (2011). *Meaning in Language: An introduction to semantics and pragmatics*. Oxford University Press.

Cruse, D. A. (1994). Prototype Theory and Lexical Relations. *Rivista di Linguistica*, *6*(2), 167–188.

Cruse, D. A. (1986). *Lexical Semantics*. Cambridge University Press.

Dai, A. M., Olah, C. & Le, Q. V. (2015). Document Embedding with Paragraph Vectors. *arXiv preprint arXiv:1507.07998*.

Damien, A. (2017). Auto-Encoder Example. Retrieved November 6, 2020, from https://goo.gl/wiBspX

Das, R., Zaheer, M. & Dyer, C. (2015). Gaussian LDA for Topic Models with Word Embeddings. In *ACL* (pp. 795–804).

Davies, M. (2012). A New Approach to Oppositions in Discourse: The Role of Syntactic Frames in the Triggering of Noncanonical Oppositions. *Journal of English Linguistics*, *40*(1), 41–73.

de Silva, N. (2017a). Sensing Puns is its Own Reword: Automatic Detection of Paronomasia. doi:10.13140/RG.2.2.12478.31046

de Silva, N. H. N. D. (2015a). SAFS3 Algorithm: Frequency Statistic and Semantic Similarity Based Semantic Classification Use Case. In *Proceedings of Advances in ICT for Emerging Regions (ICTer), 2015 Fifteenth International Conference on* (pp. 77–83). IEEE.

de Silva, N. H. N. D. (2017b). Relational Databases and Biomedical Big Data. *Bioinformatics in MicroRNA Research*, 69–81.

de Silva, N. H. N. D., Perera, A. S. & Maldeniya, M. K. D. T. (2013). Semi-Supervised Algorithm for Concept Ontology Based Word Set Expansion. In *Proceedings of Advances in ICT for Emerging Regions (ICTer), 2013 International Conference on* (pp. 125–131). IEEE.

de Silva, N. & Dou, D. (2019). Semantic Oppositeness Embedding Using an Autoencoder-based Learning Model. In *International Conference on Database and Expert Systems Applications* (pp. 159–174). Springer. doi:10.1007/978-3-030-27615-7_12

de Silva, N., Dou, D. & Huang, J. (2017). Discovering Inconsistencies in PubMed Abstracts through Ontology-Based Information Extraction. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 362–371). ACM.

de Silva, N. (2015b). Sinhala Text Classification: Observations from the Perspective of a Resource Poor Language. *ResearchGate*. doi:10.13140/RG.2.2.17380.63369/2

de Silva, N. (2019). Survey on Publicly Available Sinhala Natural Language Processing Tools and Research. *arXiv preprint arXiv:1906.02358*.

Delabastita, D. (2016). *Traductio. Essays on Punning and Translation*. Routledge.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Domm, P. (2013). False Rumor of Explosion at White House Causes Stocks to Briefly Plunge; AP Confirms Its Twitter Feed Was Hacked. Retrieved from https://www.cnbc.com/id/100646197

Eilbeck, K., Lewis, S. E., Mungall, C. J., Yandell, M. et al. (2005). The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biology*, *6*(5), R44.

Etzioni, O., Banko, M., Soderland, S. & Weld, D. S. (2008). Open information extraction from the web. *Communications of the ACM*, *51*(12), 68–74.

Etzioni, O., Fader, A., Christensen, J., Soderland, S. & Mausam, M. (2011). Open Information Extraction: The Second Generation. In *International Joint Conference on Artificial Intelligence (IJCAI)* (Vol. 11, pp. 3–10).

Exiqon. (2016). What are micrornas? Retrieved from http://www.exiqon.com/what-are-microRNAs

Fader, A., Soderland, S. & Etzioni, O. (2011). Identifying Relations for Open Information Extraction. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1535–1545). Association for Computational Linguistics. Edinburgh, United Kingdom.

Foltz, P. W., Laham, D. & Landauer, T. K. (1999). Automated Essay Scoring: Applications to Educational Technology. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications* (Vol. 1, pp. 939–944). Association for the Advancement of Computing in Education (AACE).

Franzke, M. & Streeter, L. A. (2006). Building Student Summarization, Writing and Reading Comprehension Skills With Guided Practice and Automated Feedback. *Highlights From Research at the University of Colorado, A white paper from Pearson Knowledge Technologies.*

Fung, G. P. C., Yu, J. X., Lu, H. & Yu, P. S. (2006). Text Classification without Negative Examples Revisit. *IEEE Transactions on Knowledge and Data Engineering*, *18*(1), 6–20.

Gamage, V., Warushavithana, M., de Silva, N., Perera, A. S., Ratnayaka, G. & Rupasinghe, T. (2018). Fast Approach to Build an Automatic Sentiment Annotator for Legal Domain using Transfer Learning. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 260–265).

Gomaa, W. H. & Fahmy, A. A. (2013). A Survey of Text Similarity Approaches. *International Journal of Computer Applications*, *68*(13).

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Grochowski, M. (1982). *Zarys leksykologii i leksykografii: Zagadnienia synchroniczne*. Toruń: Wydawnictwo UMK.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*(2), 199–220.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, *43*(5-6), 907–928.

Gupta, M., Zhao, P. & Han, J. (2012). Evaluating Event Credibility on Twitter. In *Proceedings of the 2012 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics* (pp. 153–164).

Gutierrez, F., Dou, D., Fickas, S., Wimalasuriya, D. & Zong, H. (2016). A hybrid ontology-based information extraction system. *Journal of Information Science*, *42*(6), 798–820.

Gutierrez, F., Dou, D., de Silva, N. & Fickas, S. (2017). Online Reasoning for Semantic Error Detection in Text. *Journal on Data Semantics*. doi:10.1007/s13740-017-0079-6

Hermann, D. J., Conti, G., Peters, D., Robbins, P. H. & Chaffin, R. J. (1979). Comprehension of Antonymy and the Generality of Categorization Models. *Journal of Experimental Psychology: Human Learning and Memory*, *5*(6), 585.

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507.

Hobbs, J. R., Appelt, D., Bear, J., Israel, D. et al. (1993). FASTUS: A System for Extracting Information from Text. In *Proceedings of the workshop on Human Language Technology* (pp. 133–137). Association for Computational Linguistics.

Hu, J., Chen, C., Liu, Q., Liu, B., Song, C., Zhu, S., ... Yao, D. et al. (2015). The role of the miR-31/FIH1 pathway in TGF-β-induced liver fibrosis. *Clinical Science*, *129*(4), 305–317.

Huang, J., Dang, J., Borchert, G. M., Eilbeck, K. et al. (2014). OMIT: Dynamic, Semi-Automated Ontology Development for the microRNA Domain. *PLOS ONE*, *9*(7), 1–16.

Huang, J., Eilbeck, K., Smith, B., Blake, J. A., Dou, D., Huang, W., ... de Silva, N. et al. (2016a). The development of non-coding RNA ontology. *International Journal of Data Mining and Bioinformatics*, *15*(3), 214–232.

Huang, J., Gutierrez, F., Strachan, H. J., Dou, D., Huang, W., Smith, B., ... de Silva, N. et al. (2016b). OmniSearch: a semantic search system based on the Ontology for MIcroRNA Target (OMIT) for microRNA-target gene interaction data. *Journal of Biomedical Semantics*, *7*(1), 1.

Jayawardana, V., Lakmal, D., de Silva, N., Perera, A. S., Sugathadasa, K., Ayesha, B. & Perera, M. (2017a). Word Vector Embeddings and Domain Specific Semantic based Semi-Supervised Ontology Instance Population. *International Journal on Advances in ICT for Emerging Regions*, *10*(1), 1.

Jayawardana, V., Lakmal, D., de Silva, N., Perera, A. S., Sugathadasa, K. & Ayesha, B. (2017b). Deriving a Representative Vector for Ontology Classes with Instance Word Vector Embeddings. In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)* (pp. 79–84). IEEE.

Jayawardana, V., Lakmal, D., de Silva, N., Perera, A. S., Sugathadasa, K., Ayesha, B. & Perera, M. (2017c). Semi-Supervised Instance Population of an Ontology using Word Vector Embedding. In *Advances in ICT for Emerging Regions (ICTer), 2017 Seventeenth International Conference on* (pp. 1–7). IEEE.

Jiang, J. J. & Conrath, D. W. (1997). Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *10th International Conference on Research in Computational Linguistics, ROCLING'97*, 19–33.

Jin, Z., Cao, J., Jiang, Y. & Zhang, Y. (2014). News Credibility Evaluation on Microblog with a Hierarchical Propagation Model. In *International Conference on Data Mining* (pp. 230–239). doi:10.1109/ICDM.2014.91

Jin, Z., Cao, J., Guo, H., Zhang, Y., Wang, Y. & Luo, J. (2017). Detection and Analysis of 2016 US Presidential Election Related Rumors on Twitter. In *International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation* (pp. 14–24).

Jones, S. (2003). *Antonymy: A Corpus-Based Perspective*. Routledge.

Jones, S., Murphy, M. L., Paradis, C. & Willners, C. (2012). *Antonyms in English: Construals, Constructions and Canonicity*. Cambridge University Press.

Justeson, J. S. & Katz, S. M. (1991). Co-occurrences of Antonymous Adjectives and Their Contexts. *Computational Linguistics*, *17*(1), 1–20.

Kawamura, T., Watanabe, K., Matsumoto, N., Egami, S. & Jibu, M. (2018). Funding map using paragraph embedding based on semantic diversity. *Scientometrics*, *116*(2), 941–958.

Kempson, R. M. (1977). *Semantic Theory*. Cambridge University Press.

Kostić, N. (2015). Antonym sequence in written discourse: A corpus-based study. *Language Sciences*, *47*, 18–31.

Kulick, S., Bies, A., Liberman, M., Mandel, M. et al. (2004). Integrated Annotation for Biomedical Information Extraction. In *Proceedings of the Human Language Technology Conference and the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)* (pp. 61–68).

Kuzmenko, Y. V., Smirnova, O. A., Ivanov, A. V., Starodubova, E. S. & Karpov, V. L. (2016). Nonstructural protein 1 of tick-borne encephalitis virus induces oxidative stress and activates antioxidant defense by the Nrf2/ARE pathway. *Intervirology*, *59*(2), 111–117.

Kwon, S., Cha, M., Jung, K., Chen, W. & Wang, Y. (2013). Prominent Features of Rumor Propagation in Online Social Media. In *13th International Conference on Data Mining* (pp. 1103–1108). IEEE.

Lau, J. H. & Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *arXiv preprint arXiv:1607.05368*.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Leech, G. (1990). The study of Meaning. *L.: Penguin Books*.

Lehrer, A. & Lehrer, K. (1982). Antonymy. *Linguistics and Philosophy*, *5*(4), 483–501.

Leskovec, J., Rajaraman, A. & Ullman, J. D. (2014). *Mining of Massive Datasets*. Cambridge University Press.

Letia, I. A. & Cornoiu, S. (2010). An Ontological Approach to Legal Literature for Improving Legal Knowledge Dissemination. *Development and Application Systems*, 90.

Levy, O., Dagan, I. & Goldberger, J. (2014). Focused Entailment Graphs for Open IE Propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning* (pp. 87–97).

Li, Q., Wang, X., Zhang, Y., Ling, F., Wu, C. H. & Han, J. (2018). Pattern Discovery for Wide-Window Open Information Extraction in Biomedical Literature. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 420–427). IEEE.

Li, Y., Bandar, Z. A. & McLean, D. (2003). An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. *IEEE Transactions on Knowledge and Data Engineering*, *15*(4), 871–882.

Lin, C. Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL-04 Workshop on Text Summarization Branches Out* (Vol. 8). Barcelona, Spain.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B. & Bengio, Y. (2017). A Structured Self-attentive Sentence Embedding. *arXiv preprint arXiv:1703.03130*.

Linderman, G. C. & Steinerberger, S. (2019). Clustering with t-SNE, Provably. *SIAM Journal on Mathematics of Data Science*, *1*(2), 313–332.

Liu, B., Tian, Y., Li, F., Zhao, Z., Jiang, X., Zhai, C., ... Zhang, L. (2016). Tumor-suppressing roles of miR-214 and miR-218 in breast cancer. *Oncology Reports*, *35*(6), 3178–3184.

Lobanova, G. V. (2012). *The Anatomy of Antonymy: a Corpus-driven Approach*. University of Groningen The Netherlands.

Lokanathan, S., Kreindler, G. E., de Silva, N. H. N., Miyauchi, Y., Dhananjaya, D. & Samarajiva, R. (2016). The Potential of Mobile Network Big Data as a Tool in Colombo's Transportation and Urban Planning. *Information Technologies & International Development*, *12*(2), pp–63.

Lord, P. W., Stevens, R. D., Brass, A. & Goble, C. A. (2002). Semantic similarity measures as tools for exploring the Gene Ontology. In *Biocomputing 2003* (pp. 601–612). World Scientific.

Lv, Y., Duan, Y., Kang, W., Li, Z. & Wang, F.-Y. (2015). Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 865–873.

Lyons, J. (1987). *Semantics. 1*. Cambridge University Press.

Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B. J., Wong, K.-F. & Cha, M. (2016). Detecting Rumors from Microblogs with Recurrent Neural Networks. In *25th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 3818–3824). New York, New York, USA.

Ma, J., Gao, W., Wei, Z., Lu, Y. & Wong, K.-F. (2015). Detect Rumors Using Time Series of Social Context Information on Microblogging Websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management* (pp. 1751–1754). doi:10.1145/2806416.2806607

Ma, J., Gao, W. & Wong, K.-F. (2017). Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning. In *Association for Computational Linguistics* (pp. 708–717). doi:10.18653/v1/P17-1066

Ma, J., Gao, W. & Wong, K.-F. (2018a). Detect Rumor and Stance Jointly by Neural Multi-task Learning. In *Companion Proceedings of The Web Conference* (pp. 585–593). doi:10.1145/3184558.3188729

Ma, J., Gao, W. & Wong, K.-F. (2018b). Rumor Detection on Twitter with Tree-structured Recursive Neural Networks. In *Association for computational linguistics* (pp. 1980–1989).

Maaten, L. v. d. & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. et al. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of Association for Computational Linguistics (ACL) System Demonstrations* (pp. 55–60).

Mapa, E., Wattaladeniya, L., Chathuranga, C., Dassanayake, S., de Silva, N., Kohomban, U. & Maldeniya, D. (2012). Text Normalization in Social Media by using Spell Correction and Dictionary Based Approach. *Systems Learning*, *1*, 1–6.

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, ... Michael Isard et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Software available from tensorflow.org.

Mausam, Schmitz, M., Soderland, S., Bart, R. & Etzioni, O. (2012). Open Language Learning for Information Extraction. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)* (pp. 523–534).

Maynard, D., Yankova, M., Kourakis, A. & Kokossis, A. (2005). Ontology-based information extraction for market monitoring and technology watch. In *ESWC Workshop" End User Apects of the Semantic Web*.

Mettinger, A. (1994). *Aspects of semantic opposition in English*. Oxford University Press.

Mikołajczak-Matyja, N. (2018). The Prototypicality of Semantic Opposition in the Light of Linguistic Studies and Psycholinguistic Experiments. *Studies in Polish Linguistics*, *13*(1), 1–23.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013a). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 3111–3119.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013b). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. J. (1990). Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, *3*(4), 235–244.

Mitchell, T., Russell, T., Broomhead, P. & Aldridge, N. (2002). Towards Robust Computerised Marking of Free-Text Responses, Loughborough University.

Al-Mubaid, H. & Umair, S. A. (2006). A New Text Categorization Technique Using Distributional Clustering and Learning Logic. *IEEE Transactions on Knowledge and Data Engineering*, *18*(9), 1156–1165.

Mudalige, C. R., Karunarathna, D., Rajapaksha, I., de Silva, N., Ratnayaka, G., Perera, A. S. & Pathirana, R. (2020). Sigmalaw-absa: Dataset for aspect-based sentiment analysis in legal opinion texts. *arXiv preprint arXiv:2011.06326*.

Muehleisen, V. & Isono, M. (2009). Antonymous adjectives in Japanese discourse. *Journal of Pragmatics*, *41*(11), 2185–2203.

Murphy, M. L. (2003). *Semantic Relations and the Lexicon: Antonymy, Synonymy and Other Paradigms*. Cambridge University Press.

Murphy, M. L., Jones, S. & Koskela, A. (2015). Signals of Contrastiveness: But, Oppositeness, and Formal Similarity in Parallel Contexts. *Journal of English Linguistics*, *43*(3), 227–249.

Nagle, A. (2017). *Kill All Normies: Online Culture Wars from 4chan and Tumblr to Trump and the Alt-Right*. John Hunt Publishing.

Natale, D. A., Arighi, C. N., Barker, W. C., Blake, J. A. et al. (2011). The Protein Ontology: a structured representation of protein forms and complexes. *Nucleic Acids Research*, *39*(suppl 1), D539–D545.

National Center for Biotechnology Information. (2020). PubMed Help. Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK3827/

Owens Jr, R. E. (2015). *Language Development: An Introduction*. Pearson.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., ... Ward, R. (2016). Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(4), 694–707.

Pan, H., Lin, Z., Fu, P., Qi, Y. & Wang, W. (2020). Modeling Intra and Inter-modality Incongruity for Multi-Modal Sarcasm Detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings* (pp. 1383–1392).

Pan, S. J. & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.

Paradis, M., Goldblum, M.-C. & Abidi, R. (1982). Alternate Antagonism with Paradoxical Translation Behavior in Two Bilingual Aphasic Patients. *Brain and Language*, *15*(1), 55–69.

Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (Vol. 14, pp. 1532–1543). doi:10.3115/v1/D14-1162

Phillips, C. I. & Pexman, P. M. (2015). When Do Children Understand "Opposite"? *Journal of Speech, Language, and Hearing Research*, *58*(4), 1233–1244.

Pisanelli, D. M., Gangemi, A. & Steve, G. (1999). A Medical Ontology Library that Integrates the UMLS Metathesaurus™. In *Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making* (pp. 239–248). Springer.

Princeton University. (2020). WordNet – A Lexical Database for English. Online: `http://wordnet.princeton.edu/`. Accessed: 2020-10-31.

Rajapaksha, I., Mudalige, C. R., Karunarathna, D., de Silva, N., Rathnayaka, G. & Perera, A. S. (2020). Rule-based approach for party-based sentiment analysis in legal opinion texts. *arXiv preprint arXiv:2011.05675*.

Ratnayaka, G., Rupasinghe, T., de Silva, N., Warushavithana, M., Gamage, V., Perera, M. & Perera, A. S. (2019a). Classifying Sentences in Court Case Transcripts using Discourse and Argumentative Properties. *ICTer*, *12*(1).

Ratnayaka, G., de Silva, N., Perera, A. S. & Pathirana, R. (2020). Effective approach to develop a sentiment annotator for legal domain in a low resource setting. *arXiv preprint arXiv:2011.00318*.

Ratnayaka, G., Rupasinghe, T., de Silva, N., Gamage, V. S., Warushavithana, M. & Perera, A. S. (2019b). Shift-of-Perspective Identification Within Legal Cases. In *Proceedings of the 3rd Workshop on Automated Detection, Extraction and Analysis of Semantic Information in Legal Texts*.

Ratnayaka, G., Rupasinghe, T., de Silva, N., Warushavithana, M., Gamage, V. & Perera, A. S. (2018). Identifying Relationships Among Sentences in Court Case Transcripts Using Discourse Relations. In *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)* (pp. 13–20).

Rothman, L. & Parker, M. (2009). Just-About-Right (JAR) Scales. *West Conshohocken, PA: ASTM International*.

Sarinnapakorn, K. & Kubat, M. (2007). Combining Subclassifiers in Text Categorization: A DST-Based Solution and a Case Study. *IEEE Transactions on Knowledge and Data Engineering*, *19*(12), 1638–1651.

Schimmack, U. (2001). Pleasure, displeasure, and mixed feelings: Are semantic opposites mutually exclusive? *Cognition & Emotion*, *15*(1), 81–97.

Shakespeare, W. (1955). *Mr. william shakespeares comedies, histories, and tragedies: A facsimile of the first folio*. Yale University Press.

Shima, H. (2016). WordNet Similarity for Java (WS4J). Retrieved from https://code.google.com/p/ws4j/

Shin, J. (2008). Morality and Internet Behavior: A study of the Internet Troll and its relation with morality on the Internet. In *Society for Information Technology & Teacher Education International Conference* (pp. 2834–2840). Association for the Advancement of Computing in Education (AACE).

Stavrianou, A., Andritsos, P. & Nicoloyannis, N. (2007). Overview and Semantic Issues of Text Mining. *Association for Computing Machinery Special Interest Group on Management of Data Record*, *36*(3), 23–34.

Sugathadasa, K., Ayesha, B., de Silva, N., Perera, A. S., Jayawardana, V., Lakmal, D. & Perera, M. (2017). Synergistic Union of Word2Vec and Lexicon for Domain Specific Semantic Similarity. *IEEE International Conference on Industrial and Information Systems (ICIIS)*, 1–6.

Sugathadasa, K., Ayesha, B., de Silva, N., Perera, A. S., Jayawardana, V., Lakmal, D. & Perera, M. (2018). Legal Document Retrieval using Document Vector Embeddings and Deep Learning. In *Science and Information Conference* (pp. 160–175). Springer.

Sun, J.-Y., Huang, Y., Li, J.-P., Zhang, X., Wang, L., Meng, Y.-L., ... Wang, W.-Z. et al. (2012). MicroRNA-320a suppresses human colon cancer cell proliferation by directly targeting β-catenin. *Biochemical and Biophysical Research Communications*, *420*(4), 787–792.

Tawfik, N. S. & Spruit, M. (2019a). UU_TAILS at MEDIQA 2019: Learning Textual Entailment in the Medical Domain. In *Proceedings of the 18th BioNLP Workshop and Shared Task* (pp. 493–499).

Tawfik, N. S. & Spruit, M. R. (2018). Automated Contradiction Detection in Biomedical Literature. In *International Conference on Machine Learning and Data Mining in Pattern Recognition* (pp. 138–148). Springer.

Tawfik, N. S. & Spruit, M. R. (2019b). Towards Recognition of Textual Entailment in the Biomedical Domain. In *International Conference on Applications of Natural Language to Information Systems* (pp. 368–375). Springer.

Teng, F., Bai, M. & Li, T. (2019). Automatic Labeling for Gene-Disease Associations through Distant Supervision. In *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* (pp. 491–497). IEEE.

Tuggy, D. (1993). Ambiguity, polysemy, and vagueness. *Cognitive Linguistics*, *4*(3), 273–290.

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *European Conference on Machine Learning* (pp. 491–502). Springer.

U.S. National Library of Medicine. (2020). Medical subject headings. Retrieved from https://www.nlm.nih.gov/mesh/

Upeksha, D., Wijayarathna, C., Siriwardena, M., Lasandun, L., Wimalasuriya, C., de Silva, N. H. N. D. & Dias, G. (2015). Comparison Between Performance of Various Database Systems for Implementing a Language Corpus. In *International Conference: Beyond Databases, Architectures and Structures* (pp. 82–91). Springer.

Valitutti, A., Strapparava, C. & Stock, O. (2008). Textual Affect Sensing for Computational Advertising. In *Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium: Creative Intelligent Systems* (pp. 117–122).

van de Weijer, J., Paradis, C., Willners, C. & Lindgren, M. (2014). Antonym canonicity: Temporal and contextual manipulations. *Brain and Language*, *128*(1), 1–8.

Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A. & Ciravegna, F. (2002). MnM: Ontology driven semi-automatic and automatic support for semantic markup. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, 213–221.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention Is All You Need. In *Advances in neural information processing systems* (pp. 5998–6008).

Veyseh, A. P. B., Thai, M. T., Nguyen, T. H. & Dou, D. (2019). Rumor Detection in Social Networks via Deep Contextual Modeling. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 113–120).

Wang, B., Zhang, J., Ding, F. & Zou, Y. (2017). Multi-Document News Summarization via Paragraph Embedding and Density Peak Clustering. In *2017 International Conference on Asian Language Processing (IALP)* (pp. 260–263). IEEE.

Wang, F., Lv, P., Liu, X., Zhu, M. & Qiu, X. (2015). microRNA-214 enhances the invasion ability of breast cancer cells by targeting p53. *International Journal of Molecular Medicine*, *35*(5), 1395–1402.

Wang, P., Ji, L., Yan, J., Dou, D., de Silva, N., Zhang, Y. & Jin, L. (2018a). Concept and Attention-Based CNN for Question Retrieval in Multi-View Learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *9*(4), 41.

Wang, P., Dou, D., Wu, F., de Silva, N. & Jin, L. (2019). Logic rules powered knowledge graph embedding. *arXiv preprint arXiv:1903.03772*.

Wang, X., Zhang, Y., Li, Q., Chen, Y. & Han, J. (2018b). Open Information Extraction with Meta-pattern Discovery in Biomedical Literature. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics* (pp. 291–300).

Wijeratne, Y. & de Silva, N. (2020). Sinhala language corpora and stopwords from a decade of sri lankan facebook. *arXiv preprint arXiv:2007.07884*. doi:10.2139/ssrn.3650976

Wijeratne, Y., de Silva, N. & Shanmugarajah, Y. (2019). Natural Language Processing for Government: Problems and Potential. *LIRNEasia*. doi:10.13140/RG.2.2.34297.31845

Wijesiri, I., Gallage, M., Gunathilaka, B., Lakjeewa, M., Wimalasuriya, D., Dias, G., ... de Silva, N. (2014). Building a WordNet for Sinhala. In *Proceedings of the seventh global wordnet conference* (pp. 100–108).

Willners, C. (2001). *Antonyms in Context : A Corpus-Based Semantic Analysis of Swedish Descriptive Adjectives*. Lund University.

Wimalasuriya, D. C. & Dou, D. (2010). Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, *36*(3), 306–323.

World Health Organization. (2020). NCD mortality and morbidity. Retrieved from http://www.who.int/gho/ncd/mortality_morbidity/en/

Wu, F. & Weld, D. S. (2010). Open Information Extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 118–127). Association for Computational Linguistics.

Wu, K., Yang, S. & Zhu, K. Q. (2015). False Rumors Detection on Sina Weibo by Propagation Structures. In *IEEE 31st International Conference on Data Engineering* (pp. 651–662).

Wu, Y., Zhao, S. & Li, W. (2020). Phrase2vec: Phrase embedding based on parsing. *Information Sciences*, *517*, 100–127.

Wu, Z. & Palmer, M. (1994). Verbs Semantics and Lexical Selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics* (pp. 133–138). ACL '94. Las Cruces, New Mexico: Association for Computational Linguistics.

Wyner, A. Z. (2010). Towards Annotating and Extracting Textual Legal Case Elements. *Informatica e Diritto: Special Issue on Legal Ontologies and Artificial Intelligent Techniques*, *19*(1-2), 9–18.

Yang, D., Yi, W., Wang, E. & Wang, M. (2016). Effects of light-emitting diode irradiation on the osteogenesis of human umbilical cord mesenchymal stem cells in vitro. *Scientific Reports*, *6*(1), 1–7.

Yang, F., Liu, Y., Yu, X. & Yang, M. (2012). Automatic Detection of Rumor on Sina Weibo. In *Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining Workshop on Mining Data Semantics* (pp. 1–7). doi:10.1145/2350190.2350203

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems* (pp. 5753–5763).

Ye, Z. (2014). Opposites in Language and Thought: A Chinese Perspective. In *Selected Papers from the 4th UK Cognitive Linguistics Conference* (pp. 320–338).

Yin, W. & Schütze, H. (2016). Discriminative Phrase Embedding for Paraphrase Identification. *arXiv preprint arXiv:1604.00503*.

Zhao, Z., Resnick, P. & Mei, Q. (2015). Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 1395–1405). doi:10.1145/2736277.2741637