Learning High-Precision Lexical Inferences

Vered Shwartz

Department of Computer Science
Ph.D. Thesis

Submitted to the Senate of Bar-Ilan University

Ramat Gan, Israel

April 2019
This work was carried out under the supervision of Prof. Ido Dagan, Department of Computer Science, Bar-Ilan University.
To my dearest parents, Haim and Tikva, and to my beloved husband, Or, who knows so much about my research topic that should be awarded an honorary degree.
Acknowledgements

The years at Bar-Ilan University, and especially the years during my PhD, were among the best of my life. It was due to passion to my research topic but also thanks to a supporting environment that I was happy to wake up every morning and come to the lab.

First and foremost I would like to express my deepest gratitude to my advisor, Ido Dagan. Ido not only introduced me to NLP but also taught me how to conduct sound scientific research, convey the results clearly and distill the main argument of a research paper. He has endless patience and a very kind and generous attitude towards his students. I would like to believe I developed a somewhat more relaxed working mode thanks to him.

I also had the privilege of working with Yoav Goldberg to whom I owe my knowledge of deep learning. It was a pleasure to collaborate with him and I look up to his productivity in work and his ability to conduct fast and high quality research.

Many colleagues played an important role in the success of my PhD. I would like to thank my lab mates at the NLP lab in Bar-Ilan University (sorted alphabetically): Alon Jacovi, Amit Moryossef, Arie Cattan, Aryeh Tiktinsky, Asher Stern, Assaf Amrami, Assaf Toledo, Aviv Weinstein, Ayal Klein, Chaya Liebkind, Daniel Juravski, Daniela Stepanov, Eli Kiperwasser, Eyal Orbach, Gabriel Stanovsky, Hila Gonen, Jessica Ficler, Karin Brisker, Lilly Kotlerman, Matan Ben Noach, Meni Adler, Micah Shlain, Natalie Shapira, Noa Lubin, Ofer Bronstein, Ofer Sabo, Ohad Rosen, Omer Levy, Oren Melamud, Ori Shapira, Paul Roit, Rachel Wities, Roe Aharoni, Shachar Rosenman, Shani Barhom, Shauli Ravfo-
gel, Vasily Konovalov, Yanai Elazar, and Yehudit Meged.

I am very grateful to my collaborators outside the NLP lab, who contributed to the success of my PhD: Alon Eirew, Chris Callison-Burch, Claudio Delli Bovi, Dan Roth, Dominik Schlechtweg, Enrico Santus, Eugenio Martinez Camara, Gavriel Habib (who doubles as my co-author and my brother), Guy Tevet, Horacio Saggion, Iryna Gurevych, Jacob Goldberger, Jonathan Berant, Jose Camacho-Collados, Kathrin Eichler, Luis Espinosa-Anke, Marianna Apidianaki, Max Glockner, Michael Bugert, Nils Reimers, Roberto Navigli, Sebastian Padó, Sergio Oramas, Shyam Upadhyay, Sneha Rajana, Tae-Gil Noh, Tommaso Pasini, Tu Vu, and Vivi Nastase.

Special thanks to Eyal Shnarch, who mentored me during my internship at IBM Research, and to Chris Waterson, who mentored me during my internship at Google Research. Both internships taught me about working in large research groups and doing research that makes an impact, and I learned and improved my working methods thanks to both.

Last but not least, I am grateful to my family: my parents Haim and Tikva, who always encouraged me to study and supported me along the way, and my siblings, Shiri, Gali, and Gavriel. In no way have I been able to complete a successful PhD without the endless help and support from my beloved husband, Or. He was there for me all along, making my success his priority, taking care of me and taking away from me other tasks so that I can focus solely on my research.
Preface

Portions of this thesis are joint work and have appeared elsewhere.


Chapter 4, “Improving Hypernymy Detection with an Integrated Path-based and Distributional Method”, appeared in the proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016) (Shwartz et al., 2016).

Chapter 5, “Path-based vs. Distributional Information in Recognizing Lexical Semantic Relations”, appeared in the proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex 2016) (Shwartz and Dagan, 2016b).


Chapter 8, “Paraphrase to Explicate: Revealing Implicit Noun-Compound Relations”, appeared in the proceedings of the 56th Annual Meeting of the As-

This work was supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 1951/17, the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), the Clore Scholars Programme (2017), and the AI2 Key Scientific Challenges Program (2017).
## Contents

Abstract i

1 Introduction 1
   1.1 Semantic Relation Classification . . . . . . . . . . . . . . . . . . 2
   1.2 Predicate Paraphrases . . . . . . . . . . . . . . . . . . . . . . . 4
   1.3 Interpreting Noun Compounds . . . . . . . . . . . . . . . . . . . 5
   1.4 Context-sensitive Lexical Inference . . . . . . . . . . . . . . . . . 7
   1.5 Contributions and Outline . . . . . . . . . . . . . . . . . . . . . . 8

2 Background 11
   2.1 Distributional Semantics . . . . . . . . . . . . . . . . . . . . . . . . 11
      2.1.1 Distributional Semantic Models . . . . . . . . . . . . . . . 12
      2.1.2 Word Embeddings . . . . . . . . . . . . . . . . . . . . . . . . 12
      2.1.3 Contextualized Word Representations . . . . . . . . . . . . 15
   2.2 Taxonomies and Knowledge Resources . . . . . . . . . . . . . . . . 15
      2.2.1 WordNet . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
      2.2.2 Knowledge Bases . . . . . . . . . . . . . . . . . . . . . . . . 16
   2.3 Lexical Inference Benchmarks . . . . . . . . . . . . . . . . . . . . . 17
      2.3.1 Word Similarity and Relatedness . . . . . . . . . . . . . . . 17
2.3.2 Lexical Substitution ............................................... 18
2.3.3 Relation Classification .......................................... 18
2.3.4 Paraphrasing ....................................................... 19

3  Hypernyms under Siege:
Linguistically-motivated Artillery for Hypernymy Detection 20

4  Improving Hypernymy Detection
with an Integrated Path-based and Distributional Method 32

5  Path-based vs. Distributional Information
in Recognizing Lexical Semantic Relations 43

6  Acquiring Predicate Paraphrases from News Tweets 50

7  Olive Oil Is Made of Olives, Baby Oil Is Made for Babies:
Interpreting Noun Compounds Using Paraphrases in a Neural Model 57

8  Paraphrase to Explicate:
Revealing Implicit Noun-Compound Relations 65

9  Adding Context to Semantic Data-Driven Paraphrasing 78

10 Conclusion .............................................................. 85
  10.1 Going Beyond the Distributional Hypothesis ............... 85
  10.2 Modeling Context-sensitivity ............................... 86
  10.3 Representations Beyond the Word-Level .................. 88
  10.4 Going Forward: Improving Downstream Applications .... 90

Bibliography ............................................................ 92

Hebrew Abstract
Abstract

Applications of natural text understanding have to deal with two major issues in human language: *lexical variability*, i.e. the same meaning can be expressed in various ways, and *ambiguity*, i.e. the same word can have different meanings, depending on the context in which it is said.

Targeting such issues, one of the most basic components in natural language understanding is recognizing lexical inferences. Lexical inference corresponds to a semantic relation that holds between two lexical items (words or multi-word expressions), when the meaning of one can be inferred from the other. These semantic relations can be of various types such as synonymy (*elevator*/*lift*), hypernymy or Is-A (*cat*/*animal*, *Google*/*company*), and part-of (*London*/*England*).

Recognizing lexical inferences can benefit various natural language processing (NLP) applications. In text summarization, lexical inference can help identifying redundancy, when two candidate sentences for the summary differ only in terms that hold a lexical inference relation. For example, when summarizing reviews of phone models, it would help knowing that “the battery is long-lasting” is redundant with “the battery is enduring”. In reading comprehension, answering the question “which phones have long-lasting batteries?” given the text “Galaxy has a long-lasting battery”, requires knowing that *Galaxy* is a model of a *phone*.

Most NLP applications today rely on word embeddings as means of information about the semantic similarity between words. A word embedding is a dense vector obtained by aggregating the contexts with which the word co-
occurred in a large text corpora, based on the distributional hypothesis (Harris, 1954). Word embeddings capture topical similarity, or relatedness (elevator/floor), and also functional similarity (elevator/escalator). However, many tasks require knowing the exact relationship between two words, and this information is not straightforwardly available in word embeddings. Instead, the most similar words of a word like cat can have various different relations with it: tail (part of), dog (sibling), kitten (hyponym), and feline (hypernym).

In this thesis I present various algorithms developed to recognize lexical inferences. On one thread, I investigate ways to recognize the semantic relation that holds between a pair of words. As part of this exploration, I propose a novel method that integrates two complementary information sources: path-based and distributional. The first relies on the joint occurrences of the two words in the corpus (e.g. cats and other animals), while the second utilizes the word embeddings of each word, obtained from their disjoint occurrences in the corpus.

The proposed model overcomes issues observed in previous work. First, previous work showed that purely distributional methods can only determine properties of each single word, e.g. “how likely is animal to be a hypernym?” rather than “how likely is animal to be a hypernym of cat?”. When this information is combined with path-based information, the model is also aware of the specific relationship between the two words. Second, the model represents paths using a neural model that allows for better generalization, capturing semantic relatedness between paths, e.g. “X is Y” and “X is defined as Y”. Finally, the model became the new state-of-the-art for this task, with significant gap from previous methods.

In another line of work, I constructed a resource of predicate paraphrases. An example of such a pair of predicates is ‘X dies at Y’ and ‘X lives until Y’, which would have a similar meaning under the assignment of a person to X and an age to Y. This resource was constructed automatically from news headlines in Twitter, relying on the redundancy of various headlines discussing the same event to extract lexically-divergent paraphrases.

Finally, I worked on interpreting the implicit semantic relation that holds between the constituents of a noun-compound. For example, olive oil is oil
made of olives, while baby oil is oil made for babies. The interpretation of noun-compounds has been addressed in the literature either by classifying them to a fixed inventory of ontological relationships (e.g. olive oil: SOURCE vs. baby oil: PURPOSE) or by generating various free text paraphrases that describe the relation in a more expressive manner (e.g. “olive oil is oil made of olives”). I explore both variants of this task.

For the classification task, I applied the same classification method that I developed to recognize semantic relations between arbitrary words, gaining performance improvement upon previous work. Even so, the performance on this task remains mediocre for all methods, partially due to the strict task definition. One of the conclusions drawn from this work was that noun compounds often exhibit multiple semantic relations, and therefore the paraphrasing paradigm is more informative.

The method I developed for the paraphrasing task achieved the state-of-the-art performance on this task, thanks to its ability to generalize for both semantically similar constituent nouns (e.g. predicting the paraphrases of the unobserved steel knife based on the similar observed plastic spoon) and semantically similar paraphrases (e.g. predicting X made of Y when X extracted from Y was observed).

My proposed models outperform highly competitive baselines and improve the state-of-the-art in several benchmarks for lexical inference. However, it is worth noting that applying lexical inferences within applications imposes additional difficulty in detecting whether the potential lexical inference is valid within a given context, as is shown in the last chapter of this thesis. The recent development of context-aware word embeddings have advanced the ability of NLP applications to recognize the correct sense of a polysemous word in a given context, but other semantic phenomena related to lexical inferences remain challenging even to them. In the conclusion of this thesis I share my thoughts on the remaining challenges and possible approaches to address them in the future.
Chapter 1

Introduction

Making informed decisions, such as which model of phone to buy, or which hotel to stay in while traveling, is often aided by reading online reviews and interacting on social networks. This requires reading and processing many texts, some of which are redundant, while others may be contradicting. Our capacity as humans to process the infeasible amount of data in the web is limited. To that end, the natural language processing community has been developing applications such as multi-document summarization (Barzilay et al., 1999) and machine reading comprehension (Hirschman et al., 1999), that aid people by processing large amounts of texts into concise information.

Such applications have to deal with lexical variability, i.e. the same meaning can be expressed in various ways. Targeting this issue, one of the basic components in natural language understanding is recognizing lexical inferences. Lexical inference corresponds to a semantic relation that holds between two lexical items (words or multiword expressions), when the meaning of one can be inferred from the other. In text summarization, for example, lexical inference can help identifying redundancy, when two candidate sentences for the summary differ only in terms that hold a lexical inference relation (e.g. “the battery is long-lasting” and “the battery is enduring”). In reading comprehension, answering the question “which phones have long-lasting batteries?” given the text “Galaxy has a long-lasting battery”, requires knowing that Galaxy is a model of a
CHAPTER 1. INTRODUCTION

In this chapter I provide background for the various aspects of lexical inference discussed in the thesis. First, I focus on 3 tasks pertaining to lexical inference: identifying the semantic relation that holds between a pair of words (Section 1.1), extracting pairs of texts that have roughly the same meaning (Section 1.2), and identifying the semantic relation that holds between the constituents of a noun compound (Section 1.3). Then I discuss the importance of considering the context in which the lexical inferences are applied (Section 1.4). Finally, in Section 1.5 I list the contributions of this work and provide an outline for the thesis.

1.1 Semantic Relation Classification

Many NLP applications today rely on word embeddings to inform them about the semantic relation between words. However, word embeddings capture a fuzzy distributional similarity between words, which conflates multiple semantic relations such as synonymy (elevator / lift), hypernymy/hyponymy (cat / animal), holonymy/meronymy (cat / tail), co-hyponymy (cat / dog), antonymy (hot / cold), and general relatedness or topical similarity (elevator / floor). Applications that require knowing the specific semantic relation that holds between a pair of words must get this information from other sources.

While semantic taxonomies, like WordNet (Fellbaum, 1998), define the semantic relations between word types (e.g. (cat, animal, hypernymy), (cat, tail, part-of)), they are limited in scope and domain. Therefore, automated methods have been developed to determine, for a given term-pair \((x, y)\), the semantic relation that holds between them, based on their occurrences in a large corpus.

Two main information sources are used to recognize these relations in corpus-based methods: path-based and distributional. Path-based methods consider the joint occurrences of \(x\) and \(y\) in the corpus, where the lexico-syntactic paths that connect the terms are typically used as features (e.g. “cat and other animals” indicates that cat is a “type of” animal). Hearst (1992) identified a small set of
frequent paths that indicate hypernymy, e.g. $Y$ such as $X$. Snow et al. (2004) represented each $(x, y)$ term-pair as the multiset of dependency paths connecting their co-occurrences in a corpus, and trained a classifier to predict hypernymy, based on these features.

Using individual paths as features results in a huge, sparse feature space. While some paths are rare, they often consist of certain unimportant components. For instance, “Spelt is a species of wheat” and “Fantasy is a genre of fiction” yield two different paths: “$X$ be species of $Y$” and “$X$ be genre of $Y$”, while both indicating that $X$ is-a $Y$. A possible solution is to generalize paths by replacing words along the path with their part-of-speech tags or with wild cards, as done in the PATTY system (Nakashole et al., 2012).

In contrast to the path based approach, distributional methods are based on the individual occurrences of each term, relying on the distributional hypothesis (Harris, 1954), according to which words that occur in similar contexts tend to have similar meanings. In recent years, this was the most common approach pursued, and it has shown to perform well on several common benchmarks (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). However, it has been later pointed out that these methods suffer from lexical memorization, and can only determine properties of a single word, e.g. “how likely is animal to be a hypernym?” rather than “how likely is animal to be a hypernym of cat?” (Levy et al., 2015a).

Overall, the state-of-the-art path-based methods perform worse than the distributional ones. This stems from a major limitation of path-based methods: they require that the terms of the pair occur together in the corpus, limiting the recall of these methods. While distributional methods have no such requirement, they are usually less precise in detecting a specific semantic relation, and perform best on detecting broad semantic similarity between terms. Though these approaches seem complementary, there has been rather little previous work on integrating them.

Chapters 4 and 5 describe an integrated path-based and distributional model for semantic relation classification. First, I focused on improving path representation using a recurrent neural network, resulting in a path-based model that
performs significantly better than prior path-based methods, and matches the performance of the previously superior distributional methods. In particular, I demonstrated that the increase in recall is a result of generalizing semantically-similar paths, in contrast to prior methods, which either make no generalizations or over-generalize paths. Then, I extend the approach to integrate both path-based and distributional signals, significantly improving upon the state-of-the-art on this task, and confirming that the approaches are indeed complementary.

I provide further analysis of specific cases in which the path-based signal especially contributes to the performance of the model: 1) in strict evaluation setups, in which a model cannot benefit from memorizing the typicality of a single word to a relation (e.g. animal is often a hypernym); 2) for rare words or senses; and 3) when the relationship is less prototypical, e.g. cherry and pick are classified to the event relation.

1.2 Predicate Paraphrases

While the above method works well for terms (words and noun phrases), the following is concerned with verbal phrases. Recognizing that various textual descriptions across multiple texts refer to the same event or action can benefit NLP applications such as recognizing textual entailment (Dagan et al., 2013) and question answering. For example, to answer “when did the US Supreme Court approve same-sex marriage?” given the text “In June 2015, the Supreme Court ruled for same-sex marriage”, approve and ruled for should be identified as describing the same action.

Much effort has been devoted to identifying predicate paraphrases, e.g. “X buy Y” and “X acquire Y”. Two main approaches were proposed for that goal; the first leverages the similarity in argument distribution across a large corpus between two predicates (Lin and Pantel, 2001; Berant et al., 2010), e.g. “X buy Y” and “X acquire Y” are similar because they are both instantiated with names of purchased companies as Y and names of companies that purchased others as X. The second approach exploits bilingual parallel corpora, extract-
ing as paraphrases pairs of texts that were translated identically to foreign languages (Barzilay and McKeown, 2001; Ganitkevitch et al., 2013).

A third approach was proposed to harvest paraphrases from multiple mentions of the same event in news articles. This approach assumes that various redundant reports make different lexical choices to describe the same event (Shinyama et al., 2002; Shinyama and Sekine, 2006; Roth and Frank, 2012; Zhang and Weld, 2013). In chapter 6, I present an ever-growing resource, currently consisting of millions of predicate paraphrase template pairs such as “X dies at Y” and “X lives until Y”, following this event coreference assumption.

The resource was from news tweets. Pairs of predicate templates (e.g. “X dies at Y” and “X lives until Y”) that share mutual arguments (e.g. Chuck Berry, 90) and which were published on the same day, were considered as paraphrases. Comparison to existing resources showed that the resource complements other, much larger paraphrase resources with non-consecutive predicates (e.g. “reveal X to Y” / “share X with Y”) and paraphrases which are highly context specific (e.g. “X get Y” / “X sentence to Y” in the context of a person and the time they are about to serve in prison).

1.3 Interpreting Noun Compounds

A special case of identifying the semantic relation that holds between a pair of nouns can be found in noun compounds. Noun compounds implicitly convey a certain semantic relation that holds between their constituents. For example, a ‘birthday cake’ is a cake eaten on a birthday, while ‘apple cake’ is a cake made of apples. Interpreting noun compounds by explicating their implicit relationship is beneficial for many natural language understanding tasks, especially given the prevalence of noun compounds in English (Nakov, 2013).

Noun compound interpretation has been addressed in the literature in two variants. The first variant is a “closed” classification task to a set of predefined ontological relations, for example, classifying apple cake to SOURCE vs. birthday cake to TIME. As an alternative to the strict classification, Nakov and Hearst
(2006) suggested that the semantics of a noun compound could be expressed with multiple prepositional and verbal paraphrases. For example, *apple cake* is a cake from, made of, or which contains apples.

Recent approaches for classification start by learning vector representations for noun compounds, and then use them as features for the classifier. Most commonly, the noun compound vector is obtained by applying a function to its constituents’ distributional representations, e.g. $\text{vec}(\text{apple cake}) = f(\text{vec(apple)}, \text{vec(cake)})$. Various functions have been proposed, usually based on vector arithmetics, and they are often learned with the objective of minimizing the distance between the learned vector and the observed vector (computed from corpus occurrences) of each noun compound. Using compositional noun compound vectors for classification leads to promising results (Van de Cruys et al., 2013; Dima and Hinrichs, 2015), however, Dima (2016) argues that it stems from memorizing prototypical words for each relation. For example, classifying any noun compound with the head *cake* to the SOURCE relation, regardless of the modifier.

In chapter 7, I applied lessons learned from the parallel task of semantic relation classification and adapted LexNET to the noun compound classification task, combining the distributional embeddings of the constituent nouns with the lexico-syntactic paths connecting them in the corpus. The proposed method outperformed the baselines in strict evaluation setups in which the model could not benefit from lexical memorization. Having said that, the analysis also revealed issues related to the dataset and task definition, suggesting that the less strict paraphrasing paradigm is more suitable for the interpretation of noun compounds.

For the paraphrasing task, it is common to start with a pre-processing step of extracting the joint occurrences of the constituents from a corpus to generate a list of candidate paraphrases. Most methods lack the ability to generalize, and have a hard time interpreting infrequent or new noun compounds. For example, if the corpus does not contain paraphrases for *plastic spoon*, it is desirable for a model to predict paraphrases of a similar compound, such as *steel knife*. Similarly, if a certain noun compound only appears with few paraphrases in the corpus, we would like the model to output additional semantically-similar
paraphrases, as in predicting “X made from Y” when “X extracted from Y” has been observed.

In chapter 8, I present a method for noun compound paraphrasing. I follow previous work and extract the joint occurrences of the constituents from a corpus to generate a list of candidate paraphrases. As opposed to previous approaches, that focus on predicting a paraphrase template for a given noun-compound (e.g. predicting “X made of Y” for apple cake), I reformulate the task as a multi-task learning problem, and train the model to also predict a missing constituent of the compound given the paraphrase template and the other constituent (e.g. predicting apple for “cake made of X”). This enables better generalization for both unseen noun compounds and rare paraphrases. Indeed, the model’s ability to generalize leads to improved performance in challenging evaluation settings, becoming the new-state-of-the-art on the task.

1.4 Context-sensitive Lexical Inference

Applying lexical inferences within applications imposes additional difficulty in detecting whether the potential lexical inference is valid within a given context. First, the application needs to address the ambiguity issue, by considering the meaning of each term within its context. For instance, play is a hypernym of compete in certain contexts, but not in the context of playing the national anthem at a sports competition. Second, the soundness of inferences within context is conditioned on the specific semantic relation that holds between the terms. For instance, in some sentences, a part infers its whole (“I live in London” → “I live in England”), while in others it may not (“I’m leaving London” ∉ “I’m leaving England”).

Most work on lexical inference today judges the relationship between a pair of terms out-of-context. The existing datasets for in-context lexical inference are annotated for either coarse-grained semantic relations, such as similarity or relatedness, which may not be sufficiently informative, or for lexical substitution which is a much stricter definition requiring words to be near synonyms.
In chapter 9, I studied pairs of words which were annotated to the semantic relations that hold between them in a context-agnostic manner. By adding context and re-annotating the data, I find that almost half of the semantically-related term-pairs become unrelated when the context is specified. This happens, for example, due to polysemy, as in the relation between piece and strip when the latter appears in the sentence “...the labor leader was found to have gone to a strip club during a taxpayer funded trip”. Furthermore, a generic out-of-context relation may change within a given context. For instance, race is more specific than competition in general, but in specific contexts it can be judged as equivalent, as in “...will pull out of the race for presidency”.

1.5 Contributions and Outline

In chapters 3 and 4 I focus on the hypernymy relationship. Detecting hypernymy relations is a key task in NLP. In chapter 3, I investigate an extensive number of unsupervised distributional measures for hypernymy detection. These measures are based on high-dimensional distributional vectors, also called distributional semantic models (DSMs). I investigated the combination of proposed measures from the literature with various DSMs that differ by context type and feature weighting, and analyzed the performance of the different methods based on their linguistic motivation. The contributions of this work are 1) suggesting a “recipe” for the best setting to differentiate hypernyms from each other semantic relation; and 2) comparison to the state-of-the-art shows that although supervised methods generally outperform the unsupervised ones, the former are sensitive to the distribution of training instances, hurting their reliability. Being based on general linguistic hypotheses and independent from training data, unsupervised measures are more robust.

In chapter 4, I present HypeNET: an integrated path-based and distributional model for hypernymy detection. The model achieved state-of-the-art performance on the task by combining these two complementary information sources and thanks to an improved path representation. The paper describing HypeNET was recognized as an “outstanding paper” at ACL 2016. In chapter 5, I extend
HypeNET to LexNET, which supports classification to multiple semantic relations. The empirical results show that this method is effective in the multi-class setting as well. LexNET achieved the new state-of-the-art for semantic relation classification, and won the CogALEX 2016 shared task.

In chapter 6, I present a resource consisting of predicate paraphrase template pairs (e.g. “X dies at Y” and “X lives until Y”). Knowing that the two can have the same meaning—under the same assignment of arguments and in certain contexts—can contribute to many NLP applications dealing with lexical variability. The resource I present was collected from news tweets. Pairs of predicate templates that share mutual arguments (e.g. Chuck Berry, 90) and which were published on the same day, were considered as paraphrases. The resource is ever-growing, currently consisting of 4 million pairs (as of April 2019).

In chapters 7 and 8 I address the interpretation of the semantic relation that holds between the constituents of a noun compound. In chapter 7, I present a method for the classification task. I adapted LexNET to this task, combining the distributional embeddings of the constituent nouns with the lexico-syntactic paths connecting them in the corpus. The method outperformed the baselines in strict evaluation setups. In chapter 8, I present a state-of-the-art method for the paraphrasing task. The model enables better generalization for both unseen noun compounds and rare paraphrases by jointly predicting a paraphrase template for a given noun-compound (e.g. predicting “X made of Y” for olive oil) and a a missing constituent of the compound given the paraphrase template and the other constituent (e.g. predicting olive for “oil made of X”).

In chapter 9, I touch upon the importance of modelling the context in which the lexical inference is applied. I studied pairs of words which were annotated to the semantic relations that hold between them in both context-agnostic manner and with respect to a given context. I find that adding context often changes this semantic relation, concluding that modelling the context in which the inference is applied is crucial. The paper described in this chapter was recognized as a “best paper” at SEM 2016.

Finally, in chapter 10, I summarize the contributions of this thesis, which include (1) going beyond the commonly used distributional representations and
Table 1: Summary of the tasks presented in this thesis, specifying inputs and outputs, the approach taken, and the chapter in which each work is presented.

investigating additional information sources for recognizing lexical inferences; (2) an analysis of the effect of context on the applicability of lexical inferences; and (3) two approaches for making inferences related to the implicit relationship between the constituents of a noun compound. I share my view of the remaining challenges in each one of these aspects of lexical inference and discuss the potential approaches for future research.

For the reader’s convenience, Table 1 presents a summary of the tasks presented in this thesis.
Chapter 2

Background

In this chapter I provide complementary background to that in the following chapters. First, I give an overview of distributional semantics, which is the basis on which words are represented across NLP applications. Then I briefly review other sources of lexical information, which are given in a structured manner. Although the algorithms presented in this thesis do not make use of such resources, but are instead corpus-based, knowledge resources are used in chapter 4 to create training data. Finally, I list a few other lexical tasks and benchmarks which were not used in this work.

2.1 Distributional Semantics

A word is often considered as the most basic unit of meaning in language. In NLP, it is a common practice to represent words as vectors. The various approaches to create such vectors rely on the distributional hypothesis, according to which words that occur in similar contexts tend to have similar meanings (Harris, 1954), or as described by Firth (1957): “You shall know a word by the company it keeps”.

2.1.1 Distributional Semantic Models

Based on this hypothesis, distributional semantic models (DSMs) compute a matrix of word-word co-occurrences in a large text corpus, and then apply a weighting function to the matrix. Each row corresponds to the vector representing a specific word, and the vectors representing semantically-similar words (e.g. cat and kitten) are expected to be similar in terms of vector similarity (typically cosine similarity).

DSMs differ from each other in the choice of context type and feature weighting. The context type defines the neighbors of each word. Most commonly, window-based context is used, where the contexts of a target word $w_i$ are the words surrounding it in a $k$-sized window: $w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}$. Alternatively, dependency-based context considers the neighbors in a dependency parse tree (Padó and Lapata, 2007; Baroni and Lenci, 2010). The context type can affect the type of similarity that the DSM captures: larger windows tend to capture topical similarity (e.g. dancer / performance), whereas smaller windows and dependency-based contexts capture more functional similarity (e.g. dancer / singer) (Levy and Goldberg, 2014).

2.1.2 Word Embeddings

The dimension of distributional vectors is the size of the vocabulary, $|V|$, which can easily reach a few hundred thousand words. The vectors are sparse, because most words do not occur next to each other. To overcome the computational obstacles, it was common to apply dimensionality reduction algorithms such as Singular Value Decomposition (SVD) to the co-occurrence matrix, obtaining $d$-dimensional vectors for some predefined dimension $d << |V|$ (Deerwester et al., 1990).

An alternative to dimensionality reduction was suggested by Bengio et al. (2003): instead of counting co-occurrences, the word vectors are initialized randomly and during training they are updated with the training objective of predicting whether certain words occurred together or not. The result is a low-
dimensional embedding space (typically 50-600 dimensions) in which semantically-similar words are embedded in proximity. Despite being first presented in 2003, word embeddings only gained popularity with the release of word2vec a decade later (Mikolov et al., 2013). Following word2vec, several other embedding algorithms were developed, the most popular of which are GloVe (Pennington et al., 2014), and fastText (Bojanowski et al., 2017), which extends word2vec by adding information about subwords (bag of character n-grams).

**Usage of Word Embeddings in Downstream Tasks**

In recent years, word embeddings have been used successfully across tasks, starting from syntactic tasks such as part-of-speech tagging, noun phrase chunking, and syntactic parsing, and continuing with semantic tasks such as named entity recognition, semantic role labeling, sentiment analysis and other text classification variants, paraphrase detection, and recognizing textual entailment (Bakarov, 2018). Initializing the representation layer with pre-trained embeddings can especially benefit tasks with a small amount of training data which does not allow learning a meaningful representation from scratch.

**Usage of Word Vectors in Intrinsic Lexical Inference Tasks**

Earlier methods developed unsupervised measures based on high-dimensional distributional vectors, to detect lexical inferences. This started with symmetric similarity measures (Lin, 1998), and followed by directional measures of hypernymy based on the distributional inclusion hypothesis (Weeds and Weir, 2003; Kotlerman et al., 2010). This hypothesis states that the contexts of a hyponym are expected to be largely included in those of its hypernym. More recent work (Santus et al., 2014; Rimell, 2014) introduced new measures, based on the assumption that the most typical linguistic contexts of a hypernym are less informative than those of its hyponyms.

More recently, the focus of the distributional approach for recognizing lexical inferences shifted to supervised methods. In these methods, a pair of words \( x \) and \( y \) is represented by a feature vector, and a classifier is trained on these vec-
tors to predict the semantic relation that holds between them. Several methods are used to represent term-pairs as a combination of each term’s embeddings vector: concatenation $\vec{x} \oplus \vec{y}$ (Baroni et al., 2012), difference $\vec{y} - \vec{x}$ (Roller et al., 2014; Weeds et al., 2014), and dot-product $\vec{x} \cdot \vec{y}$. Using word embeddings, these methods are easy to apply, and show good results (Baroni et al., 2012; Roller et al., 2014).

In several chapters of this thesis, I return to the disadvantage of relying solely on word embeddings to determine semantic relations between words. As shown by Levy et al. (2015a), these methods can only rely on the properties of each single word in a pair, rather than modelling the relationship between the words. To that end, complementary information sources can mitigate this issue, as I show in chapter 4.

**Limitations**

Two major shortcomings of any kind of word vectors based on the distributional hypothesis are 1) they conflate all the different senses of a word into a single vector; and 2) they conflate the various semantic relations into one fuzzy similarity relation. To address the polysemy issue, there have been attempts to learn separate vectors for each sense of a word. These methods typically follow one of two main approaches: they either rely on a predefined sense inventory, or induce the senses in an unsupervised manner (Camacho-Collados and Pilehvar, 2018). However, the utility of such embeddings highly depends on defining the required granularity level of the senses, and the produced vectors do not deal with minor meaning shifts within the same sense, which can happen in certain contexts.

With respect to the fuzzy similarity signal the vectors yield, consider a word like *cat*. Among the most similar words to *cat*, one may find hypernyms (*animal, pet, feline*), hyponyms (*kitten*), co-hyponyms / siblings (*dog*), parts (*tail*), and related words (*purr*). Perhaps the most concerning issue is the inability to discern, based on distributional similarity, synonyms from antonyms (*hot / cold*) and mutually exclusive terms (*Sunday / Monday*), since words related in any of these relations tend to appear in similar contexts.
Finally, while in high-dimensional distributional vectors each feature corresponds to a context, the features of word embeddings are not interpretable. In chapter 3 I describe various measures for hypernymy detection that depend on the feature interpretation as means of linguistic motivation, and therefore cannot work with word embeddings.

### 2.1.3 Contextualized Word Representations

Recently, a new paradigm has emerged: instead of using static word vectors, the representation of a word is computed dynamically given its sentential context (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018). Doing so, such contextualized word representations largely address polysemy, as they no longer conflate all the different senses of a word into a single vector. These models are pre-trained as general purpose language models using a large-scale unannotated corpus. They are meant to be used as a representation layer in downstream tasks, either fine-tuned to the task or fixed. Contextualized word representations have shown to significantly improve the performance upon using regular word embeddings, across various NLP applications.

### 2.2 Taxonomies and Knowledge Resources

While most work on recognizing lexical inferences is corpus-based, it is also possible to mine high-precision lexical inferences from structured resources, such as those detailed below.

#### 2.2.1 WordNet

WordNet (Fellbaum, 1998) is an ontology of the English language in which nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets) and are interlinked by means of lexical semantic relations (hypernymy, meronymy, etc).

WordNet is widely used for identifying lexical inferences, usually in an un-
supervised setting where the relations relevant for each specific inference task are manually selected a priori. One approach looks for chains of these pre-defined relations, e.g. inference via a chain of hypernyms: dog $\rightarrow$ canine $\rightarrow$ carnivore $\rightarrow$ placental mammal $\rightarrow$ mammal (Harabagiu and Moldovan, 1998). Another approach is via WordNet Similarity, which takes two synsets and returns a numeric value that represents their similarity based on WordNet’s hierarchical hypernymy structure (Pedersen et al., 2004). Popular similarity measures consider only hypernyms and synonyms as a manual design choice (Wu and Palmer, 1994; Resnik, 1995).

### 2.2.2 Knowledge Bases

While WordNet is quite extensive (containing 150,000 nouns annotated to 13 different semantic relations), it is hand-crafted by expert lexicographers, and thus cannot compete in terms of scale with community-built knowledge bases (KBs) which connect millions of entities through a rich variety of structured relations. Specifically, by definition, WordNet does not cover many proper-names (Beyoncé $\rightarrow$ artist) and recent terminology (Facebook $\rightarrow$ social network). These can be found in the rich and up-to-date structured KBs.

Wikidata (Vrandečić, 2012) contains facts about 55 million entities with 2,000 different relations, which were extracted automatically and edited by humans. DBPedia (Auer et al., 2007) contains structured information from Wikipedia: info boxes, redirections, disambiguation links, etc. The English version of the DBpedia currently describes 4.58 million entities and $\sim$1,300 relations. Yago (Suchanek et al., 2007) is a semantic knowledge base derived from Wikipedia, WordNet, and GeoNames, consisting of more than 10 million entities and 114 relations. Finally, ConceptNet (Speer and Havasi, 2012) contains 37 commonsense relationships between 3.9 million entities, and is collected from a variety of resources, including crowd-sourced resources, games with a purpose, and expert-created resources such as WordNet.

In recent years, numerous NLP applications leveraged KBs as means of obtaining a richer knowledge representation. Examples where such KBs proved
beneficial include word sense disambiguation, semantic role labeling, semantic parsing, and text classification (Gurevych et al., 2016). Shwartz et al. (2015) leveraged KBs to extract lexical inference relations, concluding that they are complementary to corpus-based methods, especially in scenarios in which precision is more important than recall.

2.3 Lexical Inference Benchmarks

In this section I provide an overview of lexical inference benchmarks which are complementary to those mentioned in this thesis. In these benchmarks, each entry consists of a pair of terms $x$ and $y$, annotated to certain semantic relation(s) that may hold between them. The tasks and benchmarks differ on two main axes: (1) the semantic relations; and (2) whether the terms are given within context or out-of-context.

2.3.1 Word Similarity and Relatedness

The goal in this task is to predict a number indicating the degree of similarity or association between $x$ and $y$. The task is commonly used as intrinsic evaluation to estimate the quality of word embeddings, by computing the correlation between the human judgements and the cosine similarity between the vectors. Common datasets for the task include WordSim-353 (Finkelstein et al., 2002), SimLex-999 (Hill et al., 2014), and MEN (Bruni et al., 2014).

Although all these benchmarks group several semantic relations under a single coarse-grained relation, there is a fine distinction of similarity vs. relatedness. Similar words share numerous common properties (e.g. cup and mug) while related terms appear in the same broad contexts and are pertinent to the same domain or topic (e.g. cup and coffee). The SimLex-999 dataset captures strict similarity, and WordSim-353 was divided by Zesch et al. (2008) into distinct sets consisting of either similarity or relatedness.

In most datasets, word-pairs are annotated to relatedness or similarity between them in some (unspecified) contexts. The exceptions are the Stanford’s
Contextual Word Similarity dataset (SCWS) (Huang et al., 2012) and TR9856 (Levy et al., 2015b), in which the annotation is given with respect to a given context.

### 2.3.2 Lexical Substitution

The lexical substitution task requires identifying meaning-preserving substitutes for a target word in a given sentential context. Common datasets for this task are the SemEval 2007 task (McCarthy and Navigli, 2007) and the “All-Words” corpus (Kremer et al., 2014). Lexical substitutability is stricter than similarity and is usually defined by synonymy ($\text{cold} / \text{chilly}$) or functional similarity ($\text{dollars} / \text{euros}$). The task is typically addressed by modelling the context sentence and predicting words that can fit in that context (Melamud et al., 2013, 2015, 2016).

### 2.3.3 Relation Classification

In relation classification the goal is to classify the relation that is expressed between two target terms in a given sentence to one of predefined relation classes. To illustrate, consider the following sentence, from the SemEval-2010 relation classification task dataset (Hendrickx et al., 2009): “The [apples]$_{e_1}$ are in the [basket]$_{e_2}$”. Here, the relation expressed between the target entities is $\text{Content} - \text{Container}(e_1, e_2)$.

While relation classification and semantic relation classification (section 1.1) are both concerned with identifying semantic relations that hold for pairs of terms, they differ in a major respect. In semantic relation classification, the goal is to recognize a generic lexical-semantic relation between terms that holds in many contexts, while in relation classification the relation should be expressed in the given text. A common best practice is to rely on the shortest dependency path between the target entities (Fundel et al., 2007; Xu et al., 2015; Liu et al., 2015).
2.3.4 Paraphrasing

Paraphrasing refers to extraction of pairs of differing textual realizations of the same meaning. These pairs of texts can be words, other lexical items, full sentences, or predicate templates as described in chapter 6. Two typical approaches to extract paraphrases are translation-based and event-based.

Barzilay and McKeown (2001) were the first to extract paraphrases from multiple translations of the same text, a technique referred to as “bilingual pivoting” or “back-translation”. Following, Ganitkevitch et al. (2013) applied this technique to multiple foreign languages, resulting in the paraphrase database (PPDB), consisting of 220 million pairs of English paraphrases. Mallinson et al. (2017) showed that this technique is still valid, and yields improved performance, when neural machine translation models are used.

The second approach harvests paraphrases from multiple mentions of the same event in news articles. This approach assumes that various redundant reports make different lexical choices to describe the same event (Shinyama et al., 2002; Shinyama and Sekine, 2006; Roth and Frank, 2012; Zhang and Weld, 2013). Lan et al. (2017) developed a supervised model to collect sentential paraphrases from Twitter. They used pairs of tweets linking to the same URL as positive training examples.
Chapter 3

Hypernyms under Siege: Linguistically-motivated Artillery for Hypernymy Detection
Hypernyms under Siege: Linguistically-motivated Artillery for Hypernymy Detection

Vered Shwartz¹, Enrico Santus²,³ and Dominik Schlechtweg⁴

¹Bar-Ilan University, Ramat-Gan, Israel
²Singapore University of Technology and Design, Singapore
³The Hong Kong Polytechnic University, Hong Kong
⁴University of Stuttgart, Stuttgart, Germany

{vered1986, esantus}@gmail.com, dominik.schlechtweg@gmx.de

Abstract

The fundamental role of hypernymy in NLP has motivated the development of many methods for the automatic identification of this relation, most of which rely on word distribution. We investigate an extensive number of such unsupervised measures, using several distributional semantic models that differ by context type and feature weighting. We analyze the performance of the different methods based on their linguistic motivation. Comparison to the state-of-the-art supervised methods shows that while supervised methods generally outperform the unsupervised ones, the former are sensitive to the distribution of training instances, hurting their reliability. Being based on general linguistic hypotheses and independent from training data, unsupervised measures are more robust, and therefore are still useful artillery for hypernymy detection.

1 Introduction

In the last two decades, the NLP community has invested a consistent effort in developing automated methods to recognize hypernymy. Such effort is motivated by the role this semantic relation plays in a large number of tasks, such as taxonomy creation (Snow et al., 2006; Navigli et al., 2011) and recognizing textual entailment (Dagan et al., 2013). The task has appeared to be, however, a challenging one, and the numerous approaches proposed to tackle it have often shown limitations.

Early corpus-based methods have exploited patterns that may indicate hypernymy (e.g. “animals such as dogs”) (Hearst, 1992; Snow et al., 2005), but the recall limitation of this approach, requiring both words to co-occur in a sentence, motivated the development of methods that rely on adaptations of the distributional hypothesis (Harris, 1954).

The first distributional approaches were unsupervised, assigning a score for each \((x, y)\) word-pair, which is expected to be higher for hypernym pairs than for negative instances. Evaluation is performed using ranking metrics inherited from information retrieval, such as Average Precision (AP) and Mean Average Precision (MAP). Each measure exploits a certain linguistic hypothesis such as the distributional inclusion hypothesis (Weeds and Weir, 2003; Kotlerman et al., 2010) and the distributional informativeness hypothesis (Santus et al., 2014; Rimell, 2014).

In the last couple of years, the focus of the research community shifted to supervised distributional methods, in which each \((x, y)\) word-pair is represented by a combination of \(x\) and \(y\)’s word vectors (e.g. concatenation or difference), and a classifier is trained on these resulting vectors to predict hypernymy (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). While the original methods were based on count-based vectors, in recent years they have been used with word embeddings (Mikolov et al., 2013; Pennington et al., 2014), and have gained popularity thanks to their ease of use and their high performance on several common datasets. However, there have been doubts on whether they can actually learn to recognize hypernymy (Levy et al., 2015b).

Additional recent hypernymy detection methods include a multimodal perspective (Kiela et al., 2015), a supervised method using unsupervised measure scores as features (Santus et al., 2016a), and a neural method integrating path-based and distributional information (Shwartz et al., 2016).

In this paper we perform an extensive evaluation of various unsupervised distributional measures for hypernymy detection, using several distributional semantic models that differ by context type and feature weighting. Some measure vari-
ants and context-types are tested for the first time.1

We demonstrate that since each of these measures captures a different aspect of the hypernymy relation, there is no single measure that consistently performs well in discriminating hypernymy from different semantic relations. We analyze the performance of the measures in different settings and suggest a principled way to select the suitable measure, context type and feature weighting according to the task setting, yielding consistent performance across datasets.

We also compare the unsupervised measures to the state-of-the-art supervised methods. We show that supervised methods outperform the unsupervised ones, while also being more efficient, computed on top of low-dimensional vectors. At the same time, however, our analysis reassesses previous findings suggesting that supervised methods do not actually learn the relation between the words, but only characteristics of a single word in the pair (Levy et al., 2015b). Moreover, since the features in embedding-based classifiers are latent, it is difficult to tell what the classifier has learned. We demonstrate that unsupervised methods, on the other hand, do account for the relation between words in a pair, and are easily interpretable, being based on general linguistic hypotheses.

2 Distributional Semantic Spaces

We created multiple distributional semantic spaces that differ in their context type and feature weighting. As an underlying corpus we used a concatenation of the following two corpora: ukWaC (Ferraresi, 2007), a 2-billion word corpus constructed by crawling the .uk domain, and WaCkypedia_EN (Baroni et al., 2009), a 2009 dump of the English Wikipedia. Both corpora include POS, lemma and dependency parse annotations. Our vocabulary (of target and context words) includes only nouns, verbs and adjectives that occurred at least 100 times in the corpus.

Context Type We use several context types:

- **Window-based contexts**: the contexts of a target word \(w_i\) are the words surrounding it in a \(k\)-sized window: \(w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}\). If the context-type is directional, words occurring before and after \(w_i\) are marked differently, i.e.: \(w_{i-k}/l, \ldots, w_{i-1}/l, w_{i+1}/r, \ldots, w_{i+k}/r\).

- **Dependency-based contexts**: rather than adjacent words in a window, we consider neighbors in a dependency parse tree (Padó and Lapata, 2007; Baroni and Lenci, 2010). The contexts of a target word \(w_i\) are its parent and daughter nodes in the dependency tree (dep). We also experimented with a joint dependency context inspired by Chersoni et al. (2016), in which the contexts of a target word are the parent-sister pairs in the dependency tree (joint). See Figure 1 for an illustration.

  Out-of-vocabulary words are filtered out before applying the window. We experimented with window sizes 2 and 5, directional and indirectional (win2, win2d, win5, win5d).

- **Frequency** - raw frequency (no weighting): \(M_{i,j}\) is the number of co-occurrences of \(w_i\) and \(c_j\) in the corpus.

- **Positive PMI (PPMI)** - pointwise mutual information (PMI) (Church and Hanks, 1990) is defined as the log ratio between the joint probability of \(w\) and \(c\) and the product of their marginal probabilities: \(PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)}, \) where \(\hat{P}(w), \hat{P}(c),\) and \(\hat{P}(w, c)\) are estimated by the relative frequencies of a word \(w\), a context \(c\) and a word-context pair \((w, c)\), respectively. To handle unseen pairs \((w, c)\), yielding \(PMI(w, c) = \log(0) = -\infty\), PPMI (Bullinaria and Levy, 2007) assigns zero to negative PMI scores: \(PPMI(w, c) = \max(PMI(w, c), 0)\).

In addition, one of the measures we used (Santus et al., 2014) required a third feature weighting:
• **Positive LMI (PLMI)** - positive local mutual information (PLMI) (Evert, 2005; Evert, 2008). PPMI was found to have a bias towards rare events. PLMI simply balances PPMI by multiplying it by the co-occurrence frequency of \( w \) and \( c \): 
\[
PLMI(w, c) = freq(w, c) \cdot PPMI(w, c).
\]

3 Unsupervised Hypernymy Detection Measures

We experiment with a large number of unsupervised measures proposed in the literature for distributional hypernymy detection, with some new variants. In the following section, \( \vec{v}_x \) and \( \vec{v}_y \) denote \( x \) and \( y \)'s word vectors (rows in the matrix \( M \)). We consider the scores as measuring to what extent \( y \) is a hypernym of \( x \) (\( x \rightarrow y \)).

3.1 Similarity Measures

Following the *distributional hypothesis* (Harris, 1954), similar words share many contexts, thus have a high similarity score. Although the hypernymy relation is asymmetric, similarity is one of its properties (Santus et al., 2014).

• **Cosine Similarity** (Salton and McGill, 1986) A symmetric similarity measure:
\[
\cos(x, y) = \frac{\vec{v}_x \cdot \vec{v}_y}{\| \vec{v}_x \| \cdot \| \vec{v}_y \|}
\]

• **Lin Similarity** (Lin, 1998) A symmetric similarity measure that quantifies the ratio of shared contexts to the contexts of each word:
\[
Lin(x, y) = \frac{\sum_{c \in \vec{v}_x \cap \vec{v}_y} \vec{v}_x[c] + \vec{v}_y[c]}{\sum_{c \in \vec{v}_x} \vec{v}_x[c] + \sum_{c \in \vec{v}_y} \vec{v}_y[c]}
\]

• **APSyn** (Santus et al., 2016b) A symmetric measure that computes the extent of intersection among the \( N \) most related contexts of two words, weighted according to the rank of the shared contexts (with \( N \) as a hyper-parameter):
\[
APSyn(x, y) = \sum_{c \in N(\vec{v}_x) \cap N(\vec{v}_y)} \frac{1}{\text{rank}_x(c) + \text{rank}_y(c)}
\]

3.2 Inclusion Measures

According to the *distributional inclusion hypothesis*, the prominent contexts of a hyponym \( (x) \) are expected to be included in those of its hypernym \( (y) \).

• **Weeds Precision** (Weeds and Weir, 2003) A directional precision-based similarity measure. This measure quantifies the weighted inclusion of \( x \)'s contexts by \( y \)'s contexts:
\[
WeedsPrec(x \rightarrow y) = \frac{\sum_{c \in \vec{v}_x \cap \vec{v}_y} \vec{v}_x[c]}{\sum_{c \in \vec{v}_x} \vec{v}_x[c]}
\]

• **cosWeeds** (Lenci and Benotto, 2012) Geometric mean of cosine similarity and Weeds precision:
\[
\text{cosWeeds}(x \rightarrow y) = \sqrt{\cos(x, y) \cdot WeedsPrec(x \rightarrow y)}
\]

• **ClarkeDE** (Clarke, 2009) Computes degree of inclusion, by quantifying weighted coverage of the hyponym’s contexts by those of the hypernym:
\[
CDE(x \rightarrow y) = \frac{\sum_{c \in \vec{v}_x \cap \vec{v}_y} \min(\vec{v}_x[c], \vec{v}_y[c])}{\sum_{c \in \vec{v}_x} \vec{v}_x[c]}
\]

• **balAPinc** (Kotlerman et al., 2010) Balanced average precision inclusion.
\[
balAPinc(x \rightarrow y) = \frac{\sum_{r=1}^{N_y} [P(r) \cdot \text{rel}(c_r)]}{N_y}
\]

is an adaptation of the average precision measure from information retrieval for the inclusion hypothesis. \( N_y \) is the number of non-zero contexts of \( y \) and \( P(r) \) is the precision at rank \( r \), defined as the ratio of shared contexts with \( y \) among the top \( r \) contexts of \( x \). \( \text{rel}(c) \) is the relevance of a context \( c \), set to 0 if \( c \) is not a context of \( y \), and to \( 1 - \frac{\text{rank}_y(c)}{N_y + 1} \) otherwise, where \( \text{rank}_y(c) \) is the rank of the context \( c \) in \( y \)'s sorted vector. Finally,
\[
balAPinc(x \rightarrow y) = \sqrt{\text{Lin}(x, y) \cdot APinc(x \rightarrow y)}
\]

is the geometric mean of APinc and Lin similarity.

• **invCL** (Lenci and Benotto, 2012) Measures both distributional inclusion of \( x \) in \( y \) and distributional non-inclusion of \( y \) in \( x \):
\[
invCL(x \rightarrow y) = \sqrt{CDE(x \rightarrow y) \cdot (1 - CDE(y \rightarrow x))}
\]

3.3 Informativeness Measures

According to the *distributional informativeness hypothesis*, hypernyms tend to be less informative than hyponyms, as they are likely to occur in more general contexts than their hyponyms.

• **SLQS** (Santus et al., 2014)
\[
SLQS(x \rightarrow y) = 1 - \frac{E_x}{E_y}
\]

The informativeness of a word \( x \) is evaluated as the median entropy of its top \( N \) contexts: \( E_x = \text{median}_{i=1}^{N} (H(c_i)) \), where \( H(c) \) is the entropy of context \( c \).
• **SLQS Sub** A new variant of SLQS based on the assumption that if \( y \) is judged to be a hypernym of \( x \) to a certain extent, then \( x \) should be judged to be a hyponym of \( y \) to the same extent (which is not the case for regular SLQS). This is achieved by subtraction:

\[
SLQS_{\text{sub}}(x \rightarrow y) = E_y - E_x
\]

It is weakly symmetric in the sense that \( SLQS_{\text{sub}}(x \rightarrow y) = -SLQS_{\text{sub}}(y \rightarrow x) \).

SLQS and SLQS Sub have 3 hyper-parameters: i) the number of contexts \( N \); ii) whether to use median or average entropy among the top \( N \) contexts; and iii) the feature weighting used to sort the contexts by relevance (i.e., PPMI or PLMI).

• **SLQS Row** Differently from SLQS, SLQS Row computes the entropy of the target rather than the average/median entropy of the contexts, as an alternative way to compute the generality of a word. In addition, parallel to SLQS we tested SLQS Row with subtraction, **SLQS Row Sub**.

• **RCTC** (Rimell, 2014) Ratio of change in topic coherence:

\[
RCTC(x \rightarrow y) = \frac{TC(t_x)}{TC(t_y)} \frac{TC(t_{x\setminus y})}{TC(t_{y\setminus x})}
\]

where \( t_x \) are the top \( N \) contexts of \( x \), considered as \( x \)’s topic, and \( t_{x\setminus y} \) are the top \( N \) contexts of \( x \) which are not contexts of \( y \). \( TC(A) \) is the topic coherence of a set of words \( A \), defined as the median pairwise PMI scores between words in \( A \). \( N \) is a hyper-parameter. The measure is based on the assumptions that excluding \( y \)’s contexts from \( x \)’s increases the coherence of the topic, while excluding \( x \)’s contexts from \( y \)’s decreases the coherence of the topic. We include this measure under the informativeness inclusion, as it is based on a similar hypothesis.

### 3.4 Reversed Inclusion Measures

These measures are motivated by the fact that, even though—being more general—hypernyms are expected to occur in a larger set of contexts, sentences like “the vertebrate barks” or “the mammal arrested the thieves” are not common, since hyponyms are more specialized and are hence more appropriate in such contexts. On the other hand, hyponyms are likely to occur in broad contexts (e.g., *eat, live*), where hypernyms are also appropriate. In this sense, we can define the reversed inclusion hypothesis: “hyponym’s contexts are likely to be included in the hypernym’s contexts”. The following variants are tested for the first time.

• **Reversed Weeds**

\[
RevWeeds(x \rightarrow y) = Weeds(y \rightarrow x)
\]

• **Reversed ClarkeDE**

\[
RevCDE(x \rightarrow y) = CDE(y \rightarrow x)
\]

### 4 Datasets

We use four common semantic relation datasets: BLESS (Baroni and Lenci, 2011), EVALution (Santus et al., 2015), Lenci/Benotto (Benotto, 2015), and Weeds (Weeds et al., 2014). The datasets were constructed either using knowledge resources (e.g. WordNet, Wikipedia), crowdsourcing or both. The semantic relations and the size of each dataset are detailed in Table 1.

In our distributional semantic spaces, a target word is represented by the word and its POS tag. While BLESS and Lenci/Benotto contain this information, we needed to add POS tags to the other datasets. For each pair \((x, y)\), we considered 3 pairs \((x-p, y-p)\) for \( p \in \{\text{noun, adjective, verb}\} \), and added the respective pair to the dataset only if the words were present in the corpus.

<table>
<thead>
<tr>
<th>dataset</th>
<th>relations</th>
<th>#instances</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLESS</td>
<td>hypernym</td>
<td>1,337</td>
<td>26,554</td>
</tr>
<tr>
<td></td>
<td>meronym</td>
<td>2,943</td>
<td></td>
</tr>
<tr>
<td></td>
<td>coordination</td>
<td>3,565</td>
<td></td>
</tr>
<tr>
<td></td>
<td>event</td>
<td>3,824</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td>2,731</td>
<td></td>
</tr>
<tr>
<td></td>
<td>random-n</td>
<td>6,702</td>
<td></td>
</tr>
<tr>
<td></td>
<td>random-v</td>
<td>3,265</td>
<td></td>
</tr>
<tr>
<td></td>
<td>random-j</td>
<td>2,187</td>
<td></td>
</tr>
<tr>
<td>EVALution</td>
<td>hypernym</td>
<td>3,637</td>
<td>13,463</td>
</tr>
<tr>
<td></td>
<td>meronym</td>
<td>1,819</td>
<td></td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td>2,965</td>
<td></td>
</tr>
<tr>
<td></td>
<td>synonym</td>
<td>1,888</td>
<td></td>
</tr>
<tr>
<td>Lenci/Benotto</td>
<td>antonym</td>
<td>3,136</td>
<td></td>
</tr>
<tr>
<td></td>
<td>synonym</td>
<td>1,311</td>
<td>5,010</td>
</tr>
<tr>
<td></td>
<td>antonym</td>
<td>1,766</td>
<td></td>
</tr>
<tr>
<td>Weeds</td>
<td>hypernym</td>
<td>1,409</td>
<td>2,928</td>
</tr>
<tr>
<td></td>
<td>coordination</td>
<td>1,459</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1*: The semantic relations, number of instances in each relation, and size of each dataset.

---

2In our preliminary experiments, we noticed that the entropies of the targets and those of the contexts are not highly correlated, yielding a Spearman’s correlation of up to 0.448 for window based spaces, and up to 0.097 for the dependency-based ones (\( p < 0.01 \)).

3We removed the *entailment* relation, which had too few instances, and conflated relations to coarse-grained relations (e.g. *HasProperty* and *HasA* into *attribute*).

4Lenci/Benotto includes pairs to which more than one relation is assigned, e.g. when \( x \) or \( y \) are polysemous, and re-
We split each dataset randomly to 90% test and 10% validation. The validation sets are used to tune the hyper-parameters of several measures: SLQS (Sub), APSyn and RCTC.

5 Experiments

5.1 Comparing Unsupervised Measures

In order to evaluate the unsupervised measures described in Section 3, we compute the measure scores for each \((x, y)\) pair in each dataset. We first measure the method’s ability to discriminate hypernymy from all other relations in the dataset, i.e. by considering hypernyms as positive instances, and other word pairs as negative instances. In addition, we measure the method’s ability to discriminate hypernymy from every other relation in the dataset by considering one relation at a time. For a relation \(R\) we consider only \((x, y)\) pairs that are annotated as either hypernyms (positive instances) or \(R\) (negative instances). We rank the pairs according to the measure score and compute average precision (AP) at \(k = 100\) and \(k = all\).\(^5\)

\(^5\)We tried several cut-offs and chose the one that seemed to be more informative in distinguishing between the unsupervised measures.

Table 2 reports the best performing measure(s), with respect to \(AP@100\), for each relation in each dataset. The first observation is that there is no single combination of measure, context type and feature weighting that performs best in discriminating hypernymy from all other relations. In order to better understand the results, we focus on the second type of evaluation, in which we discriminate hypernyms from each other relation.

The results show preference to the syntactic context-types (\(dep\) and \(joint\)), which might be explained by the fact that these contexts are richer (as they contain both proximity and syntactic information) and therefore more discriminative. In feature weighting there is no consistency, but interestingly, raw frequency appears to be successful in hypernym detection, contrary to previously reported results for word similarity tasks, where PPMI was shown to outperform it (Bullinaria and Levy, 2007; Levy et al., 2015a).

The new SLQS variants are on top of the list in many settings. In particular they perform well in discriminating hypernyms from symmetric relations (antonymy, synonymy, coordination).

The measures based on the reversed inclusion hypothesis performed inconsistently, achieving perfect score in the discrimination of hypernyms from unrelated words, and performing well
Table 3: Intersection of datasets’ top-performing measures when discriminating between hypernymy and each other relation.

<table>
<thead>
<tr>
<th>relation</th>
<th>measure</th>
<th>context type</th>
<th>feature weighting</th>
</tr>
</thead>
<tbody>
<tr>
<td>meronym</td>
<td>cosWeeds</td>
<td>dep</td>
<td>ppmi</td>
</tr>
<tr>
<td></td>
<td>Weeds</td>
<td>dep/joint</td>
<td>ppmi</td>
</tr>
<tr>
<td></td>
<td>ClarkeDE</td>
<td>dep/joint</td>
<td>ppmi/freq</td>
</tr>
<tr>
<td>attribute</td>
<td>APSyn</td>
<td>joint</td>
<td>freq</td>
</tr>
<tr>
<td></td>
<td>cosine</td>
<td>joint</td>
<td>freq</td>
</tr>
<tr>
<td></td>
<td>Lin</td>
<td>dep</td>
<td>ppmi</td>
</tr>
<tr>
<td></td>
<td>cosine</td>
<td>dep</td>
<td>ppmi</td>
</tr>
<tr>
<td>antonym</td>
<td>SLQS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>synonym</td>
<td>SLQS_row/SLQS_row_sub</td>
<td>dep</td>
<td>ppmi</td>
</tr>
<tr>
<td></td>
<td>invCL</td>
<td>win2/5/5d</td>
<td>freq</td>
</tr>
</tbody>
</table>

in few other cases, always in combination with syntactic contexts.

Finally, the results show that there is no single combination of measure and parameters that performs consistently well for all datasets and classification tasks. In the following section we analyze the best combination of measure, context type and feature weighting to distinguish hypernymy from any other relation.

5.2 Best Measure Per Classification Task

We considered all relations that occurred in two datasets. For such relation, for each dataset, we ranked the measures by their AP@100 score, selecting those with score \( \geq 0.8 \). Table 3 displays the intersection of the datasets’ best measures.

**Hypernym vs. Meronym** The inclusion hypothesis seems to be most effective in discriminating between hypernyms and meronyms under syntactic contexts. We conjecture that the window-based contexts are less effective since they capture topical context words, that might be shared also among holonyms and their meronyms (e.g. car will occur with many of the neighbors of wheel). However, since meronyms and holonyms often have different functions, their functional contexts, which are expressed in the syntactic context-types, are less shared. This is where they mostly differ from hyponym-hypernym pairs, which are of the same function (e.g. cat is a type of animal).

Table 2 shows that SLQS performs well in this task on BLESS. This is contrary to previous findings that suggested that SLQS is weak in discriminating between hypernyms and meronyms, as in many cases the holonym is more general than the meronym (Shwartz et al., 2016).\(^7\) The surprising result could be explained by the nature of meronymy in this dataset: most holonyms in BLESS are rather specific words.

BLESS was built starting from 200 basic level concepts (e.g. goldfish) used as the \( x \) words, to which \( y \) words in different relations were associated (e.g. eye, for meronymy; animal, for hypernymy). \( x \) words represent hyponyms in the hyponym-hypernym pairs, and should therefore not be too general. Indeed, SLQS assigns high scores to hyponym-hypernym pairs. At the same time, in the meronym relation in BLESS, \( x \) is the holonym and \( y \) is the meronym. For consistency with EVALution, we switched those pairs in BLESS, placing the meronym in the \( x \) slot and the holonym in the \( y \) slot. As a consequence, after the switching, holonyms in BLESS are rather specific words.

In most cases, they are not more general than their meronyms (eye, goldfish), yielding low SLQS scores which are easy to separate from hypernyms. We note that this is a weakness of the BLESS dataset, rather than a strength of the measure. For instance, on EVALution, SLQS performs worse (ranked only as high as 13th), as this dataset has no such restriction on the basic level concepts, and may contain pairs like (eye, animal).

**Hypernym vs. Attribute** Symmetric similarity measures computed on syntactic contexts succeed to discriminate between hypernyms and attributes. Since attributes are syntactically different from hypernyms (in attributes, \( y \) is an adjective), it is unsurprising that they occur in different syntactic contexts, yielding low similarity scores.

\(^6\) We considered at least 10 measures, allowing scores slightly lower than 0.8 when others were unavailable.

\(^7\) In the hypernymy dataset of Shwartz et al. (2016), nearly 50\% of the SLQS false positive pairs were meronym-holonym pairs, in many of which the holonym is more general than the meronym by definition, e.g. (mauritius, africa).
<table>
<thead>
<tr>
<th>dataset</th>
<th>hyper vs. relation</th>
<th>best supervised</th>
<th>best unsupervised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>method</td>
<td>vectors</td>
<td>penalty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVALation</td>
<td>meronym</td>
<td>concat</td>
<td>dep-based</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td>concat</td>
<td>Glove-100</td>
</tr>
<tr>
<td></td>
<td>antonym</td>
<td>concat</td>
<td>dep-based</td>
</tr>
<tr>
<td></td>
<td>synonym</td>
<td>concat</td>
<td>dep-based</td>
</tr>
<tr>
<td>BLESS</td>
<td>meronym</td>
<td>concat</td>
<td>Glove-50</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td>concat</td>
<td>Glove-100</td>
</tr>
<tr>
<td></td>
<td>event</td>
<td>concat</td>
<td>Glove-100</td>
</tr>
<tr>
<td></td>
<td>random-n</td>
<td>concat</td>
<td>word2vec</td>
</tr>
<tr>
<td></td>
<td>random-j</td>
<td>concat</td>
<td>Glove-200</td>
</tr>
<tr>
<td></td>
<td>random-v</td>
<td>concat</td>
<td>word2vec</td>
</tr>
<tr>
<td></td>
<td>Lenci/ Benotto</td>
<td>antonym</td>
<td>concat</td>
</tr>
<tr>
<td></td>
<td>synonym</td>
<td>concat</td>
<td>Glove-300</td>
</tr>
<tr>
<td></td>
<td>Weeds</td>
<td>coord</td>
<td>concat</td>
</tr>
</tbody>
</table>

Table 4: Best performance on the validation set (10%) of each dataset for the supervised and unsupervised measures, in terms of Average Precision (AP) at $k = 100$, for hypernym vs. each single relation.

**Hypernym vs. Antonym** In all our experiments, antonyms were the hardest to distinguish from hypernyms, yielding the lowest performance. We found that SLQS performed reasonably well in this setting. However, the measure variations, context types and feature weightings were not consistent across datasets. SLQS relies on the assumption that $y$ is a more general word than $x$, which is not true for antonyms, making it the most suitable measure for this setting.

**Hypernym vs. Synonym** SLQS performs well also in discriminating between hypernyms and synonyms, in which $y$ is also not more general than $x$. We observed that in the joint context type, the difference in SLQS scores between synonyms and hypernyms was the largest. This may stem from the restrictiveness of this context type. For instance, among the most salient contexts we would expect to find informative contexts like *drinks milk* for *cat* and less informative ones like *drinks water for animal*, whereas the non-restrictive single dependency context *drinks* would probably be present for both.

Another measure that works well is invCL: interestingly, other inclusion-based measures assign high scores to $(x, y)$ when $y$ includes many of $x$’s contexts, which might be true also for synonyms (e.g. *elevator and lift* share many contexts). invCL, on the other hand, reduces with the ratio of $y$’s contexts included in $x$, yielding lower scores for synonyms.

**Hypernym vs. Coordination** We found no consistency among BLESS and Weeds. On Weeds, inclusion-based measures (ClarkeDE, invCL and Weeds) showed the best results. The best performing measures on BLESS, however, were variants of SLQS, that showed to perform well in cases where the negative relation is symmetric (antonym, synonym and coordination). The difference could be explained by the nature of the datasets: the BLESS test set contains 1,185 hypernymy pairs, with only 129 distinct $ys$, many of which are general words like *animal* and *object*. The Weeds test set, on the other hand, was intentionally constructed to contain an overall unique $y$ in each pair, and therefore contains much more specific $ys$ (e.g. *(quirk, strangeness)*). For this reason, generality-based measures perform well on BLESS, and struggle with Weeds, which is handled better using inclusion-based measures.

### 5.3 Comparison to State-of-the-art

**Supervised Methods**

For comparison with the state-of-the-art, we evaluated several supervised hypernymy detection methods, based on the word embeddings of $x$ and $y$: concatenation $\vec{v}_x \oplus \vec{v}_y$ (Baroni et al., 2012), difference $\vec{v}_y - \vec{v}_x$ (Weeds et al., 2014), and ASYM (Roller et al., 2014). We downloaded several pre-trained embeddings (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014), and trained a logistic regression classifier to predict hypernymy. We used the 90% portion (originally the test set) as the train set, and the other 10% (originally the validation set) as a test set, reporting the best results among different vectors,
method and regularization factor.\(^8\)

Table 4 displays the performance of the best classifier on each dataset, in a hypernym vs. a single relation setting. We also re-evaluated the unsupervised measures, this time reporting the results on the validation set (10%) for comparison.

The overall performance of the embedding-based classifiers is almost perfect, and in particular the best performance is achieved using the concatenation method (Baroni et al., 2012) with either GloVe (Pennington et al., 2014) or the dependency-based embeddings (Levy and Goldberg, 2014). As expected, the unsupervised measures perform worse than the embedding-based classifiers, though generally not bad on their own.

These results may suggest that unsupervised methods should be preferred only when no training data is available, leaving all the other cases to supervised methods. This is, however, not completely true. As others previously noticed, supervised methods do not actually learn the relation between \(x\) and \(y\), but rather separate properties of either \(x\) or \(y\). Levy et al. (2015b) named this the “lexical memorization” effect, i.e. memorizing that certain \(y\)'s tend to appear in many positive pairs (prototypical hypernms).

On that account, the Weeds dataset has been designed to avoid such memorization, with every word occurring once in each slot of the relation. While the performance of the supervised methods on this dataset is substantially lower than their performance on other datasets, it is yet well above the random baseline which we might expect from a method that can only memorize words it has seen during training.\(^9\) This is an indication that supervised methods can abstract away from the words. Indeed, when we repeated the experiment with a lexical split of each dataset, i.e., such that the train and test set consist of distinct vocabularies, we found that the supervised methods’ performance did not decrease dramatically, in contrast to the findings of Levy et al. (2015b). The large performance gaps reported by Levy et al. (2015b) might be attributed to the size of their training sets. Their lexical split discarded around half of the pairs in the dataset and split the rest of the pairs equally to train and test, resulting in a relatively small train set. We performed the split such that only around 30% of the pairs in each dataset were discarded, and split the train and test sets with a ratio of roughly 90/10%, obtaining large enough train sets.

Our experiment suggests that rather than memorizing the verbatim prototypical hypernms, the supervised models might learn that certain regions in the vector space pertain to prototypical hypernms. For example, \textit{device} (from the BLESS train set) and \textit{appliance} (from the BLESS test set) are two similar words, which are both prototypical hypernms. Another interesting observation was recently made by Roller and Erk (2016): they showed that when dependency-based embeddings are used, supervised distributional methods trace \(x\) and \(y\)’s separate occurrences in different slots of Hearst patterns (Hearst, 1992).

Whether supervised methods only memorize or also learn, it is more consensual that they lack the ability to capture the relation between \(x\) and \(y\), and that they rather indicate how likely \(y (x)\) is to be a hypernym (hyponym) (Levy et al., 2015b; Santus et al., 2016a; Shwartz et al., 2016; Roller and Erk, 2016). While this information is valuable, it cannot be solely relied upon for classification.

To better understand the extent of this limitation, we conducted an experiment in a similar manner to the switched hypernym pairs in Santus et al. (2016a). We used BLESS, which is the only dataset with random pairs. For each hypernym pair \((x_1, y_1)\), we sampled a word \(y_2\) that participates in another hypernym pair \((x_2, y_2)\), such that \((x_1, y_2)\) is not in the dataset, and added \((x_1, y_2)\) as a random pair. We added 139 new pairs to the validation set, such as (\textit{rifle, animal}) and (\textit{salmon, weapon}). We then used the best supervised and unsupervised methods for hypernym vs. random on BLESS to re-classify the revised validation set. Table 5 displays the experiment results.

<table>
<thead>
<tr>
<th>method</th>
<th>AP@100 original</th>
<th>AP@100 switched</th>
<th>∆</th>
</tr>
</thead>
<tbody>
<tr>
<td>supervised</td>
<td>concat, word2vec, L1</td>
<td>0.995</td>
<td>0.575</td>
</tr>
<tr>
<td>unsupervised</td>
<td>cosWeeds, win2d, ppmi</td>
<td>0.818</td>
<td>0.882</td>
</tr>
</tbody>
</table>

Table 5: Average Precision (AP) at \(k = 100\) of the best supervised and unsupervised methods for hypernym vs. random-n, on the original BLESS validation set and the validation set with the artificially added switched hypernym pairs.
The switched hypernym experiment paints a much less optimistic picture of the embeddings’ actual performance, with a drop of 42 points in average precision. 121 out of the 139 switched hypernym pairs were falsely classified as hypernyms. Examining the $y$ words of these pairs reveals general words that appear in many hypernym pairs (e.g. animal, object, vehicle). The unsupervised measure was not similarly affected by the switched pairs, and the performance even slightly increased. This result is not surprising, since most unsupervised measures aim to capture aspects of the relation between $x$ and $y$, while not relying on information about one of the words in the pair.\textsuperscript{10}

6 Discussion

The results in Section 5 suggest that a supervised method using the unsupervised measures as features could possibly be the best of both worlds. We would expect it to be more robust than embedding-based methods on the one hand, while being more informative than any single unsupervised measure on the other hand.

Such a method was developed by Santus et al. (2016a), however using mostly features that describe a single word, e.g. frequency and entropy. It was shown to be competitive with the state-of-the-art supervised methods. With that said, it was also shown to be sensitive to the distribution of training examples in a specific dataset, like the embedding-based methods.

We conducted a similar experiment, with a much larger number of unsupervised features, namely the various measure scores, and encountered the same issue. While the performance was good, it dropped dramatically when the model was tested on a different test set.

We conjecture that the problem stems from the currently available datasets, which are all somewhat artificial and biased. Supervised methods which are strongly based on the relation between the words, e.g. those that rely on path-based information (Shwartz et al., 2016), manage to overcome the bias. Distributional methods, on the other hand, are based on a weaker notion of the relation between words, hence are more prone to overfit the distribution of training instances in a specific dataset. In the future, we hope that new datasets will be available for the task, which would be drawn from corpora and will reflect more realistic distributions of words and semantic relations.

7 Conclusion

We performed an extensive evaluation of unsupervised methods for discriminating hypernyms from other semantic relations. We found that there is no single combination of measure and parameters which is always preferred; however, we suggested a principled linguistic-based analysis of the most suitable measure for each task that yields consistent performance across different datasets.

We investigated several new variants of existing methods, and found that some variants of SLQS turned out to be superior on certain tasks. In addition, we have tested for the first time the joint context type (Chersoni et al., 2016), which was found to be very discriminative, and might hopefully benefit other semantic tasks.

For comparison, we evaluated the state-of-the-art supervised methods on the datasets, and they have shown to outperform the unsupervised ones, while also being efficient and easier to use. However, a deeper analysis of their performance demonstrated that, as previously suggested, these methods do not capture the relation between $x$ and $y$, but rather indicate the “prior probability” of either word to be a hyponym or a hypernym. As a consequence, supervised methods are sensitive to the distribution of examples in a particular dataset, making them less reliable for real-world applications. Being motivated by linguistic hypotheses, and independent from training data, unsupervised measures were shown to be more robust. In this sense, unsupervised methods can still play a relevant role, especially if combined with supervised methods, in the decision whether the relation holds or not.

Acknowledgments

The authors would like to thank Ido Dagan, Alessandro Lenci, and Yuji Matsumoto for their help and advice. Vered Shwartz is partially supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1). Enrico Santus is partially supported by HK PhD Fellowship Scheme under PF12-13656.
References


the Twenty-Second International Joint Conference on Artificial Intelligence, volume 11, pages 1872–1877.


Chapter 4

Improving Hypernymy Detection with an Integrated Path-based and Distributional Method
Improving Hypernymy Detection with an Integrated Path-based and Distributional Method

Vered Shwartz  Yoav Goldberg  Ido Dagan
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
vered1986@gmail.com yoav.goldberg@gmail.com dagan@cs.biu.ac.il

Abstract

Detecting hypernymy relations is a key task in NLP, which is addressed in the literature using two complementary approaches. Distributional methods, whose supervised variants are the current best performers, and path-based methods, which received less research attention. We suggest an improved path-based algorithm, in which the dependency paths are encoded using a recurrent neural network, that achieves results comparable to distributional methods. We then extend the approach to integrate both path-based and distributional signals, significantly improving upon the state-of-the-art on this task.

1 Introduction

Hypernymy is an important lexical-semantic relation for NLP tasks. For instance, knowing that Tom Cruise is an actor can help a question answering system answer the question “which actors are involved in Scientology?”. While semantic taxonomies, like WordNet (Fellbaum, 1998), define hypernymy relations between word types, they are limited in scope and domain. Therefore, automated methods have been developed to determine, for a given term-pair \((x, y)\), whether \(y\) is an hypernym of \(x\), based on their occurrences in a large corpus.

For a couple of decades, this task has been addressed by two types of approaches: distributional, and path-based. In distributional methods, the decision whether \(y\) is a hypernym of \(x\) is based on the distributional representations of these terms. Lately, with the popularity of word embeddings (Mikolov et al., 2013), most focus has shifted towards supervised distributional methods, in which each \((x, y)\) term-pair is represented using some combination of the terms’ embedding vectors.

In contrast to distributional methods, in which the decision is based on the separate contexts of \(x\) and \(y\), path-based methods base the decision on the lexico-syntactic paths connecting the joint occurrences of \(x\) and \(y\) in a corpus. Hearst (1992) identified a small set of frequent paths that indicate hypernymy, e.g. \(Y\) such as \(X\). Snow et al. (2004) represented each \((x, y)\) term-pair as the multiset of dependency paths connecting their co-occurrences in a corpus, and trained a classifier to predict hypernymy, based on these features.

Using individual paths as features results in a huge, sparse feature space. While some paths are rare, they often consist of certain unimportant components. For instance, “Spelt is a species of wheat” and “Fantasy is a genre of fiction” yield two different paths: \(X\) be species of \(Y\) and \(X\) be genre of \(Y\), while both indicating that \(X\) is-a \(Y\). A possible solution is to generalize paths by replacing words along the path with their part-of-speech tags or with wild cards, as done in the PATTY system (Nakashole et al., 2012).

Overall, the state-of-the-art path-based methods perform worse than the distributional ones. This stems from a major limitation of path-based methods: they require that the terms of the pair occur together in the corpus, limiting the recall of these methods. While distributional methods have no such requirement, they are usually less precise in detecting a specific semantic relation like hypernymy, and perform best on detecting broad semantic similarity between terms. Though these approaches seem complementary, there has been rather little work on integrating them (Mirkin et al., 2006; Kaji and Kitsuregawa, 2008).

In this paper, we present HypeNET, an integrated path-based and distributional method for hypernymy detection. Inspired by recent progress
in relation classification, we use a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to encode dependency paths. In order to create enough training data for our network, we followed previous methodology of constructing a dataset based on knowledge resources.

We first show that our path-based approach, on its own, substantially improves performance over prior path-based methods, yielding performance comparable to state-of-the-art distributional methods. Our analysis suggests that the neural path representation enables better generalizations. While coarse-grained generalizations, such as replacing a word by its POS tag, capture mostly syntactic similarities between paths, HypeNET captures also semantic similarities.

We then show that we can easily integrate distributional signals in the network. The integration results confirm that the distributional and path-based signals indeed provide complementary information, with the combined model yielding an improvement of up to 14 $F_1$ points over each individual model.  

### 2 Background

We introduce the two main approaches for hypernymy detection: distributional (Section 2.1), and path-based (Section 2.2). We then discuss the recent use of recurrent neural networks in the related task of relation classification (Section 2.3).

#### 2.1 Distributional Methods

Hypernymy detection is commonly addressed using distributional methods. In these methods, the decision whether $y$ is a hypernym of $x$ is based on the distributional representations of the two terms, i.e., the contexts with which each term occurs separately in the corpus.

Earlier methods developed unsupervised measures for hypernymy, starting with symmetric similarity measures (Lin, 1998), and followed by directional measures based on the distributional inclusion hypothesis (Weeds and Weir, 2003; Kotlerman et al., 2010). This hypothesis states that the contexts of a hyponym are expected to be largely included in those of its hypernym. More recent work (Santus et al., 2014; Rimell, 2014) introduce new measures, based on the assumption that the most typical linguistic contexts of a hypernym are less informative than those of its hyponyms.

More recently, the focus of the distributional approach shifted to supervised methods. In these methods, the $(x, y)$ term-pair is represented by a feature vector, and a classifier is trained on these vectors to predict hypernymy. Several methods are used to represent term-pairs as a combination of each term’s embeddings vector: concatenation $\vec{x} \oplus \vec{y}$ (Baroni et al., 2012), difference $\vec{y} - \vec{x}$ (Roller et al., 2014; Weeds et al., 2014), and dot-product $\vec{x} \cdot \vec{y}$. Using neural word embeddings (Mikolov et al., 2013; Pennington et al., 2014), these methods are easy to apply, and show good results (Baroni et al., 2012; Roller et al., 2014).

#### 2.2 Path-based Methods

A different approach to detecting hypernymy between a pair of terms $(x, y)$ considers the lexico-syntactic paths that connect the joint occurrences of $x$ and $y$ in a large corpus. Automatic acquisition of hypernyms from free text, based on such paths, was first proposed by Hearst (1992), who identified a small set of lexico-syntactic paths that indicate hypernymy relations (e.g. $Y$ such as $X$, $X$ and other $Y$).

In a later work, Snow et al. (2004) learned to detect hypernymy. Rather than searching for specific paths that indicate hypernymy, they represent each $(x, y)$ term-pair as the multiset of all dependency paths that connect $x$ and $y$ in the corpus, and train a logistic regression classifier to predict whether $y$ is a hypernym of $x$, based on these paths.

Paths that indicate hypernymy are those that were assigned high weights by the classifier. The paths identified by this method were shown to subsume those found by Hearst (1992), yielding improved performance. Variations of Snow et al.’s (2004) method were later used in tasks such as taxonomy construction (Snow et al., 2006; Kozareva and Hovy, 2010; Carlson et al., 2010; Riedel et al., 2013), analogy identification (Turney, 2006), and definition extraction (Borg et al., 2009;Navigli and Velardi, 2010).

A major limitation in relying on lexico-syntactic paths is the sparsity of the feature space. Since similar paths may somewhat vary at the lexical level, generalizing such variations into more abstract paths can increase recall. The PATTY algorithm (Nakashole et al., 2012) applied such generalizations for the purpose of acquiring a taxon-

---

1Our code and data are available in: https://github.com/vered1986/HypeNET
2.3 RNNs for Relation Classification

Relation classification is a related task whose goal is to classify the relation that is expressed between two target terms in a given sentence to one of pre-defined relation classes. To illustrate, consider the following sentence, from the SemEval-2010 relation classification task dataset (Hendrickx et al., 2009): “The apples are in the basket.” Here, the relation expressed between the target entities is Content − Container(e1, e2).

The shortest dependency paths between the target entities were shown to be informative for this task (Fundel et al., 2007). Recently, deep learning techniques showed good performance in capturing the indicative information in such paths.

In particular, several papers show improved performance using recurrent neural networks (RNN) that process a dependency path edge-by-edge. Xu et al. (2015; 2016) apply a separate long short-term memory (LSTM) network to each sequence of words, POS tags, dependency labels and WordNet hypernyms along the path. A max-pooling layer on the LSTM outputs is used as the input of a network that predicts the classification. Other papers suggest incorporating additional network architectures to further improve performance (Nguyen and Grishman, 2015; Liu et al., 2015).

While relation classification and hypernymy detection are both concerned with identifying semantic relations that hold for pairs of terms, they differ in a major respect. In relation classification the relation should be expressed in the given text, while in hypernymy detection, the goal is to recognize a generic lexical-semantic relation between terms that holds in many contexts. Accordingly, in relation classification a term-pair is represented by a single dependency path, while in hypernymy detection it is represented by the multiset of all dependency paths in which they co-occur in the corpus.

3 LSTM-based Hypernymy Detection

We present HypeNET, an LSTM-based method for hypernymy detection. We first focus on improving path representation (Section 3.1), and then integrate distributional signals into our network, resulting in a combined method (Section 3.2).

3.1 Path-based Network

Similarly to prior work, we represent each dependency path as a sequence of edges that leads from x to y in the dependency tree. Each edge contains the lemma and part-of-speech tag of the source node, the dependency label, and the edge direction between two subsequent nodes. We denote each edge as lemma/POS/dep/dir. See figure 1 for an illustration.

Rather than treating an entire dependency path as a single feature, we encode the sequence of edges using a long short-term memory (LSTM) network. The vectors obtained for the different paths of a given (x, y) pair are pooled, and the resulting vector is used for classification. Figure 2 depicts the overall network structure, which is described below.

Edge Representation We represent each edge by the concatenation of its components’ vectors:

\[ v_e = [v_1, v_{pos}, v_{dep}, v_{dir}] \]

where \( v_1, v_{pos}, v_{dep}, v_{dir} \) represent the embedding vectors of the lemma, part-of-speech, dependency label and dependency direction (along the path from x to y), respectively.

Path Representation For a path \( p \) composed of edges \( e_1, ..., e_k \), the edge vectors \( v_{e_1}, ..., v_{e_k} \) are fed in order to an LSTM encoder, resulting in a vector \( \tilde{o}_p \) representing the entire path \( p \). The LSTM architecture is effective at capturing temporal patterns in sequences. We expect the training procedure to drive the LSTM encoder to focus on parts of the path that are more informative for the classification task while ignoring others.

\(^2\)Like Snow et al. (2004), we added for each path, additional paths containing single daughters of \( x \) or \( y \) not already contained in the path, referred by Snow et al. (2004) as “satellite edges”. This enables including paths like Such Y as X, in which the word “such” is not in the path between \( x \) and \( y \).
Term-Pair Classification Each \((x, y)\) term-pair is represented by the multisets of lexico-syntactic paths that connected \(x\) and \(y\) in the corpus, denoted as \(\text{paths}(x, y)\), while the supervision is given for the term pairs. We represent each \((x, y)\) term-pair as the weighted-average of its path vectors, by applying average pooling on its path vectors, as follows:

\[
v_{xy}^\ast = \overline{v}_{\text{paths}(x,y)} = \frac{\sum_{p \in \text{paths}(x,y)} f_p(x,y) \cdot \overline{v}_p}{\sum_{p \in \text{paths}(x,y)} f_p(x,y)} \tag{1}
\]

where \(f_p(x,y)\) is the frequency of \(p\) in \(\text{paths}(x, y)\). We then feed this path vector to a single-layer network that performs binary classification to decide whether \(y\) is a hypernym of \(x\).

\[
c = \text{softmax}(W \cdot v_{xy}^\ast) \tag{2}
\]

c is a 2-dimensional vector whose components sum to 1, and we classify a pair as positive if \(c[1] > 0.5\).

Implementation Details To train the network, we used PyCNN.\(^3\) We minimize the cross entropy loss using gradient-based optimization, with mini-batches of size 10 and the Adam update rule (Kingma and Ba, 2014). Regularization is applied by a dropout on each of the components’ embeddings. We tuned the hyper-parameters (learning rate and dropout rate) on the validation set (see the appendix for the hyper-parameters values).

We initialized the lemma embeddings with the pre-trained GloVe word embeddings (Pennington et al., 2014), trained on Wikipedia. We tried both the 50-dimensional and 100-dimensional embedding vectors and selected the ones that yield better performance on the validation set.\(^4\) The other embeddings, as well as out-of-vocabulary lemmas, are initialized randomly. We update all embedding vectors during training.

3.2 Integrated Network

The network presented in Section 3.1 classifies each \((x, y)\) term-pair based on the paths that connect \(x\) and \(y\) in the corpus. Our goal was to improve upon previous path-based methods for hypernymy detection, and we show in Section 6 that our network indeed outperforms them. Yet, as path-based and distributional methods are considered complementary, we present a simple way to integrate distributional features in the network, yielding improved performance.

We extended the network to take into account distributional information on each term. Inspired by the supervised distributional concatenation method (Baroni et al., 2012), we simply concatenate \(x\) and \(y\) word embeddings to the \((x, y)\) feature vector, redefining \(v_{xy}^\ast\):

\[
v_{xy} = [v_{wx}^\ast, \overline{v}_{\text{paths}(x,y)}, v_{wy}^\ast] \tag{3}
\]

where \(v_{wx}^\ast\) and \(v_{wy}^\ast\) are \(x\) and \(y\)'s word embeddings, respectively, and \(\overline{v}_{\text{paths}(x,y)}\) is the averaged path vector defined in equation 1. This way, each \((x, y)\) pair is represented using both the distributional features of \(x\) and \(y\), and their path-based features.

\(^4\)Higher-dimensional embeddings seem not to improve performance, while hurting the training runtime.
4 Dataset

4.1 Creating Instances

Neural networks typically require a large amount of training data, whereas the existing hypernymy datasets, like BLESS (Baroni and Lenci, 2011), are relatively small. Therefore, we followed the common methodology of creating a dataset using distant supervision from knowledge resources (Snow et al., 2004; Riedel et al., 2013). Following Snow et al. (2004), who constructed their dataset based on WordNet hypernymy, and aiming to create a larger dataset, we extract hypernymy relations from several resources: WordNet (Fellbaum, 1998), DBPedia (Auer et al., 2007), Wikidata (Vrandečić, 2012) and Yago (Suchanek et al., 2007).

All instances in our dataset, both positive and negative, are pairs of terms that are directly related in at least one of the resources. These resources contain thousands of relations, some of which indicate hypernymy with varying degrees of certainty. To avoid including questionable relation types, we consider as denoting positive examples only indisputable hypernymy relations (Table 1), which we manually selected from the set of hypernymy indicating relations in Shwartz et al. (2015).

Term-pairs related by other relations (including hyponymy), are considered as negative instances. Using related rather than random term-pairs as negative instances tests our method’s ability to distinguish between hypernymy and other kinds of semantic relatedness. We maintain a ratio of 1:4 positive to negative pairs in the dataset.

Like Snow et al. (2004), we include only term-pairs that have joint occurrences in the corpus, requiring at least two different dependency paths for each pair.

4.2 Random and Lexical Dataset Splits

As our primary dataset, we perform standard random splitting, with 70% train, 25% test and 5% validation sets.

As pointed out by Levy et al. (2015), supervised distributional lexical inference methods tend to perform “lexical memorization”, i.e., instead of learning a relation between the two terms, they mostly learn an independent property of a single term in the pair: whether it is a “prototypical hypernym” or not. For instance, if the training set contains term-pairs such as (dog, animal), (cat, animal), and (cow, animal), all annotated as positive examples, the algorithm may learn that animal is a prototypical hypernym, classifying any new (x, animal) pair as positive, regardless of the relation between x and animal. Levy et al. (2015) suggested to split the train and test sets such that each will contain a distinct vocabulary (“lexical split”), in order to prevent the model from overfitting by lexical memorization.

To investigate such behaviors, we present results also for a lexical split of our dataset. In this case, we split the train, test and validation sets such that each contains a distinct vocabulary. We note that this differs from Levy et al. (2015), who split only the train and the test sets, and dedicated a subset of the train for validation. We chose to deviate from Levy et al. (2015) because we noticed that when the validation set contains terms from the train set, the model is rewarded for lexical memorization when tuning the hyper-parameters, consequently yielding suboptimal performance on the lexically-distinct test set. When each set has a distinct vocabulary, the hyper-parameters are tuned to avoid lexical memorization and are likely to perform better on the test set. We tried to keep roughly the same 70/25/5 ratio in our lexical split. The sizes of the two datasets are shown in Table 2.

Indeed, training a model on a lexically split dataset may result in a more general model, that can better handle pairs consisting of two unseen terms during inference. However, we argue that in the common applied scenario, the inference involves an unseen pair (x, y), in which x and/or y have already been observed separately. Models trained on a random split may introduce the model with a term’s “prior probability” of being a hypernym or a hyponym, and this information can be exploited beneficially at inference time.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
resource & relations & random split & lexical split \\
\hline
WordNet & instance hypernym, hypernym & 49,475 & 20,335 \\
DBPedia & type & 17,670 & 6,670 \\
Wikidata & subclass of, instance of & 3,534 & 1,350 \\
Yago & subclass of & 70,679 & 28,295 \\
\hline
\end{tabular}
\caption{The number of instances in each dataset.}
\end{table}
5 Baselines

We compare HypeNET with several state-of-the-art methods for hypernymy detection, as described in Section 2: path-based methods (Section 5.1), and distributional methods (Section 5.2). Due to different works using different datasets and corpora, we replicated the baselines rather than comparing to the reported results.

We use the Wikipedia dump from May 2015 as the underlying corpus of all the methods, and parse it using spaCy.6 We perform model selection on the validation set to tune the hyper-parameters of each method.7 The best hyper-parameters are reported in the appendix.

5.1 Path-based Methods

Snow We follow the original paper, and extract all shortest paths of four edges or less between terms in a dependency tree. Like Snow et al. (2004), we add paths with “satellite edges”, i.e., single words not already contained in the dependency path, which are connected to either X or Y, allowing paths like such Y as X. The number of distinct paths was 324,578. We apply χ² feature selection to keep only the 100,000 most informative paths and train a logistic regression classifier.

Generalization We also compare our method to a baseline that uses generalized dependency paths. Following PATTY’s approach to generalizing paths (Nakashole et al., 2012), we replace edges with their part-of-speech tags as well as with wild cards. We generate the powerset of all possible generalizations, including the original paths. See Table 3 for examples. The number of features after generalization went up to 2,093,220. Similarly to the first baseline, we apply feature selection, this time keeping the 1,000,000 most informative paths, and train a logistic regression classifier over the generalized paths.8

5.2 Distributional Methods

Unsupervised SLQS (Santus et al., 2014) is an entropy-based measure for hypernymy detection, reported to outperform previous state-of-the-art unsupervised methods (Weeds and Weir, 2003; Kotlerman et al., 2010). The original paper was evaluated on the BLESS dataset (Baroni and Lenci, 2011), which consists of mostly frequent words. Applying the vanilla settings of SLQS on our dataset, that contains also rare terms, resulted in low performance. Therefore, we received assistance from Enrico Santus, who kindly provided the results of SLQS on our dataset after tuning the system as follows.

The validation set was used to tune the threshold for classifying a pair as positive, as well as the maximum number of each term’s most associated contexts (N). In contrast to the original paper, in which the number of each term’s contexts is fixed to N, in this adaptation it was set to the minimum between the number of contexts with LMI score above zero and N. In addition, the SLQS scores were not multiplied by the cosine similarity scores between terms, and terms were lemmatized prior to computing the SLQS scores, significantly improving recall.

As our results suggest, while this method is state-of-the-art for unsupervised hypernymy detection, it is basically designed for classifying specificity level of related terms, rather than hypernymy in particular.

Supervised To represent term-pairs with distributional features, we tried several state-of-the-art methods: concatenation $\vec{x} \oplus \vec{y}$ (Baroni et al., 2012), difference $\vec{y} - \vec{x}$ (Roller et al., 2014; Weeds et al., 2014), and dot-product $\vec{x} \cdot \vec{y}$. We downloaded several pre-trained embeddings (Mikolov et al., 2013; Pennington et al., 2014) of different sizes, and trained a number of classifiers: logistic regression, SVM, and SVM with RBF kernel, which was reported by Levy et al. (2015) to perform best in this setting. We perform model selection on the validation set to select the best vectors, method and regularization factor (see the appendix).

---

Table 3: Example generalizations of X was established as Y.

| X/NOUN/dobj/> establish/VERB/ROOT/- as/ADP/prep/< Y/NOUN/pobj/< |
| X/NOUN/dobj/> establish/VERB/ROOT/- * Y/NOUN/pobj/< |
| X/NOUN/dobj/> VERB as/ADP/prep/< Y/NOUN/pobj/< |
| X/NOUN/dobj/> * as/ADP/prep/< Y/NOUN/pobj/< |
| X/NOUN/dobj/> establish/VERB/ROOT/- * Y/NOUN/pobj/< |
| X/NOUN/dobj/> establish/VERB/ROOT/- as/ADP/prep/< Y/NOUN/pobj/< |

---

6https://spacy.io/
7We applied grid search for a range of values, and picked the ones that yield the highest F₁ score on the validation set.
8We also tried keeping the 100,000 most informative paths, but the performance was worse.
Table 4: Performance scores of our method compared to the path-based baselines and the state-of-the-art distributional methods for hypernymy detection, on both variations of the dataset – with lexical and random split to train / test / validation.

<table>
<thead>
<tr>
<th>method</th>
<th>random split</th>
<th>lexical split</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>precision</td>
<td>recall</td>
</tr>
<tr>
<td>Path-based</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Snow</td>
<td>0.843</td>
<td>0.452</td>
</tr>
<tr>
<td>Snow + Gen</td>
<td>0.852</td>
<td>0.561</td>
</tr>
<tr>
<td>HypeNET Path-based</td>
<td>0.811</td>
<td>0.716</td>
</tr>
<tr>
<td>Distributional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLQS (Santus et al., 2014)</td>
<td>0.491</td>
<td>0.737</td>
</tr>
<tr>
<td>Best supervised (concatenation)</td>
<td>0.901</td>
<td>0.637</td>
</tr>
<tr>
<td>Combined</td>
<td>0.913</td>
<td>0.890</td>
</tr>
</tbody>
</table>

6 Results

Table 4 displays performance scores of HypeNET and the baselines. HypeNET Path-based is our path-based recurrent neural network model (Section 3.1) and HypeNET Integrated is our combined method (Section 3.2). Comparing the path-based methods shows that generalizing paths improves recall while maintaining similar levels of precision, reassessing the behavior found in Nakashole et al. (2012). HypeNET Path-based outperforms both path-based baselines by a significant improvement in recall and with slightly lower precision. The recall boost is due to better path generalization, as demonstrated in Section 7.1.

Regarding distributional methods, the unsupervised SLQS baseline performed slightly worse on our dataset. The low precision stems from its inability to distinguish between hypernyms and meronyms, which are common in our dataset, causing many false positive pairs such as (zabrze, poland) and (kibbutz, israel). We sampled 50 false positive pairs of each dataset split, and found that 38% of the false positive pairs in the random split and 48% of those in the lexical split were holonym-meronym pairs.

In accordance with previously reported results, the supervised embedding-based method is the best performing baseline on our dataset as well. HypeNET Path-based performs slightly better, achieving state-of-the-art results. Adding distributional features to our method shows that these two approaches are indeed complementary. On both dataset splits, the performance differences between HypeNET Integrated and HypeNET Path-based, as well as the supervised distributional method, are substantial, and statistically significant with p-value of 1% (paired t-test).

We also reassess that indeed supervised distributional methods perform worse on a lexical split (Levy et al., 2015). We further observe a similar reduction when using HypeNET, which is not a result of lexical memorization, but rather stems from over-generalization (Section 7.1).

7 Analysis

7.1 Qualitative Analysis of Learned Paths

We analyze HypeNET’s ability to generalize over path structures, by comparing prominent indicative paths which were learned by each of the path-based methods. We do so by finding high-scoring paths that contributed to the classification of true-positive pairs in the dataset. In the path-based baselines, these are the highest-weighted features as learned by the logistic regression classifier. In the LSTM-based method, it is less straightforward to identify the most indicative paths. We assess the contribution of a certain path \( p \) to classification by regarding it as the only path that appeared for the term-pair, and compute its TRUE label score from the class distribution: \( \text{softmax}(W \cdot v_{xy})[1] \), setting \( v_{xy} = [0, e_{p}, 0] \).

A notable pattern is that Snow’s method learns specific paths, like \( X \) is \( Y \) from (e.g. Megadeth is an American thrash metal band from Los Angeles). While Snow’s method can only rely on verbatim paths, limiting its recall, the generalized version of Snow often makes coarse generalizations, such as \( X \) VERB \( Y \) from. Clearly, such a path is too general, and almost any verb assigned to it results in a non-indicative path (e.g. \( X \) take \( Y \) from). Efforts by the learning method to avoid such generalization, again, lower the recall. HypeNET provides a better midpoint, making fine-grained generalizations by learning additional semantically similar paths such as \( X \) become \( Y \) from and \( X \) remain \( Y \) from. See table 5 for additional example paths which illustrate these behaviors.

We also noticed that while on the random split our model learns a range of specific paths such as \( X \) is \( Y \) published (learned for e.g. \( Y=\text{magazine} \) and \( X \) is \( Y \) produced (\( Y=\text{film} \), in the lexical split it only learns the general \( X \) is \( Y \) path for these re-
Eyeball is a 1975 Italian-Spanish film directed by Umberto Lenzi

Allure is a U.S. women’s beauty magazine published monthly

Calico Light Weapons Inc. (CLWS) is an American privately held manufacturing company based in Cornelius, Oregon

Snow is a 1923 American comedy film directed by Edward Sedgwick

Retailix Ltd. is a software company

Table 5: Examples of indicative paths learned by each method, with corresponding true positive term-pairs from the random split test set. Hypernyms are marked red and hyponyms are marked blue.

### Table 6: Distribution of relations holding between each pair of terms in the resources among false positive pairs.

<table>
<thead>
<tr>
<th>Relation</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>synonymy</td>
<td>21.37%</td>
</tr>
<tr>
<td>hyponymy</td>
<td>29.45%</td>
</tr>
<tr>
<td>holonymy / meronymy</td>
<td>9.36%</td>
</tr>
<tr>
<td>hypernymy-like relations</td>
<td>21.03%</td>
</tr>
<tr>
<td>other relations</td>
<td>18.77%</td>
</tr>
</tbody>
</table>

7.2 Error Analysis

False Positives  We categorized the false positive pairs on the random split according to the relation holding between each pair of terms in the resources used to construct the dataset. We grouped several semantic relations from different resources to broad categories, e.g. synonymy includes also alias and Wikipedia redirection. Table 6 displays the distribution of semantic relations among false positive pairs.

More than 20% of the errors stem from confusing synonymy with hypernymy, which are known to be difficult to distinguish.

An additional 30% of the term-pairs are reversed hypernym-hyponym pairs ($y$ is a hyponym of $x$). Examining a sample of these pairs suggests that they are usually near-synonyms, i.e., it is not that clear whether one term is truly more general than the other or not. For instance, fiction is annotated in WordNet as a hypernym of story, while our method classified fiction as its hyponym.

A possible future research direction might be to quite simply extend our network to classify term-pairs simultaneously to multiple semantic relations, as in Pavlick et al. (2015). Such a multi-class model can hopefully better distinguish between these similar semantic relations.

Another notable category is hypernymy-like relations: these are other relations in the resources that could also be considered as hypernymy, but were annotated as negative due to our restrictive selection of only indisputable hypernymy relations from the resources (see Section 4.1). These include instances like (Goethe, occupation, novelist) and (Homo, subdivisionRanks, species).

Lastly, other errors made by the model often correspond to term-pairs that co-occur very few times in the corpus, e.g. xebec, a studio producing Anime, was falsely classified as a hyponym of anime.

False Negatives  We sampled 50 term-pairs that were falsely annotated as negative, and analyzed the major (overlapping) types of errors (Table 7).

Most of these pairs had only few co-occurrences in the corpus. This is often either due to infrequent terms (e.g. cbc.ca), or a rare sense of $x$ in which the hypernymy relation holds (e.g. (night,
Table 7: (Overlapping) categories of false negative pairs: (1) $x$ and $y$ co-occurred less than 25 times (average co-occurrences for true positive pairs is 99.7). (2) Either $x$ or $y$ is infrequent. (3) The hypernymy relation holds for a rare sense of $x$. (4) $(x, y)$ was incorrectly annotated as positive.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>low statistics</td>
<td>80%</td>
</tr>
<tr>
<td>infrequent term</td>
<td>36%</td>
</tr>
<tr>
<td>rare hyponym sense</td>
<td>16%</td>
</tr>
<tr>
<td>annotation error</td>
<td>8%</td>
</tr>
</tbody>
</table>
Table 8: The best hyper-parameters in every model.

<table>
<thead>
<tr>
<th>method</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snow</td>
<td>regularisation: $L_2$</td>
</tr>
<tr>
<td>LSTM</td>
<td>embeddings: GloVe-100-Wikipedia</td>
</tr>
<tr>
<td>SLQS</td>
<td>N=100, threshold = 0.0000064</td>
</tr>
<tr>
<td>Best Supervised</td>
<td>method: concatenation, classifier: SVM</td>
</tr>
<tr>
<td>LSTM-Integrated</td>
<td>embeddings: GloVe-50-Wiki</td>
</tr>
<tr>
<td></td>
<td>learning rate: $\alpha = 0.001$</td>
</tr>
<tr>
<td></td>
<td>dropout: $d = 0.5$</td>
</tr>
<tr>
<td>Snow</td>
<td>regularisation: $L_2$</td>
</tr>
<tr>
<td>Snow + Gen</td>
<td>regularisation: $L_2$</td>
</tr>
<tr>
<td>LSTM</td>
<td>embeddings: GloVe-50-Wiki</td>
</tr>
<tr>
<td>SLQS</td>
<td>N=100, threshold = 0.000062</td>
</tr>
<tr>
<td>Best Supervised</td>
<td>method: concatenation, classifier: SVM</td>
</tr>
<tr>
<td>LSTM-Integrated</td>
<td>embeddings: GloVe-50-Wiki</td>
</tr>
<tr>
<td></td>
<td>learning rate: $\alpha = 0.001$</td>
</tr>
<tr>
<td></td>
<td>word dropout: $d = 0.3$</td>
</tr>
</tbody>
</table>

Table 8: The best hyper-parameters in every model.
Chapter 5

Path-based vs. Distributional Information in Recognizing Lexical Semantic Relations
Path-based vs. Distributional Information in Recognizing Lexical Semantic Relations

Vered Shwartz  
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel  
vered1986@gmail.com

Ido Dagan  
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel  
dagan@cs.biu.ac.il

Abstract

Recognizing various semantic relations between terms is beneficial for many NLP tasks. While path-based and distributional information sources are considered complementary for this task, the superior results the latter showed recently suggested that the former’s contribution might have become obsolete. We follow the recent success of an integrated neural method for hypernymy detection (Shwartz et al., 2016) and extend it to recognize multiple relations. The empirical results show that this method is effective in the multiclass setting as well. We further show that the path-based information source always contributes to the classification, and analyze the cases in which it mostly complements the distributional information.

1 Introduction

Automated methods to recognize the lexical semantic relation the holds between terms are valuable for NLP applications. Two main information sources are used to recognize such relations: path-based and distributional. Path-based methods consider the joint occurrences of the two terms in a given pair in the corpus, where the dependency paths that connect the terms are typically used as features (A. Hearst, 1992; Snow et al., 2004; Nakashole et al., 2012; Riedel et al., 2013). Distributional methods are based on the disjoint occurrences of each term and have recently become popular using word embeddings (Mikolov et al., 2013; Pennington et al., 2014), which provide a distributional representation for each term. These embedding-based methods were reported to perform well on several common datasets (Baroni et al., 2012; Roller et al., 2014), consistently outperforming other methods (Santus et al., 2016; Neculescu et al., 2015).

While these two sources have been considered complementary, recent results suggested that path-based methods have no marginal contribution over the distributional ones. Recently, however, Shwartz et al. (2016) showed that a good path representation can provide substantial complementary information to the distributional signal in hypernymy detection, notably improving results on a new dataset.

In this paper, we apply Shwartz et al.’s (2016) method to recognize multiple semantic relations, and reexamine their findings over common datasets for the broader task. We show that this integrated method is indeed effective also in the multiclass setting, often performing better than each individual method. We further assess the contribution of path-based information to semantic relation classification. Even though the distributional source is dominant across most datasets, path-based information always contributed to it. In particular, path-based information seems to better capture the relationship between terms, rather than their individual properties, and can do so even for rare words or senses. Our code and data are available at https://github.com/vered1986/LexNET.

2 Background: HypeNET

Recently, Shwartz et al. (2016) introduced HypeNET, a hypernymy detection method based on the integration of the best-performing distributional method with a novel neural path representation, improving

This work is licenced under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/
upon state-of-the-art methods. In HypeNET, a term-pair \((x, y)\) is represented as a feature vector, consisting of both distributional and path-based features: 

\[
\vec{v}_{xy} = [\vec{v}_{w_x}, \vec{v}_{\text{paths}(x,y)}, \vec{v}_{w_y}],
\]

where \(\vec{v}_{w_x}\) and \(\vec{v}_{w_y}\) are \(x\) and \(y\)'s word embeddings, providing their distributional representation, and \(\vec{v}_{\text{paths}(x,y)}\) is a vector representing the dependency paths connecting \(x\) and \(y\) in the corpus. A binary classifier is trained on these vectors, yielding 

\[
c = \text{softmax}(W \cdot \vec{v}_{xy}),
\]

predicting hypernymy if \(c[1] > 0.5\).

Each dependency path is embedded using an LSTM (Hochreiter and Schmidhuber, 1997), as illustrated in the top row of Figure 1. This results in a path vector space in which semantically-similar paths (e.g. \(X\) is defined as \(Y\) and \(X\) is described as \(Y\)) have similar vectors. The vectors of all the paths that connect \(x\) and \(y\) are averaged to create \(\vec{v}_{\text{paths}(x,y)}\).

Shwartz et al. (2016) showed that this new path representation outperforms prior path-based methods for hypernymy detection, and that the integrated model yields a substantial improvement over each indi-
individual model. While HypeNET is designed for detecting hypernymy relations, it seems straightforward to extend it to classify term-pairs simultaneously to multiple semantic relations, as we describe next.

3 Classification Methods

We experiment with several classification models, as illustrated in Figure 1:

Path-based HypeNET’s path-based model (PB) is a binary classifier trained on the path vectors alone:

\[ \vec{v}_{paths(x,y)} \]. We adapt the model to classify multiple relations by changing the network softmax output \( c \) to a distribution over \( k \) target relations, classifying a pair to the highest scoring relation:

\[ r = \arg\max_i c[i]. \]

Distributional We train an SVM classifier on the concatenation of \( x \) and \( y \)’s word embeddings \([\vec{v}_{w_x}, \vec{v}_{w_y}]\) (Baroni et al., 2012) (DS).1 Levy et al. (2015) claimed that such a linear classifier is incapable of capturing interactions between \( x \) and \( y \)’s features, and that instead it learns separate properties of \( x \) or \( y \), e.g., that \( y \) is a prototypical hypernym. To examine the effect of non-linear expressive power on the model, we experiment with a neural network with a single hidden layer trained on \([\vec{v}_{w_x}, \vec{v}_{w_y}]\) (DS).2

Integrated We similarly adapt the HypeNET integrated model to classify multiple semantic relations (LexNET). Based on the same motivation of DS, we also experiment with a version of the network with a hidden layer (LexNET\(_h\)), re-defining

\[ c = \text{softmax}(W_2 \cdot \vec{h} + b_2), \]

where \( \vec{h} = \tanh(W_1 \cdot \vec{v}_{xy} + b_1) \) is the hidden layer. The technical details of our network are identical to Shwartz et al. (2016).

4 Datasets

We use four common semantic relation datasets that were created using semantic resources: K&H+N (Necusulescu et al., 2015) (an extension to Kozareva and Hovy (2010)), BLESS (Baroni and Lenci, 2011), EVALution (Santus et al., 2015), and ROOT09 (Santus et al., 2016).

Table 1 displays the relation types and number of instances in each dataset. Most dataset relations are parallel to WordNet relations, such as hypernymy (cat, animal) and meronymy (hand, body), with an additional random relation for negative instances. BLESS contains the event and attribute relations, connecting a concept with a typical activity/property (e.g., (alligator, swim) and (alligator, aquatic)). EVALution contains a richer schema of semantic relations, with some redundancy: it contains both meronymy and holonymy (e.g., for bicycle and wheel), and the fine-grained substance-holonymy relation. We removed two relations with too few instances: Entails and MemberOf.

To prevent the lexical memorization effect (Levy et al., 2015), Santus et al. (2016) added negative switched hypernym-hypernym pairs (e.g. (apple, animal), (cat, fruit)) to ROOT09, which were reported to reduce this effect.

5 Results

Like Shwartz et al. (2016), we tuned the methods’ hyper-parameters on the validation set of each dataset, and used Wikipedia as the corpus. Table 2 displays the performance of the different methods on all datasets, in terms of recall, precision and \( F_1 \).

Our first empirical finding is that Shwartz et al.’s (2016) algorithm is effective in the multiclass setting as well (LexNET). The only dataset on which performance is mediocre is EVALution, which seems to be inherently harder for all methods, due to its large number of relations and small size. The performance differences between LexNET and DS are statistically significant on all datasets with \( p \)-value of 0.01 (paired

---

1We experimented also with difference \( \vec{v}_{w_x} - \vec{v}_{w_y} \) and other classifiers, but concatenation yielded the best performance.

2This was previously done by Bowman et al. (2014), with promising results, but on a small artificial vocabulary.
Table 2: Performance scores (precision, recall and $F_1$) of each individual approach and the integrated models. To compute the metrics we used scikit-learn (Pedregosa et al., 2011) with the “averaged” setup, which computes the metrics for each relation, and reports their average, weighted by support (the number of true instances for each relation). Note that it can result in an $F_1$ score that is not the harmonic mean of precision and recall.

<table>
<thead>
<tr>
<th>method</th>
<th>K&amp;H+N</th>
<th>BLESS</th>
<th>ROOT09</th>
<th>EVALution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P$</td>
<td>$R$</td>
<td>$F_1$</td>
<td>$P$</td>
</tr>
<tr>
<td>PB</td>
<td>0.713</td>
<td>0.604</td>
<td>0.55</td>
<td>0.759</td>
</tr>
<tr>
<td>DS$_h$</td>
<td>0.909</td>
<td>0.906</td>
<td>0.904</td>
<td>0.811</td>
</tr>
<tr>
<td>DS$_A$</td>
<td>0.983</td>
<td>0.984</td>
<td>0.983</td>
<td>0.891</td>
</tr>
<tr>
<td>LexNET</td>
<td>0.985</td>
<td>0.986</td>
<td>0.985</td>
<td>0.894</td>
</tr>
<tr>
<td>LexNET$_{h}$</td>
<td>0.984</td>
<td>0.985</td>
<td>0.984</td>
<td><strong>0.895</strong></td>
</tr>
</tbody>
</table>

Table 3: The number of term-pairs that were correctly classified by the integrated model while being incorrectly classified by DS$_h$, in each test set, with corresponding examples of such term-pairs.

<table>
<thead>
<tr>
<th>dataset</th>
<th>#pairs</th>
<th>x</th>
<th>y</th>
<th>gold label</th>
<th>DS$_h$ prediction</th>
<th>possible explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>K&amp;H+N</td>
<td>102</td>
<td>firefly</td>
<td>racehorse</td>
<td>false</td>
<td>hypo</td>
<td>(x, car) frequent label is hypo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>larvacean</td>
<td></td>
<td>false</td>
<td>false</td>
<td>out of the embeddings vocabulary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>salp</td>
<td></td>
<td>sibl</td>
<td>false</td>
<td>rare terms larvacean and salp</td>
</tr>
<tr>
<td>BLESS</td>
<td>275</td>
<td>tankcr</td>
<td></td>
<td>ship</td>
<td>hyper</td>
<td>(x, ship) frequent label is event</td>
</tr>
<tr>
<td></td>
<td></td>
<td>herring</td>
<td></td>
<td>lie</td>
<td>random</td>
<td>event</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>salt</td>
<td>event</td>
<td>random</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>non-prototypical relation</td>
</tr>
<tr>
<td>ROOT09</td>
<td>562</td>
<td>toaster</td>
<td></td>
<td>vehicle</td>
<td>RANDOM</td>
<td>(x, vehicle) frequent label is HYPER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rice</td>
<td></td>
<td>grain</td>
<td>HYPER</td>
<td>(x, grain) frequent label is RANDOM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lung</td>
<td></td>
<td>organ</td>
<td>HYPER</td>
<td>(x, organ) frequent label is polysemous term organ</td>
</tr>
<tr>
<td>EVALution</td>
<td>235</td>
<td>pick</td>
<td></td>
<td>metal</td>
<td>MadeOf</td>
<td>ISA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>abstract</td>
<td></td>
<td>concrete</td>
<td>MadeOf</td>
<td>rare sense of pick</td>
</tr>
<tr>
<td></td>
<td></td>
<td>line</td>
<td></td>
<td>thread</td>
<td>Synonym</td>
<td>polysemous term concrete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(x, thread) frequent label is MadeOf</td>
</tr>
</tbody>
</table>

The performance differences between LexNET and DS$_h$ are statistically significant on BLESS and ROOT09 with p-value of 0.01, and on EVALution with p-value of 0.05.

DS$_A$ consistently improves upon DS. The hidden layer seems to enable interactions between $x$ and $y$’s features, which is especially noticed in ROOT09, where the hypernym $F_1$ score in particular rose from 0.25 to 0.45. Nevertheless, we did not observe a similar behavior in LexNET, which worked similarly or slightly worse than LexNET$_h$, which worked similarly or slightly worse than LexNET. It is possible that the contributions of the hidden layer and the path-based source over the distributional signal are redundant. It may also be that the larger number of parameters in LexNET$_h$ prevents convergence to the optimal values given the modest amount of training data, stressing the need for large-scale datasets that will benefit training neural methods.

6 Analysis

Table 2 demonstrates that the distributional source is dominant across most datasets, with DS performing better than PB. Although by design DS does not consider the relation between $x$ and $y$, but rather learns properties of $x$ or $y$, it performs well on BLESS and K&H+N. DS$_h$ further manages to capture relations at the distributional level, leaving the path-based source little room for improvement on these two datasets.

On ROOT09, on the other hand, DS achieved the lowest performance. Our analysis reveals that this is due to the switched hypernym pairs, which drastically hurt the ability to memorize individual single words, hence reducing performance. The $F_1$ scores of DS on this dataset were 0.91 for co-hyponyms but only 0.25 for hypernyms, while PB scored 0.87 and 0.66 respectively. Moreover, LexNET gains 10 points over DS$_h$, suggesting the better capacity of path-based methods to capture relations between terms.

6.1 Analysis of Information Sources

To analyze the contribution of the path-based information source, we examined the term-pairs that were correctly classified by the best performing integrated model (LexNET/LexNET$_h$) while being incorrectly classified by DS$_h$. Table 3 displays the number of such pairs in each dataset, with corresponding term-pair examples. The common errors are detailed below:

Lexical Memorization $DS_h$ often classifies $(x, y)$ term-pairs according to the most frequent relation of one of the terms (usually $y$) in the train set. The error is mostly prominent in ROOT09 (405/562 pairs, 3 We also tried adding a hidden layer only over the distributional features of LexNET, but it did not improve performance.
Non-prototypical Relations \( D_{5b} \) might fail to recognize non-prototypical relations between terms, i.e. when \( y \) is a less-prototypical relatum of \( x \), as in mero:(villa, guest), event:(cherry, pick), and attri:(piano, electric). While being overlooked by the distributional methods, these relations are often expressed in joint occurrences in the corpus, allowing the path-based component to capture them.

Rare Terms The integrated method often managed to classify term-pairs in which at least one of the terms is rare (e.g. hyper:(mastodon, proboscidean)), where the distributional method failed. It is a well-known shortcoming of path-based methods that they require informative co-occurrences of \( x \) and \( y \), which are not always available. With that said, thanks to the averaged path representation, \( PB \) can capture the relation between terms even if they only co-occur once within an informative path, while the distributional representation of rare terms is of lower quality. We note that the path-based information of \((x, y)\) is encoded in the vector \( \vec{v}_{paths(x,y)} \), which is the averaged vector representation of all paths that connected \( x \) and \( y \) in the corpus. Unlike other path-based methods in the literature, this representation is indifferent to the total number of paths, and as a result, even a single informative path can lead to successful classification.

Rare Senses Similarly, the path-based component succeeded to capture relations for rare senses of words where \( D_{5b} \) failed, e.g. mero:(piano, key), event:(table, draw). Distributional representations suffer from insufficient representation of rare senses, while \( PB \) may capture the relation with a single meaningful occurrence of the rare sense with its related term. At the same time, it is less likely for a polysemous term to co-occur, in its non-related senses, with the candidate relatum. For instance, paths connecting piano to key are likely to correspond to the keyboard sense of key, indicating the relation that does hold for this pair with respect to this rare sense.

Finally, we note that \textsc{lexnet}, as well as the individual methods, perform poorly on synonyms and antonyms. The synonymy \( F_1 \) score in \textit{Evaluation} was 0.35 in \textsc{lexnet} and in \( D_{5b} \) and only 0.09 in \( PB \), reassessing prior findings (Mirkin et al., 2006) that the path-based approach is weak in recognizing synonyms, which do not tend to co-occur. \( D_{5b} \) performed poorly also on antonyms (\( F_1 = 0.54 \)), which were often mistaken for synonyms, since both tend to occur in the same contexts. It seems worthwhile to try improving the model with insights from prior work on these specific relations (Santus et al., 2014; Mohammad et al., 2013) or by using additional information sources, like multilingual data (Pavlick et al., 2015).

7 Conclusion

We presented an adaptation to HypeNET (Shwartz et al., 2016) that classifies term-pairs to one of multiple semantic relations. Evaluation on common datasets shows that HypeNET is extensible to the multi-class setting and performs better than each individual method.

Although the distributional information source is dominant across most datasets, it consistently benefits from path-based information, particularly when finer modeling of inter-term relationship is needed.

Finally, we note that all common datasets were created synthetically using semantic resources, leading to inconsistent behavior of the different methods, depending on the particular distribution of examples in each dataset. This stresses the need to develop “naturally” distributed datasets that would be drawn from corpora, while reflecting realistic distributions encountered by semantic applications.

Acknowledgments

This work was partially supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).
References


Enrico Santus, Qin Lu, Alessandro Lenci, and Churen Huang. 2014. Unsupervised antonym-synonym discrimination in vector space.


Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. In *LREC*.


Chapter 6

Acquiring Predicate Paraphrases from News Tweets
Acquiring Predicate Paraphrases from News Tweets

Vered Shwartz  
Gabriel Stanovsky  
Ido Dagan  
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel  
{vered1986, gabriel.satanovsky}@gmail.com, dagan@cs.biu.ac.il

Abstract

We present a simple method for ever-growing extraction of predicate paraphrases from news headlines in Twitter. Analysis of the output of ten weeks of collection shows that the accuracy of paraphrases with different support levels is estimated between 60-86%. We also demonstrate that our resource is to a large extent complementary to existing resources, providing many novel paraphrases. Our resource is publicly available, continuously expanding based on daily news.

1 Introduction

Recognizing that various textual descriptions across multiple texts refer to the same event or action can benefit NLP applications such as recognizing textual entailment (Dagan et al., 2013) and question answering. For example, to answer “when did the US Supreme Court approve same-sex marriage?” given the text “In June 2015, the Supreme Court ruled for same-sex marriage”, approve and ruled for should be identified as describing the same action.

To that end, much effort has been devoted to identifying predicate paraphrases, some of which resulted in releasing resources of predicate entailment or paraphrases. Two main approaches were proposed for that matter; the first leverages the similarity in argument distribution across a large corpus between two predicates (e.g. [a]0 buy [a]1 / [a]0 acquire [a]1) (Lin and Pantel, 2001; Berant et al., 2010). The second approach exploits bilingual parallel corpora, extracting as paraphrases pairs of texts that were translated identically to foreign languages (Ganitkevitch et al., 2013).

While these methods have produced exhaustive resources which are broadly used by applications, their accuracy is limited. Specifically, the first approach may extract antonyms, that also have similar argument distribution (e.g. [a]0 raise to [a]1 / [a]0 fall to [a]1) while the second may conflate multiple senses of the foreign phrase.

A third approach was proposed to harvest paraphrases from multiple mentions of the same event in news articles.1 This approach assumes that various redundant reports make different lexical choices to describe the same event. Although there has been some work following this approach (e.g. Shinyama et al., 2002; Shinyama and Sekine, 2006; Roth and Frank, 2012; Zhang and Weld, 2013), it was less exhaustively investigated and did not result in creating paraphrase resources.

In this paper we present a novel unsupervised method for ever-growing extraction of lexically-divergent predicate paraphrase pairs from news tweets. We apply our methodology to create a resource of predicate paraphrases, exemplified in Table 1.

Analysis of the resource obtained after ten

---

1This corresponds to instances of event coreference (Bagga and Baldwin, 1999).

Table 1: A sample from the top-ranked predicate paraphrases.

<table>
<thead>
<tr>
<th>[a]0 introduce [a]1</th>
<th>[a]0 welcome [a]1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a]0 appoint [a]1</td>
<td>[a]0 to become [a]1</td>
</tr>
<tr>
<td>[a]0 die at [a]1</td>
<td>[a]0 pass away at [a]1</td>
</tr>
<tr>
<td>[a]0 hit [a]1</td>
<td>[a]0 sink to [a]1</td>
</tr>
<tr>
<td>[a]0 be investigate [a]1</td>
<td>[a]0 be probe [a]1</td>
</tr>
<tr>
<td>[a]0 eliminate [a]1</td>
<td>[a]0 slash [a]1</td>
</tr>
<tr>
<td>[a]0 announce [a]1</td>
<td>[a]0 unveil [a]1</td>
</tr>
<tr>
<td>[a]0 quit after [a]1</td>
<td>[a]0 resign after [a]1</td>
</tr>
<tr>
<td>[a]0 announce as [a]1</td>
<td>[a]0 to become [a]1</td>
</tr>
<tr>
<td>[a]0 threaten [a]1</td>
<td>[a]0 warn [a]1</td>
</tr>
<tr>
<td>[a]0 die at [a]1</td>
<td>[a]0 live until [a]1</td>
</tr>
<tr>
<td>[a]0 double down on [a]1</td>
<td>[a]0 stand by [a]1</td>
</tr>
<tr>
<td>[a]0 kill [a]1</td>
<td>[a]0 shoot [a]1</td>
</tr>
<tr>
<td>[a]0 approve [a]1</td>
<td>[a]0 pass [a]1</td>
</tr>
<tr>
<td>[a]0 would be cut under [a]1</td>
<td>[a]0 slash [a]0</td>
</tr>
<tr>
<td>seize [a]0 at [a]1</td>
<td>to grab [a]0 at [a]1</td>
</tr>
</tbody>
</table>
weeks of acquisition shows that the set of paraphrases reaches accuracy of 60-86% at different levels of support. Comparison to existing resources shows that, even as our resource is still smaller in orders of magnitude from existing resources, it complements them with non-consecutive predicates (e.g. take \([a_0]\) from \([a_1]\)) and paraphrases which are highly context specific.

The resource and the source code are available at http://github.com/vered1986/Chirps. As of the end of May 2017, it contains 456,221 predicate pairs in 1,239,463 different contexts. Our resource is ever-growing and is expected to contain around 2 million predicate paraphrases within a year. Until it reaches a large enough size, we will release a daily update, and at a later stage, we plan to release a periodic update.

2 Background

2.1 Existing Paraphrase Resources

A prominent approach to acquire predicate paraphrases is to compare the distribution of their arguments across a corpus, as an extension to the distributional hypothesis (Harris, 1954). DIRT (Lin and Pantel, 2001) is a resource of 10 million paraphrases, in which the similarity between predicate pairs is estimated by the geometric mean of the similarities of their argument slots. Berant (2012) constructed an entailment graph of distributionally similar predicates by enforcing transitivity constraints and applying global optimization, releasing 52 million directional entailment rules (e.g. \([a_0]\) shoot \([a_1]\) → \([a_0]\) kill \([a_1]\)).

A second notable source for extracting paraphrases is multiple translations of the same text (Barzilay and McKeown, 2001). The Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015) is a huge collection of paraphrases extracted from bilingual parallel corpora. Paraphrases are scored heuristically, and the database is available for download in six increasingly large sizes according to scores (the smallest size being the most accurate). In addition to lexical paraphrases, PPDB also consists of 140 million syntactic paraphrases, some of which include predicates with non-terminals as arguments.

2.2 Using Multiple Event Descriptions

Another line of work extracts paraphrases from redundant comparable news articles (e.g. Shinyama et al., 2002; Barzilay and Lee, 2003). The assumption is that multiple news articles describing the same event use various lexical choices, providing a good source for paraphrases. Heuristics are applied to recognize that two news articles discuss the same event, such as lexical overlap and same publish date (Shinyama and Sekine, 2006). Given such a pair of articles, it is likely that predicates connecting the same arguments will be paraphrases, as in the following example:

1. GOP lawmakers introduce new health care plan
2. GOP lawmakers unveil new health care plan

Zhang and Weld (2013) and Zhang et al. (2015) introduced methods that leverage parallel news streams to cluster predicates by meaning, using temporal constraints. Since this approach acquires paraphrases from descriptions of the same event, it is potentially more accurate than methods that acquire paraphrases from the entire corpus or translation phrase table. However, there is currently no paraphrase resource acquired in this approach.3

Finally, Xu et al. (2014) developed a supervised model to collect sentential paraphrases from Twitter. They used Twitter’s trending topic service, and considered two tweets from the same topic as paraphrases if they shared a single anchor word.

3 Resource Construction

We present a methodology to automatically collect binary verbal predicate paraphrases from Twitter. We first obtain news related tweets (§3.1) from which we extract propositions (§3.2). For a candidate pair of propositions, we assume that if both arguments can be matched then the predicates are likely paraphrases (§3.3). Finally, we rank the predicate pairs according to the number of instances in which they were aligned (§3.4).

3.1 Obtaining News Headlines

We use Twitter as a source of readily available news headlines. The 140 characters limit makes tweets concise, informative and independent of each other, obviating the need to resolve document-level entity coreference. We query the Twitter Search API4 via Twitter Search.5 We use

---

2Chirp is a paraphrase of tweet.

---

4https://apps.twitter.com/
5https://github.com/ckoepp/TwitterSearch
Turkey intercepts the plane which took off from Moscow, furious about the plane, threatens to retaliate (1) [Turkey] intercepts [plane] (2) [plane] took off from [Moscow] (Russia) threatens to [retaliate].

Figure 1: PropS structures and the corresponding propositions extracted by our process. Left: multi-word predicates and multiple extractions per tweet. Right: argument reduction.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
<th>Argument Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manafort hid payments from Ukraine party with Moscow ties</td>
<td>hide [a]</td>
<td>Paul Manafort payments</td>
<td></td>
</tr>
<tr>
<td>Manafort laundered the payments through Belize</td>
<td>launder [a]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send immigration judges to cities to speed up deportations</td>
<td>to send [a]</td>
<td>immigration judges cities</td>
<td></td>
</tr>
<tr>
<td>Immigration judges headed to 12 cities to speed up deportations</td>
<td>headed to [a]</td>
<td>immigration judges 12 cities</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Examples of predicate paraphrase instances in our resource: each instance contains two tweets, predicate types extracted from them, and the instantiations of arguments.

Twitter’s news filter that retrieves tweets containing links to news websites, and limit the search to English tweets.

### 3.2 Proposition Extraction

We extract propositions from news tweets using PropS (Stanovsky et al., 2016), which simplifies dependency trees by conveniently marking a wide range of predicates (e.g., verbal, adjectival, non-lexical) and positioning them as direct heads of their corresponding arguments. Specifically, we run PropS over dependency trees predicted by spaCy and extract predicate types (as in Table 1) composed of verbal predicates, datives, prepositions, and auxiliaries.

Finally, we employ a pre-trained argument reduction model to remove non-restrictive argument modifications (Stanovsky and Dagan, 2016). This is essential for our subsequent alignment step, as it is likely that short and concise phrases will tend to match more frequently in comparison to longer, more specific arguments. Figure 1 exemplifies some of the phenomena handled by this process, along with the automatically predicted output.

### 3.3 Generating Paraphrase Instances

Following the assumption that different descriptions of the same event are bound to be redundant (as discussed in Section 2.2), we consider two predicates as paraphrases if: (1) They appear on the same day, and (2) Each of their arguments aligns with a unique argument in the other predicate, either by strict matching (short edit distance, abbreviations, etc.) or by a looser matching (partial token matching or WordNet synonyms).\(^7\) Table 2 shows examples of predicate paraphrase instances in the resource.

### 3.4 Resource Release

The resource release consists of two files:

1. **Instances**: the specific contexts in which the predicates are paraphrases (as in Table 2). In practice, to comply with Twitter policy, we release predicate paraphrase pair types along with their arguments and tweet IDs, and provide a script for downloading the full texts.

2. **Types**: predicate paraphrase pair types (as in Table 1). The types are ranked in a descending order according to a heuristic accuracy score:

   \[
   s = count \cdot \left(1 + \frac{d}{N}\right)
   \]

   where \(count\) is the number of instances in which the predicate types were aligned (Section 3.3), \(d\) is the number of different days in which they were aligned, and \(N\) is the number of days since the resource collection began.

Taking into account the number of different days in which predicates were aligned reduces the noise caused by two entities that undergo two different actions on the same day. For example, the following tweets from the day of Chuck Berry’s death:

1. Last year when Chuck Berry turned 90
2. Chuck Berry dies at 90

\(^7\)In practice, our publicly available code requires that at least one pair of arguments will strictly match.
yield the incorrect type \([a]_0\) turn \([a]_1\) / \([a]_0\) die at \([a]_1\). While there may be several occurrences of that type on the same day, it is not expected to re-occur in other news events (in different days), yielding a low accuracy score.

4 Analysis of Resource Quality

We estimate the quality of the resource obtained after ten weeks of collection by annotating a sample of the extracted paraphrases.

The annotation task was carried out in Amazon Mechanical Turk.\(^8\) To ensure the quality of workers, we applied a qualification test and required a 99% approval rate for at least 1,000 prior tasks. We assigned each annotation to 3 workers and used the majority vote to determine the correctness of paraphrases.

We followed a similar approach to instance-based evaluation (Szpektor et al., 2007), and let workers judge the correctness of a predicate pair (e.g. \([a]_0\) purchase \([a]_1\) \(\rightarrow\) \([a]_0\) acquire \([a]_1\)) through 5 different instances (e.g. Intel purchased Mobileye/Intel acquired Mobileye). We considered the type as correct if at least one of its instance-pairs were judged as correct. The idea that lies behind this type of evaluation is that predicate pairs are difficult to judge out-of-context.

Differently from Szpektor et al. (2007), we used the instances in which the paraphrases appeared originally, as those are available in the resource.

\(^8\)https://www.mturk.com/mturk

4.1 Quality of Extractions and Ranking

To evaluate the resource accuracy, and following the instance-based evaluation scheme, we only considered paraphrases that occurred in at least 5 instances (which currently constitute 10% of the paraphrase types). We partition the types into four increasingly large bins according to their scores (the smallest bin being the most accurate), similarly to PPDB (Ganitkevitch et al., 2013), and annotate a sample of 50 types from each bin. Figure 2(a) shows that the frequent types achieve up to 86% accuracy.

The accuracy expectedly increases with the score, except for the lowest-score bin \((0, 10]\) which is more accurate than the next one \((10, 20]\). At the current stage of the resource there is a long tail of paraphrases that appeared few times. While many of them are incorrect, there are also true paraphrases that are infrequent and therefore have a low accuracy score. We expect that some of these paraphrases will occur again in the future and their accuracy score will be strengthened.

4.2 Size and Accuracy Over Time

To estimate future usefulness, Figure 2(b) plots the resource size (in terms of types and instances) and estimated accuracy through each week in the first 10 weeks of collection.

The accuracy at a specific time was estimated by annotating a sample of 50 predicate pair types with accuracy score \(\geq 20\) in the resource obtained...
at that time, which roughly correspond to the top ranked 1.5% types.

Figure 2(b) demonstrates that these types maintain a level of around 80% in accuracy. The resource growth rate (i.e. the number of new types) is expected to change with time. We predict that the resource will contain around 2 million types in one year.\footnote{For up-to-date resource statistics, see: https://github.com/vered1986/Chirps/tree/master/resource.}

Table 3 exemplifies some of the predicate pairs that do not exist in both resources. Specifically, our resource contains many non-consecutive predicates (e.g. reveal \([a_0]_0 \) to \([a_1]_1 \) / share \([a_0]_0 \) with \([a_1]_1 \) that by definition do not exist in Berant.

Some pairs, such as \([a_0]_0 \) get \([a_1]_1 \) / \([a_0]_0 \) sentence to \([a_1]_1 \), are context-specific, occurring when \([a_0]_0 \) is a person and \([a_1]_1 \) is the time they are about to serve in prison. Given that get has a broad distribution of argument instantiations, this paraphrase and similar paraphrases are less likely to exist in resources that rely on the distribution of arguments in the entire corpus.

## 5 Comparison to Existing Resources

The resources which are most similar to ours are Berant (Berant, 2012), a resource of predicate entailments, and PPDB (Pavlick et al., 2015), a resource of paraphrases, both described in Section 2.

We expect our resource to be more accurate than resources which are based on the distributional approach (Berant, 2012; Lin and Pantel, 2001). In addition, in comparison to PPDB, we specialize on binary verbal predicates, and apply an additional phase of proposition extraction, handling various phenomena such as non-consecutive particles and minimality of arguments.

Berant (2012) evaluated their resource against a dataset of predicate entailments (Zeichner et al., 2012), using a recall-precision curve to show the performance obtained with a range of thresholds on the resource score. This kind of evaluation is less suitable for our resource; first, predicate entailment is directional, causing paraphrases with the wrong entailment direction to be labeled negative in the dataset. Second, since our resource is still relatively small, it is unlikely to have sufficient coverage of the dataset at that point. We therefore leave this evaluation to future work.

To demonstrate the added value of our resource, we show that even in its current size, it already contains accurate predicate pairs which are absent from the existing resources. Rather than comparing against labeled data, we use types with score \( \geq 50 \) from our resource (1,778 pairs), which were assessed as accurate (Section 4.2).

We checked whether these predicate pairs are covered by Berant and PPDB. To eliminate directionality, we looked for types in both directions, i.e. for a predicate pair \((p_1, p_2)\) we searched for both \((p_1, p_2)\) and \((p_2, p_1)\). Overall, we found that 67% of these types do not exist in Berant, 62% in PPDB, and 49% in neither.

## 6 Conclusion

We presented a new unsupervised method to acquire fairly accurate predicate paraphrases from news tweets discussing the same event. We release a growing resource of predicate paraphrases. Qualitative analysis shows that our resource adds value over existing resources. In the future, when the resource is comparable in size to the existing resources, we plan to evaluate its intrinsic accuracy on annotated test sets, as well as its extrinsic benefits in downstream NLP applications.

## Acknowledgments

This work was partially supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).
References


Chapter 7

Olive Oil Is Made of Olives, Baby
Oil Is Made for Babies:
Interpreting Noun Compounds
Using Paraphrases in a Neural Model
Olive Oil is Made of Olives, Baby Oil is Made for Babies: Interpreting Noun Compounds using Paraphrases in a Neural Model

Vered Shwartz*
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
vered1986@gmail.com

Chris Waterson
Google Inc.
1600 Amphitheatre Parkway
Mountain View, California 94043
waterson@google.com

Abstract
Automatic interpretation of the relation between the constituents of a noun compound, e.g. olive oil (source) and baby oil (purpose) is an important task for many NLP applications. Recent approaches are typically based on either noun-compound representations or paraphrases. While the former has initially shown promising results, recent work suggests that the success stems from memorizing single prototypical words for each relation. We explore a neural paraphrasing approach that demonstrates superior performance when such memorization is not possible.

1 Introduction
Automatic classification of a noun-compound (NC) to the implicit semantic relation that holds between its constituent words is beneficial for applications that require text understanding. For instance, a personal assistant asked “do I have a morning meeting tomorrow?” should search the calendar for meetings occurring in the morning, while for group meeting it should look for meetings with specific participants. The NC classification task is a challenging one, as the meaning of an NC is often not easily derivable from the meaning of its constituent words (Spärck Jones, 1983).

Previous work on the task falls into two main approaches. The first maps NCs to paraphrases that express the relation between the constituent words (e.g. Nakov and Hearst, 2006; Nulty and Costello, 2013), such as mapping coffee cup and garbage dump to the pattern \([w_1] \text{ CONTAINS } [w_2] \). The second approach computes a representation for NCs from the distributional representation of their individual constituents. While this approach yielded promising results, recently, Dima (2016) showed that similar performance is achieved by representing the NC as a concatenation of its constituent embeddings, and attributed it to the lexical memorization phenomenon (Levy et al., 2015).

In this paper we apply lessons learned from the parallel task of semantic relation classification. We adapt HypeNET (Shwartz et al., 2016) to the NC classification task, using their path embeddings to represent paraphrases and combining with distributional information. We experiment with various evaluation settings, including settings that make lexical memorization impossible. In these settings, the integrated method performs better than the baselines. Even so, the performance is mediocre for all methods, suggesting that the task is difficult and warrants further investigation.¹

2 Background
Various tasks have been suggested to address noun-compound interpretation. NC paraphrasing extracts texts explicitly describing the implicit relation between the constituents, for example student protest is a protest LED BY, BE SPONSORED BY, or BE ORGANIZED BY students (e.g. Nakov and Hearst, 2006; Kim and Nakov, 2011; Hendrickx et al., 2013; Nulty and Costello, 2013). Compositionality prediction determines to what extent the meaning of the NC can be expressed in terms of the meaning of its constituents, e.g. spelling bee is non-compositional, as it is not related to bee (e.g. Reddy et al., 2011). In this paper we focus on the NC classification task, which is defined as follows: given a pre-defined set of relations, classify \( nc = w_1w_2 \) to the relation that holds between \( w_1 \) and \( w_2 \). We review the various

¹The code is available at https://github.com/tensorflow/models/tree/master/research/lexnet_nc.
features used in the literature for classification.  

2.1 Compositional Representations

In this approach, classification is based on a vector representing the NC \((w_1, w_2)\), which is obtained by applying a function to its constituents’ distributional representations: \(\vec{v}_{w_1}, \vec{v}_{w_2} \in \mathbb{R}^n\). Various functions have been proposed in the literature.

Mitchell and Lapata (2010) proposed 3 simple combinations of \(\vec{v}_{w_1}\) and \(\vec{v}_{w_2}\) (additive, multiplicative, dilation). Others suggested to represent compositions by applying linear functions, encoded as matrices, over word vectors. Baroni and Zamparelli (2010) focused on adjective-noun compositions (AN) and represented adjectives as matrices, nouns as vectors, and ANs as their multiplication. Matrices were learned with the objective of minimizing the distance between the learned vector and the observed vector (computed from corpus occurrences) of each AN. The full-additive model (Zanzotto et al., 2010; Dinu et al., 2013) is a similar approach that works on any two-word composition, multiplying each word by a square matrix: \(nc = A \cdot \vec{v}_{w_1} + B \cdot \vec{v}_{w_2}\).

Socher et al. (2012) suggested a non-linear composition model. A recursive neural network operates bottom-up on the output of a constituency parser to represent variable-length phrases. Each constituent is represented by a vector that captures its meaning and a matrix that captures how it modifies the meaning of constituents that it combines with. For a binary NC, \(nc = g(W \cdot [\vec{v}_{w_1}; \vec{v}_{w_2}])\), where \(W \in \mathbb{R}^{2n \times n}\) and \(g\) is a non-linear function.

These representations were used as features in NC classification, often achieving promising results (e.g. Van de Cruys et al., 2013; Dima and Hinrichs, 2015). However, Dima (2016) recently showed that similar performance is achieved by representing the NC as a concatenation of its constituent embeddings, and argued that it stems from memorizing prototypical words for each relation. For example, classifying any NC with the head oil to the SOURCE relation, regardless of the modifier.

2.2 Paraphrasing

In this approach, the paraphrases of an NC, i.e. the patterns connecting the joint occurrences of the constituents in a corpus, are treated as features. For example, both paper cup and steel knife may share the feature MADE OF. Séaghdhda and Copestake (2013) leveraged this “relational similarity” in a kernel-based classification approach. They combined the relational information with the complementary lexical features of each constituent separately. Two NCs labeled to the same relation may consist of similar constituents (paper-steel, cup-knife) and may also appear with similar paraphrases. Combining the two information sources has shown to be beneficial, but it was also noted that the relational information suffered from data sparsity: many NCs had very few paraphrases, and paraphrase similarity was based on ngram overlap.

Recently, Surtani and Paul (2015) suggested to represent NCs in a vector space model (VSM) using paraphrases as features. These vectors were used to classify new NCs based on the nearest neighbor in the VSM. However, the model was only tested on a small dataset and performed similarly to previous methods.

3 Model

We similarly investigate the use of paraphrasing for NC relation classification. To generate a signal for the joint occurrences of \(w_1\) and \(w_2\), we follow the approach used by HypeNET (Shwartz et al., 2016). For an \(w_1w_2\) in the dataset, we collect all the dependency paths that connect \(w_1\) and \(w_2\) in the corpus, and learn path embeddings as detailed in Section 3.2. Section 3.1 describes the classification models with which we experimented.

3.1 Classification Models

Figure 1 provides an overview of the models: path-based, integrated, and integrated-NC, each of which incrementally adds new features not present in the previous model. In the following sections, \(\vec{x}\) denotes the input vector representing the NC. The network classifies NC to the highest scoring relation: \(r = \arg \max_i \text{softmax}(\vec{o}_i)\), where \(\vec{o}\) is the output layer. All networks contain a single hidden layer whose dimension is \(|x|^k\), \(k\) is the number of relations in the dataset. See Appendix A for additional technical details.

Path-based. Classifies the NC based only on the paths connecting the joint occurrences of \(w_1\) and \(w_2\) in the corpus, denoted \(P(w_1, w_2)\). We define the feature vector as the average of its path embeddings, where the path embedding \(\vec{p}\) of a path \(p\) is
weighted by its frequency \( f_{p,(w_1,w_2)} \):

\[
\vec{x} = \vec{v}_{P(w_1,w_2)} = \frac{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)} \cdot \vec{p}}{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)}}
\]

**Integrated.** We concatenate \( w_1 \) and \( w_2 \)'s word embeddings to the path vector, to add distributional information: \( x = [\vec{v}_{w_1}, \vec{v}_{w_2}, \vec{v}_{nc}, \vec{v}_{P(w_1,w_2)}] \). Potentially, this allows the network to utilize the contextual properties of each individual constituent, e.g. assigning high probability to SUBSTANCE-MATERIAL-INGREDIENT for edible \( w_1 \)'s (e.g. *vanilla pudding*, *apple cake*).

**Integrated-NC.** We add the NC’s observed vector \( \vec{v}_{nc} \) as additional distributional input, providing the contexts in which \( w_1 \) and \( w_2 \) occur as an NC: 

\[
\vec{v}_{nc} = [\vec{v}_{w_1}, \vec{v}_{w_2}, \vec{v}_{nc}, \vec{v}_{P(w_1,w_2)}].
\]

Like Dima (2016), we learn NC vectors using the GloVe algorithm (Pennington et al., 2014), by replacing each NC occurrence in the corpus with a single token.

This information can potentially help clustering NCs that appear in similar contexts despite having low pairwise similarity scores between their constituents. For example, *gun violence* and *abortion rights* belong to the TOPIC relation and may appear in similar news-related contexts, while (*gun*, *abortion*) and (*violence*, *rights*) are dissimilar.

### 3.2 Path Embeddings

Following HypeNET, for a path \( p \) composed of edges \( e_1, \ldots, e_k \), we represent each edge by the concatenation of its lemma, part-of-speech tag, dependency label and direction vectors: \( \vec{v}_e = [\vec{v}_1, \vec{v}_{pos}, \vec{v}_{dep}, \vec{v}_{dir}] \). The edge vectors \( \vec{v}_{e_1}, \ldots, \vec{v}_{e_k} \) are encoded using an LSTM (Hochreiter and Schmidhuber, 1997), and the last output vector \( \vec{p} \) is used as the path embedding.

We use the NC labels as distant supervision. While HypeNET predicts a word pair’s label from the frequency-weighted average of the path vectors, we differ from it slightly and compute the label from the frequency-weighted average of the predictions obtained from each path separately:

\[
\vec{o} = \frac{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)} \cdot \text{softmax}(\vec{p})}{\sum_{p \in P(w_1,w_2)} f_{p,(w_1,w_2)}}
\]

\[
r = \text{argmax}_i \vec{o}_i
\]

We conjecture that label distribution averaging allows for more efficient training of path embeddings when a single NC contains multiple paths.

### 4 Evaluation

#### 4.1 Dataset

We follow Dima (2016) and evaluate on the Tratz (2011) dataset, with 19,158 instances and two levels of labels: fine-grained (tratz-fine, 37 relations) and coarse-grained (tratz-coarse, 12 relations). We report results on both versions. See Tratz (2011) for the list of relations.

**Dataset Splits**  Dima (2016) showed that a classifier based only on \( v_{w_1} \) and \( v_{w_2} \) performs on par with compound representations, and that the success comes from lexical memorization (Levy et al., 2015): memorizing the majority label of single words in particular slots of the compound (e.g. *TOPIC for travel guide, fishing guide*, etc.). This memorization paints a skewed picture of the state-of-the-art performance on this difficult task.

To better test this hypothesis, we evaluate on 4 different splits of the datasets to train, test, and validation sets: (1) **random**, in a 75:20:5 ratio, (2) **lexical-full**, in which the train, test, and validation
Table 1: All methods’ performance ($F_1$) on the various splits. best freq: best performing frequency baseline (head / modifier),\(^3\) best comp: best model from Dima (2016).

Table 2: Number of instances in each dataset split.

sets each consists of a distinct vocabulary. The split was suggested by Levy et al. (2015), and it randomly assigns words to distinct sets, such that for example, including travel guide in the train set promises that fishing guide would not be included in the test set, and the models do not benefit from memorizing that the head guide is always annotated as TOPIC. Given that the split discards many NCs, we experimented with two additional splits: (3) lexical-mod split, in which the $w_1$ words are unique in each set, and (4) lexical-head split, in which the $w_2$ words are unique in each set. Table 2 displays the sizes of each split.

4.2 Baselines

Frequency Baselines. mod freq classifies $w_1 w_2$ to the most common relation in the train set for NCs with the same modifier ($w_1 w_2'$), while head freq considers NCs with the same head ($w_1' w_2$).\(^4\)

Distributional Baselines. Ablation of the path-based component from our models: Dist uses only $w_1$ and $w_2$’s word embeddings: $\vec{x} = [\vec{v}_{w_1}, \vec{v}_{w_2}]$, while Dist-NC includes also the NC embedding: $\vec{x} = [\vec{v}_{w_1}, \vec{v}_{w_2}, \vec{v}_{nc}]$. The network architecture is defined similarly to our models (Section 3.1).

Compositional Baselines. We re-train Dima’s (2016) models, various combinations of NC representations (Zanzotto et al., 2010; Socher et al., 2012) and single word embeddings in a fully connected network.\(^5\)

4.3 Results

Table 1 shows the performance of various methods on the datasets. Dima’s (2016) compositional models perform best among the baselines, and on the random split, better than all the methods. On the lexical splits, however, the baselines exhibit a dramatic drop in performance, and are outperformed by our methods. The gap is larger in the lexical-full split. Finally, there is usually no gain from the added NC vector in Dist-NC and Integrated-NC.

5 Analysis

Path Embeddings. To focus on the changes from previous work, we analyze the performance of the path-based model on the Tratz-fine random split. This dataset contains 37 relations and the model performance varies across them. Some relations, such as measure and personal_title yield reasonable performance ($F_1$ score of 0.87 and 0.68). Table 3 focuses on these relations and illustrates the indicative paths that the model has learned for each relation. We compute these by performing the analysis in Shwartz et al. (2016), where each path is fed into the path-based model, and is assigned to its best-scoring relation. For each relation, we consider paths with a score $\geq 0.8$.

Other relations achieve very low $F_1$ scores, indicating that the model is unable to learn them at all. Interestingly, the four relations with the lowest performance in our model\(^6\) are also those with the highest error rate in Dima (2016), very
likely since they express complex relations. For example, the LEXICALIZED relation contains non-compositional NCs (soap opera) or lexical items whose meanings departed from the combination of the constituent meanings. It is expected that there are no paths that indicate lexicalization. In PARTIAL_ATTRIBUTE_TRANSFER (bullet train), \( w_1 \) transfers an attribute to \( w_2 \) (e.g., bullet transfers speed to train). These relations are not expected to be expressed in text, unless the text aims to explain them (e.g. train as fast as a bullet).

Looking closer at the model confusions shows that it often defaulted to general relations like OBJECTIVE (recovery plan) or RELATIONAL-NOUN-COMPLEMENT (eye shape). The latter is described as “indicating the complement of a relational noun (e.g., son of, price of)”, and the indicative paths for this relation indeed contain many variants of “[\( w_2 \)] of [\( w_1 \)]”, which potentially can occur with NCs in other relations. The model also confused between relations with subtle differences, such as the different topic relations. Given that these relations were conflated to a single relation in the inter-annotator agreement computation in Tratz and Hovy (2010), we can conjecture that even humans find it difficult to distinguish between them.

**NC Embeddings.** To understand why the NC embeddings did not contribute to the classification, we looked into the embeddings of the Tratz-fine test NCs; 3091/3831 (81%) of them had embeddings. For each NC, we looked for the 10 most similar NC vectors (in terms of cosine similarity), and compared their labels. We have found that only 27.61% of the NCs were mostly similar to NCs with the same label. The problem seems to be inconsistency of annotations rather than low embeddings quality. Table 4 displays some examples of NCs from the test set, along with their most similar NC in the embeddings, where the two NCs have different labels.

### Table 3: Indicative paths for selected relations, along with NC examples.

<table>
<thead>
<tr>
<th>Relation Path</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEASURE ( [w_2] ) varies by ( [w_1] ) state limit, age limit</td>
<td>( 2,560 [w_1] ) portion of ( [w_2] ) acre estate</td>
</tr>
<tr>
<td>PERSONAL TITLE ( [w_2] ) ( [w_1] ) Anderson, ( [w_2] ) Sheridan</td>
<td>( [w_1] ) Mrs. Brown, ( [w_1] ) Gen. Johnson</td>
</tr>
<tr>
<td>CREATE-PROVIDE-GENERATE-SELL ( [w_2] ) produce, ( [w_1] ) food producer, drug group</td>
<td>( [w_1] ) manufacture, ( [w_1] ) engine plant, sugar company</td>
</tr>
<tr>
<td>TIME-OF ( [w_1] ) ( [w_2] ) begin, ( [w_1] ) morning program</td>
<td>( [w_1] ) held Saturday, ( [w_1] ) afternoon meeting, morning session</td>
</tr>
<tr>
<td>SUBSTANCE-MATERIAL-INGREDIENT ( [w_1] ) ( [w_2] ) made of wood, ( [w_1] ) marble table, ( [w_1] ) vinyl siding</td>
<td>( [w_1] ) material includes type of ( [w_1] ) steel pipe</td>
</tr>
</tbody>
</table>

### Table 4: Example of NCs from the Tratz-fine random split test set, along with the most similar NC in the embeddings, where the two NCs have different labels.

<table>
<thead>
<tr>
<th>Test NC</th>
<th>Label</th>
<th>Most Similar NC</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>majority party</td>
<td>EQUIVATIVE</td>
<td>minority party</td>
<td>WHOLE+PART+MEMBER+OF</td>
</tr>
<tr>
<td>enforcement director</td>
<td>OBJECTIVE</td>
<td>enforcement chief</td>
<td>PERFORM+ENGAGE+IN</td>
</tr>
<tr>
<td>fire investigator</td>
<td>OBJECTIVE</td>
<td>fire marshal</td>
<td>ORGANIZE+SUPERVISE+AUTHORITY</td>
</tr>
<tr>
<td>stabilization plan</td>
<td>OBJECTIVE</td>
<td>stabilization program</td>
<td>PERFORM+ENGAGE+IN</td>
</tr>
<tr>
<td>investor sentiment</td>
<td>EXPERIENCER-OF-EXPERIENCE</td>
<td>market sentiment</td>
<td>TOPIC+OF+COGNITION+EMOTION</td>
</tr>
<tr>
<td>alliance member</td>
<td>WHOLE+PART+MEMBER+OF</td>
<td>alliance leader</td>
<td>OBJECTIVE</td>
</tr>
</tbody>
</table>

6 Conclusion

We used an existing neural dependency path representation to represent noun-compound paraphrases, and along with distributional information applied it to the NC classification task. Following previous work, that suggested that distributional methods succeed due to lexical memorization, we show that when lexical memorization is not possible, the performance of all methods is much worse. Adding the path-based component helps mitigate this issue and increase performance.

Acknowledgments

We would like to thank Marius Pasca, Susanne Riehemann, Colin Evans, Octavian Ganea, and Xiang Li for the fruitful conversations, and Corina Dima for her help in running the compositional baselines.
References


A Technical Details

To extract paths, we use a concatenation of English Wikipedia and the Gigaword corpus.\(^7\) We consider sentences with up to 32 words and dependency paths with up to 8 edges, including satellites, and keep only 1,000 paths for each noun-compound. We compute the path embeddings in advance for all the paths connecting NCs in the dataset (§3.2), and then treat them as fixed embeddings during classification (§3.1).

We use TensorFlow (Abadi et al., 2016) to train the models, fixing the values of the hyperparameters after performing preliminary experiments on the validation set. We set the mini-batch size to 10, use Adam optimizer (Kingma and Ba, 2014) with the default learning rate, and apply word dropout with probability 0.1. We train up to 30 epochs with early stopping, stopping the training when the $F_1$ score on the validation set drops 8 points below the best performing score.

We initialize the distributional embeddings with the 300-dimensional pre-trained GloVe embeddings (Pennington et al., 2014) and the lemma embeddings (for the path-based component) with the 50-dimensional ones. Unlike HypeNET, we do not update the embeddings during training. The lemma, POS, and direction embeddings are initialized randomly and updated during training. NC embeddings are learned using a concatenation of Wikipedia and Gigaword. Similarly to the original GloVe implementation, we only keep the most frequent 400,000 vocabulary terms, which means that roughly 20% of the noun-compounds do not have vectors and are initialized randomly in the model.

\(^7\)https://catalog.ldc.upenn.edu/ldc2003t05
Chapter 8

Paraphrase to Explicate:
Revealing Implicit
Noun-Compound Relations
Paraphrase to Explicate: Revealing Implicit Noun-Compound Relations

Vered Shwartz
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
vered1986@gmail.com

Ido Dagan
Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
dagan@cs.biu.ac.il

Abstract

Revealing the implicit semantic relation between the constituents of a noun-compound is important for many NLP applications. It has been addressed in the literature either as a classification task to a set of pre-defined relations or by producing free text paraphrases explicating the relations. Most existing paraphrasing methods lack the ability to generalize, and have a hard time interpreting infrequent or new noun-compounds. We propose a neural model that generalizes better by representing paraphrases in a continuous space, generalizing for both unseen noun-compounds and rare paraphrases. Our model helps improving performance on both the noun-compound paraphrasing and classification tasks.

1 Introduction

Noun-compounds hold an implicit semantic relation between their constituents. For example, a ‘birthday cake’ is a cake eaten on a birthday, while ‘apple cake’ is a cake made of apples. Interpreting noun-compounds by explicating the relationship is beneficial for many natural language understanding tasks, especially given the prevalence of noun-compounds in English (Nakov, 2013).

The interpretation of noun-compounds has been addressed in the literature either by classifying them to a fixed inventory of ontological relationships (e.g. Nastase and Szpakowicz, 2003) or by generating various free text paraphrases that describe the relation in a more expressive manner (e.g. Hendrickx et al., 2013).

Methods dedicated to paraphrasing noun-compounds usually rely on corpus co-occurrences of the compound’s constituents as a source of explicit relation paraphrases (e.g. Wubben, 2010; Versley, 2013). Such methods are unable to generalize for unseen noun-compounds. Yet, most noun-compounds are very infrequent in text (Kim and Baldwin, 2007), and humans easily interpret the meaning of a new noun-compound by generalizing existing knowledge. For example, consider interpreting parsley cake as a cake made of parsley vs. resignation cake as a cake eaten to celebrate quitting an unpleasant job.

We follow the paraphrasing approach and propose a semi-supervised model for paraphrasing noun-compounds. Differently from previous methods, we train the model to predict either a paraphrase expressing the semantic relation of a noun-compound (predicting ‘[w₂] made of [w₁]’ given ‘apple cake’), or a missing constituent given a combination of paraphrase and noun-compound (predicting ‘apple’ given ‘cake made of [w₁]’). Constituents and paraphrase templates are represented as continuous vectors, and semantically-similar paraphrase templates are embedded in proximity, enabling better generalization. Interpreting ‘parsley cake’ effectively reduces to identifying paraphrase templates whose “selectional preferences” (Pantel et al., 2007) on each constituent fit ‘parsley’ and ‘cake’.

A qualitative analysis of the model shows that the top ranked paraphrases retrieved for each noun-compound are plausible even when the constituents never co-occur (Section 4). We evaluate our model on both the paraphrasing and the classification tasks (Section 5). On both tasks, the model’s ability to generalize leads to improved performance in challenging evaluation settings.¹

¹The code is available at github.com/vered1986/panic
2 Background

2.1 Noun-compound Classification

Noun-compound classification is the task concerned with automatically determining the semantic relation that holds between the constituents of a noun-compound, taken from a set of pre-defined relations.

Early work on the task leveraged information derived from lexical resources and corpora (e.g. Girju, 2007; Ó Séaghdha and Copestake, 2009; Tratz and Hovy, 2010). More recent work broke the task into two steps: in the first step, a noun-compound representation is learned from the distributional representation of the constituent words (e.g. Mitchell and Lapata, 2010; Zanzotto et al., 2010; Socher et al., 2012). In the second step, the noun-compound representations are used as feature vectors for classification (e.g. Dima and Hinrichs, 2015; Dima, 2016).

The datasets for this task differ in size, number of relations and granularity level (e.g. Nastase and Szpakowicz, 2003; Kim and Baldwin, 2007; Tratz and Hovy, 2010). The decision on the relation inventory is somewhat arbitrary, and subsequently, the inter-annotator agreement is relatively low (Kim and Baldwin, 2007). Specifically, a noun-compound may fit into more than one relation: for instance, in Tratz (2011), business zone is labeled as contained (zone contains business), although it could also be labeled as purpose (zone whose purpose is business).

2.2 Noun-compound Paraphrasing

As an alternative to the strict classification to pre-defined relation classes, Nakov and Hearst (2006) suggested that the semantics of a noun-compound could be expressed with multiple prepositional and verbal paraphrases. For example, apple cake is a cake from, made of, or which contains apples.

The suggestion was embraced and resulted in two SemEval tasks. SemEval 2010 task 9 (Butnariu et al., 2009) provided a list of plausible human-written paraphrases for each noun-compound, and systems had to rank them with the goal of high correlation with human judgments. In SemEval 2013 task 4 (Hendrickx et al., 2013), systems were expected to provide a ranked list of paraphrases extracted from free text.

Various approaches were proposed for this task. Most approaches start with a pre-processing step of extracting joint occurrences of the constituents from a corpus to generate a list of candidate paraphrases. Unsupervised methods apply information extraction techniques to find and rank the most meaningful paraphrases (Kim and Nakov, 2011; Xavier and Lima, 2014; Pasca, 2015; Pavlick and Pasca, 2017), while supervised approaches learn to rank paraphrases using various features such as co-occurrence counts (Wubben, 2010; Li et al., 2010; Surtani et al., 2013; Versley, 2013) or the distributional representations of the noun-compounds (Van de Cruys et al., 2013).

One of the challenges of this approach is the ability to generalize. If one assumes that sufficient paraphrases for all noun-compounds appear in the corpus, the problem reduces to ranking the existing paraphrases. It is more likely, however, that some noun-compounds do not have any paraphrases in the corpus or have just a few. The approach of Van de Cruys et al. (2013) somewhat generalizes for unseen noun-compounds. They represented each noun-compound using a compositional distributional vector (Mitchell and Lapata, 2010) and used it to predict paraphrases from the corpus. Similar noun-compounds are expected to have similar distributional representations and therefore yield the same paraphrases. For example, if the corpus does not contain paraphrases for plastic spoon, the model may predict the paraphrases of a similar compound such as steel knife.

In terms of sharing information between semantically-similar paraphrases, Nulty and Costello (2010) and Surtani et al. (2013) learned “is-a” relations between paraphrases from the co-occurrences of various paraphrases with each other. For example, the specific ‘[w2] extracted from [w1]’ template (e.g. in the context of olive oil) generalizes to ‘[w2] made from [w1]’. One of the drawbacks of these systems is that they favor more frequent paraphrases, which may co-occur with a wide variety of more specific paraphrases.

2.3 Noun-compounds in other Tasks

Noun-compound paraphrasing may be considered as a subtask of the general paraphrasing task, whose goal is to generate, given a text fragment, additional texts with the same meaning. However, general paraphrasing methods do not guarantee to explicate implicit information conveyed in the original text. Moreover, the most notable source for extracting paraphrases is multiple translations of the same text (Barzilay and McKeown, 2010).
Figure 1: An illustration of the model predictions for \( w_1 \) and \( p \) given the triplet \((\text{cake}, \text{made of}, \text{apple})\). The model predicts each component given the encoding of the other two components, successfully predicting ‘apple’ given ‘cake made of \([w_1]\)’, while predicting ‘\([w_2]\) containing \([w_1]\)’ for ‘cake \([p]\) apple’.

Section 3.2 details the creation of training data, and Section 3.3 describes the model.

### 3.1 Multi-task Reformulation

Each training example consists of two constituents and a paraphrase \((w_2, p, w_1)\), and we train the model on 3 subtasks: (1) predict \( p \) given \( w_1 \) and \( w_2 \), (2) predict \( w_1 \) given \( p \) and \( w_2 \), and (3) predict \( w_2 \) given \( p \) and \( w_1 \). Figure 1 demonstrates the predictions for subtasks (1) (right) and (2) (left) for the training example \((\text{cake}, \text{made of}, \text{apple})\). Effectively, the model is trained to answer questions such as “what can cake be made of?”, “what can be made of apple?”, and “what are the possible relationships between cake and apple?”.

The multi-task reformulation helps learning better representations for paraphrase templates, by embedding semantically-similar paraphrases in proximity. Similarity between paraphrases stems either from lexical similarity and overlap between the paraphrases (e.g. “is made of” and “made of”), or from shared constituents, e.g. \([w_2]\) involved in \([w_1]\)” and “\([w_2]\) in \([w_1]\) industry’ can share \([w_1]\) = \text{insurance} and \([w_2]\) = \text{company}. This allows the model to predict a correct paraphrase for a given noun-compound, even when the constituents do not occur with that paraphrase in the corpus.

### 3.2 Training Data

We collect a training set of \((w_2, p, w_1, s)\) examples, where \(w_1\) and \(w_2\) are constituents of a noun-compound \(w_1w_2\), \(p\) is a templated paraphrase, and \(s\) is the score assigned to the training instance.\(^2\)

\(^2\)We refer to “paraphrases” and “paraphrase templates” interchangeably. In the extracted templates, \([w_2]\) always precedes \([w_1]\), probably because \([w_2]\) is normally the head noun.
We use the 19,491 noun-compounds found in the SemEval tasks datasets (Butnariu et al., 2009; Hendrickx et al., 2013) and in Tratz (2011). To extract patterns of part-of-speech tags that can form noun-compound paraphrases, such as ‘[w2] VERB PREP [w1]’, we use the SemEval task training data, but we do not use the lexical information in the gold paraphrases.

Corpus. Similarly to previous noun-compound paraphrasing approaches, we use the Google N-gram corpus (Brants and Franz, 2006) as a source of paraphrases (Wubben, 2010; Li et al., 2010; Surtani et al., 2013; Versley, 2013). The corpus consists of sequences of n terms (for \( n \in \{3, 4, 5\} \)) that occur more than 40 times on the web. We search for n-grams following the extracted patterns and containing \( w_1 \) and \( w_2 \)’s lemmas for some noun-compound in the set. We remove punctuation, adjectives, adverbs, and some determiners to unite similar paraphrases. For example, from the 5-gram ‘cake made of sweet apples’ we extract the training example ‘cake, made of, apple’.

We keep only paraphrases that occurred at least 5 times, resulting in 136,609 instances.

Weighting. Each n-gram in the corpus is accompanied with its frequency, which we use to assign scores to the different paraphrases. For instance, ‘cake of apples’ may also appear in the corpus, although with lower frequency than ‘cake from apples’. As also noted by Surtani et al. (2013), the shortcoming of such a weighting mechanism is that it prefers shorter paraphrases, which are much more common in the corpus (e.g. \( \text{count}('\text{cake of apples}') \ll \text{count}('\text{cake from apples}') \)). We overcome this by normalizing the frequencies for each paraphrase length, creating a distribution of paraphrases in a given length.

Negative Samples. We add 1% of negative samples by selecting random corpus words \( w_1 \) and \( w_2 \) that do not co-occur, and adding an example \((w_2, [w_2])\) unrelated to \([w_1], w_1, s_n\) for some predefined negative sample score \( s_n\). Similarly, for a word \( w_1 \) that did not occur in a paraphrase \( p \) we add \((w_1, p, \text{UNK}, s_n)\) or \((\text{UNK}, p, w_1, s_n)\), where UNK is the unknown word. This may help the model deal with non-compositional noun-compounds, where \( w_1 \) and \( w_2 \) are unrelated, rather than forcibly predicting some relation between them.

3.3 Model

For a training instance \((w_2, p, w_1, s)\), we predict each item given the encoding of the other two.

Encoding. We use the 100-dimensional pre-trained GloVe embeddings (Pennington et al., 2014), which are fixed during training. In addition, we learn embeddings for the special words \([w_1], [w_2], \) and \([p]\), which are used to represent a missing component, as in “cake made of [\( w_1 \)]”, “[\( w_2 \)] made of apple”, and “cake [\( p \)] apple”.

For a missing component \( x \in \{[p], [w_1], [w_2]\} \) surrounded by the sequences of words \( v_{1:i-1} \) and \( v_{i+1:n} \), we encode the sequence using a bidirectional long-short term memory (bi-LSTM) network (Graves and Schmidhuber, 2005), and take the \( i \)-th output vector as representing the missing component: \( bLS(v_{1:i}, x, v_{i+1:n}) \).

In bi-LSTMs, each output vector is a concatenation of the outputs of the forward and backward LSTMs, so the output vector is expected to contain information on valid substitutions both with respect to the previous words \( v_{1:i-1} \) and the subsequent words \( v_{i+1:n} \).

Prediction. We predict a distribution of the vocabulary of the missing component, i.e. to predict \( w_1 \) correctly we need to predict its index in the word vocabulary \( V_w \), while the prediction of \( p \) is from the vocabulary of paraphrases in the training set, \( V_p \).

We predict the following distributions:

\[
\hat{p} = \text{softmax}(W_p \cdot bLS(\vec{w}_2, [p], \vec{w}_1)_2) \\
\hat{w}_1 = \text{softmax}(W_w \cdot bLS(\vec{w}_2, \vec{p}_{1:n}, [w_1])_{n+1}) \\
\hat{w}_2 = \text{softmax}(W_w \cdot bLS([w_2], \vec{p}_{1:n}, \vec{w}_1))
\]

where \( W_w \in \mathcal{R}^{|V_w| \times 2d}, W_p \in \mathcal{R}^{|V_p| \times 2d} \), and \( d \) is the embeddings dimension.

During training, we compute cross-entropy loss for each subtask using the gold item and the prediction, sum up the losses, and weight them by the instance score. During inference, we predict the missing components by picking the best scoring index in each distribution:

\[
\hat{p}_i = \text{argmax}(\hat{p}) \\
\hat{w}_{1i} = \text{argmax}(\hat{w}_1) \\
\hat{w}_{2i} = \text{argmax}(\hat{w}_2)
\]

The subtasks share the pre-trained word embeddings, the special embeddings, and the bi-LSTM parameters. Subtasks (2) and (3) also share \( W_w \), the MLP that predicts the index of a word.

\footnote{In practice, we pick the \( k \) best scoring indices in each distribution for some predefined \( k \), as we discuss in Section 5.}
The projection positions semantically-similar but lexically-divergent paraphrases in proximity, likely due to ...

Table 1: Examples of top ranked predicted components using the model: predicting the paraphrase given $w_1$ and $w_2$ (left), $w_1$ given $w_2$ and the paraphrase (middle), and $w_2$ given $w_1$ and the paraphrase (right).

![Figure 2: A t-SNE map of a sample of paraphrases, using the paraphrase vectors encoded by the biLSTM, for example $bLS([w_2] \text{ made of } [w_1])$.](image)

Implementation Details. The model is implemented in DyNet (Neubig et al., 2017). We dedicate a small number of noun-compounds from the corpus for validation. We train for up to 10 epochs, stopping early if the validation loss has not improved in 3 epochs. We use Momentum SGD (Nesterov, 1983), and set the batch size to 10 and the other hyper-parameters to their default values.

4 Qualitative Analysis

To estimate the quality of the proposed model, we first provide a qualitative analysis of the model outputs. Table 1 displays examples of the model outputs for each possible usage: predicting the paraphrase given the constituent words, and predicting each constituent word given the paraphrase and the other word.

The examples in the table are from among the top 10 ranked predictions for each component-pair. We note that most of the $(w_2, \text{paraphrase}, w_1)$ triplets in the table do not occur in the training data, but are rather generalized from similar examples. For example, there is no training instance for “company in the software industry” but there is a “firm in the software industry” and a company in many other industries.

While the frequent prepositional paraphrases are often ranked at the top of the list, the model also retrieves more specified verbal paraphrases. The list often contains multiple semantically-similar paraphrases, such as ‘$[w_2]$ involved in $[w_1]$ industry’ and ‘$[w_2]$ in $[w_1]$ industry’. This is a result of the model training objective (Section 3) which positions the vectors of semantically-similar paraphrases close to each other in the embedding space, based on similar constituents.

To illustrate paraphrase similarity we compute a t-SNE projection (Van Der Maaten, 2014) of the embeddings of all the paraphrases, and draw a sample of 50 paraphrases in Figure 2. The projection positions semantically-similar but lexically-divergent paraphrases in proximity, likely due to
many shared constituents. For instance, ‘with’, ‘from’, and ‘out of’ can all describe the relation between food words and their ingredients.

5 Evaluation: Noun-Compound Interpretation Tasks

For quantitative evaluation we employ our model for two noun-compound interpretation tasks. The main evaluation is on retrieving and ranking paraphrases (§5.1). For the sake of completeness, we also evaluate the model on classification to a fixed inventory of relations (§5.2), although it wasn’t designed for this task.

5.1 Paraphrasing

**Task Definition.** The general goal of this task is to interpret each noun-compound to multiple prepositional and verbal paraphrases. In SemEval 2013 Task 4, the participating systems were asked to retrieve a ranked list of paraphrases for each noun-compound, which was automatically evaluated against a similarly ranked list of paraphrases proposed by human annotators.

**Model.** For a given noun-compound $w_1w_2$, we first predict the $k = 250$ most likely paraphrases: $\hat{p}_1,\ldots,\hat{p}_k = \arg\max_k \hat{p}$, where $\hat{p}$ is the distribution of paraphrases defined in Equation 1.

While the model also provides a score for each paraphrase (Equation 1), the scores have not been optimized to correlate with human judgments. We therefore developed a re-ranking model that receives a list of paraphrases and re-ranks the list to better fit the human judgments.

We follow Herbrich (2000) and learn a pairwise ranking model. The model determines which of two paraphrases of the same noun-compound should be ranked higher, and it is implemented as an SVM classifier using scikit-learn (Pedregosa et al., 2011). For training, we use the available training data with gold paraphrases and ranks provided by the SemEval task organizers. We extract the following features for a paraphrase $p$:

1. The part-of-speech tags contained in $p$
2. The prepositions contained in $p$
3. The number of words in $p$
4. Whether $p$ ends with the special $[w_1]$ symbol
5. $\cos(bLS([w_2], p_1; [w_1])_2, V_p^{\hat{p}_i}) \cdot \hat{p}_i$

where $V_p^{\hat{p}_i}$ is the biLSTM encoding of the predicted paraphrase computed in Equation 1 and $\hat{p}_i$ is its confidence score. The last feature incorporates the original model score into the decision, as to not let other considerations such as preposition frequency in the training set take over.

During inference, the model sorts the list of paraphrases retrieved for each noun-compound according to the pairwise ranking. It then scores each paraphrase by multiplying its rank with its original model score, and prunes paraphrases with final score $< 0.025$. The values for $k$ and the threshold were tuned on the training set.

**Evaluation Settings.** The SemEval 2013 task provided a scorer that compares words and n-grams from the gold paraphrases against those in the predicted paraphrases, where agreement on a prefix of a word (e.g. in derivations) yields a partial scoring. The overall score assigned to each system is calculated in two different ways. The ‘isomorphic’ setting rewards both precision and recall, and performing well on it requires accurately reproducing as many of the gold paraphrases as possible, and in much the same order. The ‘non-isomorphic’ setting rewards only precision, and performing well on it requires accurately reproducing the top-ranked gold paraphrases, with no importance to order.

**Baselines.** We compare our method with the published results from the SemEval task. The SemEval 2013 baseline generates for each noun-compound a list of prepositional paraphrases in an arbitrary fixed order. It achieves a moderately good score in the non-isomorphic setting by generating a fixed set of paraphrases which are both common and generic. The MELODI system performs similarly: it represents each noun-compound using a compositional distributional vector (Mitchell and Lapata, 2010) which is then used to predict paraphrases from the corpus. The performance of MELODI indicates that the system was rather conservative, yielding a few common paraphrases rather than many specific ones. SFS and IIITH, on the other hand, show a more balanced trade-off between recall and precision.

As a sanity check, we also report the results of a baseline that retrieves ranked paraphrases from the training data collected in Section 3.2. This baseline has no generalization abilities, therefore it is expected to score poorly on the recall-aware isomorphic setting.
<table>
<thead>
<tr>
<th>Method</th>
<th>isomorphic</th>
<th>non-isomorphic</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFS (Versley, 2013)</td>
<td>23.1</td>
<td>17.9</td>
</tr>
<tr>
<td>IIITH (Surtani et al., 2013)</td>
<td>23.1</td>
<td>25.8</td>
</tr>
<tr>
<td>MELODI (Van de Cruys et al., 2013)</td>
<td>13.0</td>
<td>54.8</td>
</tr>
<tr>
<td>SemEval 2013 Baseline (Hendrickx et al., 2013)</td>
<td>13.8</td>
<td>40.6</td>
</tr>
<tr>
<td>This paper Baseline</td>
<td>3.8</td>
<td>16.1</td>
</tr>
<tr>
<td>Our method</td>
<td>28.2</td>
<td>28.4</td>
</tr>
</tbody>
</table>

Table 2: Results of the proposed method and the baselines on the SemEval 2013 task.

<table>
<thead>
<tr>
<th>Category</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive</td>
<td></td>
</tr>
<tr>
<td>(1) Valid paraphrase missing from gold</td>
<td>44%</td>
</tr>
<tr>
<td>(2) Valid paraphrase, slightly too specific</td>
<td>15%</td>
</tr>
<tr>
<td>(3) Incorrect, common prepositional paraphrase</td>
<td>14%</td>
</tr>
<tr>
<td>(4) Incorrect, other errors</td>
<td>14%</td>
</tr>
<tr>
<td>(5) Syntactic error in paraphrase</td>
<td>8%</td>
</tr>
<tr>
<td>(6) Valid paraphrase, but borderline grammatical</td>
<td>5%</td>
</tr>
<tr>
<td>False Negative</td>
<td></td>
</tr>
<tr>
<td>(1) Long paraphrase (more than 5 words)</td>
<td>30%</td>
</tr>
<tr>
<td>(2) Prepositional paraphrase with determiners</td>
<td>25%</td>
</tr>
<tr>
<td>(3) Inflected constituents in gold</td>
<td>10%</td>
</tr>
<tr>
<td>(4) Other errors</td>
<td>35%</td>
</tr>
</tbody>
</table>

Table 3: Categories of false positive and false negative predictions along with their percentage.

Results. Table 2 displays the performance of the proposed method and the baselines in the two evaluation settings. Our method outperforms all the methods in the isomorphic setting. In the non-isomorphic setting, it outperforms the other two systems that score reasonably on the isomorphic setting (SFS and IIITH) but cannot compete with the systems that focus on achieving high precision.

The main advantage of our proposed model is in its ability to generalize, and that is also demonstrated in comparison to our baseline performance. The baseline retrieved paraphrases only for a third of the noun-compounds (61/181), expectedly yielding poor performance on the isomorphic setting. Our model, which was trained on the very same data, retrieved paraphrases for all noun-compounds. For example, welfare system was not present in the training data, yet the model predicted the correct paraphrases “system of welfare benefits”, “system to provide welfare” and others.

Error Analysis. We analyze the causes of the false positive and false negative errors made by the model. For each error type we sample 10 noun-compounds. For each noun-compound, false positive errors are the top 10 predicted paraphrases which are not included in the gold paraphrases, while false negative errors are the top 10 gold paraphrases not found in the top \( k \) predictions made by the model. Table 3 displays the manually annotated categories for each error type.

Many false positive errors are actually valid paraphrases that were not suggested by the human annotators (error 1, “discussion by group”). Some are borderline valid with minor grammatical changes (error 6, “force of coalition forces”) or too specific (error 2, “life of women in community” instead of “life in community”). Common prepositional paraphrases were often retrieved although they are incorrect (error 3). We conjecture that this error often stem from an n-gram that does not respect the syntactic structure of the sentence, e.g., a sentence such as “rinse away the oil from baby’s head” produces the n-gram “oil from baby”.

With respect to false negative examples, they consisted of many long paraphrases, while our model was restricted to 5 words due to the source of the training data (error 1, “holding done in the case of a share”). Many prepositional paraphrases consisted of determiners, which we conflated with the same paraphrases without determiners (error 2, “mutation of a gene”). Finally, in some paraphrases, the constituents in the gold paraphrase appear in inflectional forms (error 3, “holding of shares” instead of “holding of share”).

5.2 Classification

Noun-compound classification is defined as a multiclass classification problem: given a pre-defined set of relations, classify \( w_1 w_2 \) to the relation that holds between \( w_1 \) and \( w_2 \). Potentially, the corpus co-occurrences of \( w_1 \) and \( w_2 \) may contribute to the classification, e.g. \([w_2]\) held at \([w_1]\) indicates a TIME relation. Tratz and Hovy (2010) included such features in their classifier, but ablation tests showed that these features had a relatively small contribution, probably due to the sparseness of the paraphrases. Recently, Schwartz and Watson (2018) showed that paraphrases may contribute to the classification when represented in a continuous space.
Model. We generate a paraphrase vector representation \( \vec{par}(w_1w_2) \) for a given noun-compound \( w_1w_2 \) as follows. We predict the indices of the \( k \) most likely paraphrases: \( \hat{p}_1, \ldots, \hat{p}_k = \arg\max_{\hat{p}} \hat{p} \), where \( \hat{p} \) is the distribution on the paraphrase vocabulary \( V_p \), as defined in Equation 1. We then encode each paraphrase using the biLSTM, and average the paraphrase vectors, weighted by their confidence scores in \( \hat{p} \):

\[
\hat{\text{par}}(w_1w_2) = \frac{\sum_{i=1}^{k} \hat{p}_i \cdot \vec{V}_p}{\sum_{i=1}^{k} \hat{p}_i} \tag{3}
\]

We train a linear classifier, and represent \( w_1w_2 \) in a feature vector \( f(w_1w_2) \) in two variants: paraphrase: \( f(w_1w_2) = \hat{\text{par}}(w_1w_2) \), or integrated: concatenated to the constituent word embeddings \( f(w_1w_2) = [\hat{\text{par}}(w_1w_2), \vec{w}_1, \vec{w}_2] \). The classifier type (logistic regression/SVM), \( k \), and the penalty are tuned on the validation set. We also provide a baseline in which we ablate the paraphrase component from our model, representing a noun-compound by the concatenation of its constituent embeddings \( f(w_1w_2) = [\vec{w}_1, \vec{w}_2] \) (distributional).

Datasets. We evaluate on the Tratz (2011) dataset, which consists of 19,158 instances, labeled in 37 fine-grained relations (Tratz-fine) or 12 coarse-grained relations (Tratz-coarse).

We report the performance on two different dataset splits to train, test, and validation: a random split in a 75:20:5 ratio, and, following concerns raised by Dima (2016) about lexical memorization (Levy et al., 2015), on a lexical split in which the sets consist of distinct vocabularies. The lexical split better demonstrates the scenario in which a noun-compound whose constituents have not been observed needs to be interpreted based on similar observed noun-compounds, e.g. inferring the relation in pear tart based on apple cake and other similar compounds. We follow the random and full-lexical splits from Shwartz and Waterson (2018).

Baselines. We report the results of 3 baselines representative of different approaches:

1) Feature-based (Tratz and Hovy, 2010): we re-implement a version of the classifier with features from WordNet and Roget’s Thesaurus.

2) Compositional (Dima, 2016): a neural architecture that operates on the distributional representations of the noun-compound and its constituents. Noun-compound representations are learned with the Full-Additive (Zanzotto et al., 2010) and Matrix (Socher et al., 2012) models. We report the results from Shwartz and Waterson (2018).

3) Paraphrase-based (Shwartz and Waterson, 2018): a neural classification model that learns an LSTM-based representation of the joint occurrences of \( w_1 \) and \( w_2 \) in a corpus (i.e. observed paraphrases), and integrates distributional information using the constituent embeddings.

Results. Table 4 displays the methods’ performance on the two versions of the Tratz (2011) dataset and the two dataset splits. The paraphrase model on its own is inferior to the distributional model, however, the integrated version improves upon the distributional model in 3 out of 4 settings, demonstrating the complementary nature of the distributional and paraphrase-based methods. The contribution of the paraphrase component is especially noticeable in the lexical splits.

As expected, the integrated method in Shwartz and Waterson (2018), in which the paraphrase representation was trained with the objective of classification, performs better than our integrated model. The superiority of both integrated models in the lexical splits confirms that paraphrases are beneficial for classification.

<table>
<thead>
<tr>
<th>Dataset &amp; Split</th>
<th>Method</th>
<th>( F_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tratz fine Random</td>
<td>Tratz and Hovy (2010)</td>
<td>0.739</td>
</tr>
<tr>
<td></td>
<td>Dima (2016)</td>
<td>0.725</td>
</tr>
<tr>
<td></td>
<td>Shwartz and Waterson (2018)</td>
<td>0.714</td>
</tr>
<tr>
<td>Tratz fine Lexical</td>
<td>Tratz and Hovy (2010)</td>
<td>0.677</td>
</tr>
<tr>
<td></td>
<td>Dima (2016)</td>
<td>0.505</td>
</tr>
<tr>
<td></td>
<td>Shwartz and Waterson (2018)</td>
<td>0.673</td>
</tr>
<tr>
<td>Tratz coarse Random</td>
<td>Tratz and Hovy (2010)</td>
<td>0.340</td>
</tr>
<tr>
<td></td>
<td>Dima (2016)</td>
<td>0.334</td>
</tr>
<tr>
<td></td>
<td>Shwartz and Waterson (2018)</td>
<td>0.429</td>
</tr>
<tr>
<td>Tratz coarse Lexical</td>
<td>Tratz and Hovy (2010)</td>
<td>0.764</td>
</tr>
<tr>
<td></td>
<td>Dima (2016)</td>
<td>0.775</td>
</tr>
<tr>
<td></td>
<td>Shwartz and Waterson (2018)</td>
<td>0.736</td>
</tr>
<tr>
<td>Tratz coarse Lexical</td>
<td>distributional paraphrase integrated</td>
<td>0.689</td>
</tr>
<tr>
<td></td>
<td>distributional integrated</td>
<td>0.557</td>
</tr>
<tr>
<td></td>
<td>integrated</td>
<td>0.700</td>
</tr>
</tbody>
</table>

Table 4: Classification results. For each dataset split, the top part consists of baseline methods and the bottom part contains methods from this paper. The best performance in each part appears in bold.
**Table 5:** Examples of noun-compounds that were correctly classified by the integrated model while being incorrectly classified by distributional, along with top ranked indicative paraphrases.

**Analysis.** To analyze the contribution of the paraphrase component to the classification, we focused on the differences between the distributional and integrated models on the Tratz-Coarse lexical split. Examination of the per-relation $F_1$ scores revealed that the relations for which performance improved the most in the integrated model were **topical** (+1.1 $F_1$ points), **objective** (+5.5), **attribute** (+3.8) and **location/part-whole** (+3.5).

Table 5 provides examples of noun-compounds that were correctly classified by the integrated model while being incorrectly classified by the distributional model. For each noun-compound, we provide examples of top ranked indicative paraphrases which are indicative of the gold label relation.

### 6 Compositionality Analysis

Our paraphrasing approach at its core assumes compositionality: only a noun-compound whose meaning is derived from the meanings of its constituent words can be rephrased using them. In §3.2 we added negative samples to the training data to simulate non-compositional noun-compounds, which are included in the classification dataset (§5.2). We assumed that these compounds, more often than compositional ones would consist of unrelated constituents (spelling bee, sacred cow), and added instances of random unrelated nouns with ‘[w2] is unrelated to [w1]’. Here, we assess whether our model succeeds to recognize non-compositional noun-compounds.

We used the compositionality dataset of Reddy et al. (2011) which consists of 90 noun-compounds along with human judgments about their compositionality in a scale of 0-5, 0 being non-compositional and 5 being compositional. For each noun-compound in the dataset, we predicted the 15 best paraphrases and analyzed the errors. The most common error was predicting paraphrases for idiomatic compounds which may have a plausible concrete interpretation or which originated from one. For example, it predicted that silver spoon is simply a spoon made of silver and that monkey business is a business that buys or raises monkeys. In other cases, it seems that the strong prior on one constituent leads to ignoring the other, unrelated constituent, as in predicting ‘wedding made of diamond’. Finally, the ‘unrelated’ paraphrase was predicted for a few compounds, but those are not necessarily non-compositional (application form, head teacher). We conclude that the model does not address compositionality and suggest to apply it only to compositional compounds, which may be recognized using compositionality prediction methods as in Reddy et al. (2011).

### 7 Conclusion

We presented a new semi-supervised model for noun-compound paraphrasing. The model differs from previous models by being trained to predict both a paraphrase given a noun-compound, and a missing constituent given the paraphrase and the other constituent. This results in better generalization abilities, leading to improved performance in two noun-compound interpretation tasks. In the future, we plan to take generalization one step further, and explore the possibility to use the biLSTM for generating completely new paraphrase templates unseen during training.

**Acknowledgments**

This work was supported in part by an Intel ICRI-CI grant, the Israel Science Foundation grant 1951/17, the German Research Foundation through the German–Israeli Project Cooperation (DIP, grant DA 1600/1-1), and Theo Hoffenberg. Vered is also supported by the Clore Scholars Programme (2017), and the AI2 Key Scientific Challenges Program (2017).
References


Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1.


Chapter 9

Adding Context to Semantic Data-Driven Paraphrasing
Adding Context to Semantic Data-Driven Paraphrasing

Vered Shwartz  Ido Dagan
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
vered1986@gmail.com dagan@cs.biu.ac.il

Abstract

Recognizing lexical inferences between pairs of terms is a common task in NLP applications, which should typically be performed within a given context. Such context-sensitive inferences have to consider both term meaning in context as well as the fine-grained relation holding between the terms. Hence, to develop suitable lexical inference methods, we need datasets that are annotated with fine-grained semantic relations in-context. Since existing datasets either provide out-of-context annotations or refer to coarse-grained relations, we propose a methodology for adding context-sensitive annotations. We demonstrate our methodology by applying it to phrase pairs from PPDB 2.0, creating a novel dataset of fine-grained lexical inferences in-context and showing its utility in developing context-sensitive methods.

1 Introduction

Recognizing lexical inference is an essential component in semantic tasks. In question answering, for instance, identifying that broadcast and air are synonymous enables answering the question “When was ‘Friends’ first aired?” given the text “‘Friends’ was first broadcast in 1994”. Semantic relations such as synonymy (tall, high) and hyponymy (cat, pet) are used to infer the meaning of one term from another, in order to overcome lexical variability.

In semantic tasks, such terms appear within corresponding contexts, thus making two aspects necessary in order to correctly apply inferences: First, the meaning of each term should be considered within its context (Szpektor et al., 2007), e.g., play entails compete in certain contexts, but not in the context of playing the national anthem at a sports competition. Second, the soundness of inferences within context is conditioned on the fine-grained semantic relation that holds between the terms, as studied within natural logic (MacCartney and Manning, 2007). For instance, in upward-monotone sentences a term entails its hypernym (“my iPhone’s battery is low” ⇒ “my phone’s battery is low”), while in downward monotone ones it entails its hyponym (“talking on the phone is prohibited” ⇒ “talking on the iPhone is prohibited”).

Accordingly, developing algorithms that properly apply lexical inferences in context requires datasets in which inferences are annotated in-context by fine-grained semantic relations. Yet, such a dataset is not available (see 2.1). Most existing datasets provide out-of-context annotations, while the few available in-context annotations refer to coarse-grained relations, such as relatedness or similarity.

In recent years, the PPDB paraphrase database (Ganitkevitch et al., 2013) became a popular resource among semantic tasks, such as monolingual alignment (Sultan et al., 2014) and recognizing textual entailment (Noh et al., 2015). Recently, Pavlick et al. (2015) classified each paraphrase pair to the fine-grained semantic relation that holds between the phrases, following natural logic (MacCartney and Manning, 2007). To that end, a subset of PPDB paraphrase-pairs were manually annotated, forming a fine-grained lexical inference dataset. Yet, annotations are given out-of-context, limiting its utility.

In this paper, we aim to fill the current gap in the inventory of lexical inference datasets, and present a methodology for adding context to out-of-context datasets. We apply our methodology on a subset of phrase pairs from Pavlick et al. (2015),...
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>contexts</th>
<th>out-of-context relation</th>
<th>in-context relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>piece</td>
<td>strip</td>
<td>Roughly 1,500 gold and silver <strong>pieces</strong> were found and the hoard contains roughly 5kgs of gold and 2.5kgs of silver. A huge political storm has erupted around Australia after labor leader Kevin Rudd was found to have gone to a <strong>strip</strong> club during a taxpayer funded trip.</td>
<td>Equivalence</td>
<td>Independent</td>
</tr>
<tr>
<td>competition</td>
<td>race</td>
<td>Three countries withdrew from the <strong>competition:</strong> Germany, Spain and Switzerland. Morgan Tsvangirai, the leader of the Movement for Democratic Change (MDC), Zimbabwe’s main opposition party, has said that he will pull out of the <strong>race</strong> to become the president of Zimbabwe.</td>
<td>Reverse Entailment</td>
<td>Equivalence</td>
</tr>
<tr>
<td>boy</td>
<td>family</td>
<td>The birth of the <strong>boy</strong>, whose birth name is disputed among different sources, is considered very important in the entertainment world. Bill will likely disrupt the Obama <strong>family</strong>’s vacation to Martha’s Vineyard.</td>
<td>Forward Entailment</td>
<td>Other-related</td>
</tr>
<tr>
<td>jump</td>
<td>walk</td>
<td>Amid wild scenes of joy on the pitch he <strong>jumped</strong> onto the podium and lifted the trophy, the fourth of Italy’s history. In a game about rescuing hostages a hero might <strong>walk</strong> past Coca-Cola machine’s one week and Pepsi the next.</td>
<td>Other-related</td>
<td>Alternation</td>
</tr>
</tbody>
</table>

| 1 | piece | strip | Roughly 1,500 gold and silver **pieces** were found and the hoard contains roughly 5kgs of gold and 2.5kgs of silver. A huge political storm has erupted around Australia after labor leader Kevin Rudd was found to have gone to a **strip** club during a taxpayer funded trip. | Equivalence | Independent |
| 2 | competition | race | Three countries withdrew from the **competition:** Germany, Spain and Switzerland. Morgan Tsvangirai, the leader of the Movement for Democratic Change (MDC), Zimbabwe’s main opposition party, has said that he will pull out of the **race** to become the president of Zimbabwe. | Reverse Entailment | Equivalence |
| 3 | boy | family | The birth of the **boy**, whose birth name is disputed among different sources, is considered very important in the entertainment world. Bill will likely disrupt the Obama **family**’s vacation to Martha’s Vineyard. | Forward Entailment | Other-related |
| 4 | jump | walk | Amid wild scenes of joy on the pitch he **jumped** onto the podium and lifted the trophy, the fourth of Italy’s history. In a game about rescuing hostages a hero might **walk** past Coca-Cola machine’s one week and Pepsi the next. | Other-related | Alternation |

Table 1: Illustration of annotation shifts when context is given. [1] the sense of **strip** in the given context is different from the one which is equivalent to **piece**. [2] the term **race** is judged out-of-context as more specific than **competition**, but is considered equivalent to it in a particular context. [3] a meronymy relation is (often) considered out-of-context as entailment, while in a given context this judgment doesn’t hold. [4] general relations may become more concrete when the context is given.

Creating a novel dataset for fine-grained lexical inference in-context. For each term-pair, we add a pair of context sentences, and re-annotate these term-pairs with respect to their contexts. We show that almost half of the semantically-related term-pairs become unrelated when the context is specified. Furthermore, a generic out-of-context relation may change within a given context (see table 1). We further report baseline results that demonstrate the utility of our dataset in developing fine-grained context-sensitive lexical inference methods.

## 2 Background

### 2.1 Lexical Inference Datasets

Figure 1 lists prominent human-annotated datasets used for developing lexical inference methods. In these datasets, each entry consists of an \((x, y)\) term-pair, annotated to whether a certain semantic relation holds between \(x\) and \(y\). Each dataset either specifies fine-grained semantic relations (see 2.2), or groups several semantic relations under a single coarse-grained relation (e.g. lexical substitution, similarity).

In some datasets, term-pairs are annotated to whether the relation holds between them in some unspecified contexts (out-of-context), while in others, the annotation is given with respect to a given context (in-context). In these datasets, each entry consists of a term-pair, \(x\) and \(y\), and context, where some of the datasets provide a single context in which \(x\) occurs while others provide a separate context for each of \(x\) and \(y\) (corresponding to the 1 context and 2 contexts columns in Figure 1). The latter simulates a frequent need in NLP applications, for example, a question answering system recognizes that **broadcast** entails **air** given the context of the question (“When was ‘Friends’ first aired?”) and that of the candidate passage (“‘Friends’ was first broadcast in 1994”).

We observe that most lexical inference datasets provide out-of-context annotations. The existing in-context datasets are annotated for coarse-grained semantic relations, such as similarity or relatedness, which may not be sufficiently informative.
In this paper, we focus on human-annotated datasets, and therefore find the above mentioned subset of human-annotated paraphrases particularly relevant; we refer to this dataset as PPDB-fine-human. This dataset, as well as the PPDB 2.0 automatically created resource, are still missing a key feature in lexical inference, since the semantic relation for each paraphrase pair is specified out of context.

### 3 Dataset Construction Methodology

In this section, we present a methodology of adding context to lexical inference datasets, that we apply on PPDB-fine-human.

#### 3.1 Selecting Phrase-Pairs

PPDB-fine-human is a quite large dataset (14k pairs), albeit with some phrase-pairs that are less useful for our purpose. We therefore applied the following filtering and editing on the phrase pairs:

**Relation Types**

We expected that phrase pairs that were annotated out-of-context as independent will remain independent in almost every context; indeed, out of a sample of 100 such pairs that we annotated within context, only 8% were annotated with another semantic relation. As this was too sparse to justify the cost of human annotations, we chose to omit such phrase pairs.

**Grammaticality-based Filtering**

Many phrases in PPDB-fine-human are ungrammatical, e.g. boy is. We consider such phrases less useful for our purpose, as semantic applications
usually apply lexical inferences on syntactically coherent constituents. We therefore parse the original SICK (Marelli et al., 2014) sentences containing these phrases, and omit pairs in which one of the phrases is not a constituent.

**Filtering Trivial Pairs** In order to avoid trivial paraphrase pairs, we filter out inflections (Iraq, Iraqi) and alternate spellings (center, centre), by omitting pairs that share the same lemma, or those that have Levenshtein distance $\leq 3$. In addition, we omit pairs that have lexical overlaps (a young lady, lady) and filter out pairs in which one of the two phrases is just a stop word.

**Removing Determiners** The annotation seems to be indifferent to the presence of a determiner, e.g., the labelers annotated all of (kid, the boy), (the boy, the kid), and (a kid, the boy) as reverse entailment. To avoid repetitive pairs, and to get a single “normalized” phrase, we remove preceding determiners, e.g., yielding (kid, boy).

Finally, it is interesting to note that PPDB-fine-human includes term-pairs in which terms are of different grammatical categories. Our view is that such cross-category term-pairs are often relevant for semantic inference (e.g. (bicycle, riding)) and therefore we decided to stick to the PPDB setting, and kept such pairs.

At the end of this filtering process we remained with 1385 phrase pairs from which we sampled 375 phrase pairs for our dataset, preserving the relative frequency across relation types in PPDB.

### 3.2 Adding Context Sentences

We used Wikinews\(^2\) to extract context sentences. We used the Wikinews dump from November 2015, converted the Wiki Markup to clean text using WikiExtractor\(^3\), and parsed the corpus using spaCy.\(^4\)

For each $(x, y)$ phrase-pair, we randomly sampled 10 sentence-pairs of the form $(s_x, s_y)$, such that $s_x$ contains $x$ and $s_y$ contains $y$. In the sampling process we require, for each of the two terms, that its 10 sentences are taken from different Wikinews articles, to obtain a broader range of the term’s senses. This yields 10 tuples of the form $(x, y, s_x, s_y)$ for each phrase pair and 3750 tuples in total.\(^5\)

We split the dataset to 70% train, 25% test, and 5% validation sets. Each of the sets contains different term-pairs, to avoid overfitting for the most common relation of a term-pair in the training set.

### 3.3 Annotation Task

Our annotation task, carried out on Amazon Mechanical Turk, followed that of Pavlick et al. (2015). We used their guidelines, and altered them only to consider the contexts. We instructed annotators to select the relation that holds between the terms $(x$ and $y$) while interpreting each term’s meaning *within* its given context $(s_x$ and $s_y$). To ensure the quality of workers, we applied a qualification test and required a US location, and a 99% approval rate for at least 1,000 prior HITS. We assigned each annotation to 5 workers, and, following Pavlick et al. (2015), selected the gold label using the majority rule, breaking ties at random. We note that for 91% of the examples, at least 3 of the annotators agreed.\(^6\)

The annotations yielded moderate levels of agreement, with Fleiss’ Kappa $\kappa = 0.51$ (Landis and Koch, 1977). For a fair comparison, we replicated the original out-of-context annotation on a sample of 100 pairs from our dataset, yielding agreement of $\kappa = 0.46$, while the in-context agreement for these pairs was $\kappa = 0.51$. As expected, adding context improves the agreement, by directing workers toward the same term senses while revealing rare senses that some workers may miss without context.\(^7\)

### 4 Analysis

Figure 2 displays the confusion matrix of relation annotations in context compared to the out-of-context annotations. Most prominently, while the original relation holds in many of the contexts, it is also common for term-pairs to become independent. In some cases, the semantic relation is changed (as in table 1).

---
\(^2\)https://en.wikinews.org/
\(^3\)https://github.com/attardi/wikiextractor
\(^4\)http://spacy.io/
\(^5\)Our dataset is comparable in size to most of the datasets in Figure 1. In particular, the SCWS dataset (Huang et al., 2012), which is the most similar to ours, contains 2003 term-pairs with context sentences.
\(^6\)We also released an additional version of the dataset, including only the agreeable 91%.
\(^7\)The gap between the reported agreement in Pavlick et al. (2015) ($\kappa = 0.56$) and our agreement for out-of-context annotations ($\kappa = 0.46$) may be explained by our filtering process, removing obvious and hence easily consensual pairs.
Figure 2: percentages of each relation annotation in-context, for annotations out-of-context. The diagonal shows out-of-context relations that hold in-context, and the last column shows term-pairs that become independent, usually due to sense-shifts. In all other cells, semantic relations are changed. Recall that we didn’t annotate out-of-context independent pairs.

4.1 Baseline Results

To demonstrate our dataset’s utility, we report several baseline performances on our test set (table 3). The first two are context-insensitive, assigning the same label to a term-pair in all its contexts; the first assigns manual labels from PPDB-fine-human, and the second assigns PPDB 2.0 classifier predictions. We also trained a context-sensitive logistic regression classifier on our train set, using the available PPDB 2.0 features, plus additional context-sensitive features. To represent words as vectors, we used pretrained GloVe embeddings of 300 dimensions, trained on Wikipedia (Pennington et al., 2014), and added the following features:

\[
\text{max}_{w \in s_y} \vec{w} \cdot \vec{w} \\
\text{max}_{w \in s_y} \vec{y} \cdot \vec{w} \\
\text{max}_{w_x \in s_y, w_y \in s_y} \vec{w}_x \cdot \vec{w}_y
\]

(1) and (2) measure similarities between a term and its most similar term in the other term’s context, and (3) measures the maximal word similarity across the contexts.

This context-sensitive method, trained on our dataset, notably outperforms context-insensitive baselines, thus illustrating the potential utility of our dataset for developing fine-grained context-sensitive lexical inference methods. Yet, the absolute performance is still mediocre, emphasizing the need to develop better such methods, using our dataset or similar ones created by our methodology.

5 Conclusion

In this paper, we presented a methodology for adding context to context-insensitive lexical inference datasets, and demonstrated it by creating such dataset over PPDB 2.0 fine-grained paraphrase-pair annotations. We then demonstrated that our dataset can indeed be used for developing fine-grained context-sensitive lexical inference methods, which outperform the corresponding context-insensitive baselines.

Acknowledgments

We would like to thank Ellie Pavlick and Chris Callison-Burch for their assistance and insightful comments.

This work was partially supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. JAIR, 49:1–47.


Chapter 10

Conclusion

In this chapter I summarize the contributions described in this thesis, and present open issues and the way forward to addressing them. The first contribution is going beyond relying on distributional representations and incorporating additional information sources when addressing various tasks pertaining to lexical inference (Section 10.1). The second contribution is an analysis of the effect of context on the applicability of lexical inferences (Section 10.2). The third contribution is two approaches for uncovering the implicit relationship between the constituent of a noun compound, which are part of a broader problem of learning meaningful representations for multi-word expressions (Section 10.3). Finally, I briefly discuss the imperative next step in lexical inferences - applying these inferences within downstream applications (Section 10.4).

10.1 Going Beyond the Distributional Hypothesis

Many NLP applications today rely on word embeddings to inform them about the semantic relations between words. However, word embeddings capture a fuzzy distributional similarity between words, which conflates multiple semantic relations. In many cases it would be beneficial to know the specific semantic relation that holds for a pair of words, e.g. identifying that “it is cold today” and “it is warm today” contradict each other.
Supervised models that use word embeddings to predict the specific semantic relations between words suffer from the “lexical memorization” effect (Levy et al., 2015a): they provide information on each of the words in isolation, by memorizing their distribution in the training data (e.g. predicting with certainty that entity is a hypernym of any other word). It is therefore clear that additional information sources are required for providing more accurate information about the relationship between a pair of words.

In this work I demonstrated that additional information sources are beneficial for various lexical inference tasks. In chapters 4 and 5 I used the lexico-syntactic patterns connecting the joint occurrences of the words in the corpus to identify the specific semantic relation that holds between them. I showed that this information is complementary to information originating in their distributional representations. The model consistently benefited from this information, especially in challenging evaluation settings, for rare words or senses, and for non-typical relations. In chapter 7 I applied a similar model to recognize the implicit semantic relation between the constituents of a noun compound. In all cases the models succeeded better than purely distributional models, especially in settings in which lexical memorization was disabled.

Finally, in chapter 6 I extracted lexically-divergent paraphrases. Many of these paraphrase pairs were not generally similar to each other, and in particular are not expected to have similar word embeddings, but given a certain assignment to their arguments and in specific contexts they had roughly the same meaning. What made this extraction possible was the event coreference assumption: both texts referred to the same entities within descriptions of the same news event, suggesting that they have roughly the same meaning, even despite having possibly very different distributional representations.

10.2 Modeling Context-sensitivity

Applying lexical inferences within applications imposes additional difficulty in detecting whether the potential lexical inference is valid within a given context. Such an application needs to address polysemy, at the basic level: words may
have different meanings in different contexts. In chapter 9 I showed that the context may cause nuanced changes in a word’s meaning, without choosing a completely different sense. This may change the semantic relation that holds between terms in different contexts. For instance, while a race is considered more specific than a competition, in some contexts “race” is used as a metaphor of “competition”, and they are considered equivalent in meaning.

Moreover, the validity of the inference depends on the monotonicity of the sentence. In an upward monotone sentence, a word entails its hypernym, as in “I like cats” → “I like animals”, while in downward monotone sentences it is the other way around: “I do not own animals” → “I do not own cats”. In previous-generation textual inference approaches, this phenomenon was handled with “natural logic” (MacCartney and Manning, 2007), a system of logical inference which operates over natural language.

Modern neural models represent a word in a given context by encoding the context, typically by using recurrent neural networks. Recently, it has been made possible even for applications with a scarce amount of training data. Instead of learning context representations from scratch with the downstream training objective, they rely on pre-trained contextualized word embeddings which may be plugged in into any neural model (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018), either as a fixed representation layer or fine-tuned to the task. Such models compute a dynamic vector representation of a word given its context. Doing so, they largely address polysemy, as they no longer conflate all the different senses of a word into a single vector.

Contextualized word representations have shown to significantly improve the performance upon using regular word embeddings, across various NLP applications. Recent work investigated what these representations capture with respect to the syntax and semantics of a sentence, by testing them on a broad range of tasks. Tenney et al. (2019) found that all the models produced strong representations for syntactic phenomena, but gained smaller performance improvements upon the baselines in the more semantic tasks. Liu et al. (2019) found that some tasks (e.g., identifying the tokens that comprise the conjuncts in a coordination construction) required fine-grained linguistic knowledge which
The contribution of this thesis is orthogonal to that of contextualized word embeddings with respect to improving upon standard word embeddings. Most of the work presented in the thesis largely addresses the out-of-context lexical inference setting, as a step towards learning and applying such inferences within context. The focus was mainly on going beyond learning fuzzy distributional similarity between words or phrases, and modelling the specific semantic relations between them. Contextualized word embeddings, despite their other advantages, suffer from similar issues as standard word embeddings with respect to their distributional nature. While standard word embeddings often confuse synonyms with antonyms or with similar but mutually exclusive terms, contextualized word embeddings, now being used as source of knowledge, may produce wrongly-generalized facts like “Barack Obama’s wife is Hillary” (Logan et al., 2019), and even a strong structural indication of negation does not stop them from completing “facts” like “birds cannot” with “fly” (Kassner and Schütze, 2019).

10.3 Representations Beyond the Word-Level

In recent years, with the help of recurrent neural networks which facilitate processing texts in arbitrary lengths, phrase, paragraph, and sentence embeddings became popular (e.g. Socher et al., 2011; Kiros et al., 2015). Conneau et al. (2018) showed that sentence embeddings capture various syntactic and semantic properties of a sentence. However, this approach works under the assumption of full compositionality: the sentence’s meaning is derived fully from the sum of its word meanings.

Multiword expressions (MWE) have long been a “pain in the neck” for NLP (Sag et al., 2002), specifically because they often hold meanings which are not derived fully from their constituent words. Idioms (“look what the cat dragged in”), fixed expressions (“ad hoc”), and proper names (“New York”) easily break the compositionality assumption. In adjective-noun composition, the adjective usually adds information to the noun (e.g. red car), but often it is already implied
(e.g. little baby, because babies are always little), or completely contradicts the meaning of the noun (e.g. fake gun) (Pavlick and Callison-Burch, 2016). Finally, noun compounds are often non-compositional (monkey business) or only related to one of the constituents (spelling bee is related to spelling). Even when they are fully-compositional, there is no “one-size fits all” composition function. As I showed in chapters 7 and 8, various different semantic relations may hold between the constituents of a noun compound (e.g. “olive oil” is oil made of olives, while “baby oil” is oil made for babies).

As part of the broader goal of addressing this representation gap, I focused on noun compounds and have developed two approaches for uncovering the semantic relation that holds between the constituents of a noun compound. In chapter 7 I developed a method for the classification task, which maps a noun compound to one of a set of pre-defined relationships, while in chapter 8 I worked on the paraphrasing task, which extracts a list of prepositional and verbal paraphrases describing each noun compound. The common to both models was using the joint occurrences of the noun compound’s constituents in the corpus as means of explaining the semantic relation, as in “cake made of apples”, and encoding the patterns connecting the joint occurrences with neural networks, which help generalize semantically similar patterns.

Given the prevalence of MWEs in English, the current representation gap likely hurts the performance of downstream applications. In Shwartz and Dagan (2019), I have tested the ability of the new contextualized word representations to handle phenomena related to lexical composition, and found that they were impressively good at recognizing meaning shift. For example, that spelling bee is unrelated to bees or that carry on has a different meaning from carry. This is not a surprising result given that the context-sensitive nature of such representation makes them extremely good in word sense disambiguation. With that said, an attempt to query such representations for lexical substitutes suggests that the representation of the shifted senses is of lower quality, likely due to their infrequency in the training corpus.

Moreover, none of the representations tested was particularly successful in recovering implicit meaning, e.g. recognizing the relationships implicitly con-
veyed in olive oil vs. in baby oil. There is no reason to expect such representation to be better than standard word embeddings in recovering something that wasn’t said. I expect that future research in representation learning will attempt to address “reading between the lines” and incorporating knowledge which is conveyed implicitly.

10.4 Going Forward: Improving Downstream Applications

Recognizing semantic relations between lexical items is crucial for NLP applications and infrastructure tasks, in order to overcome lexical variability. A representative example is recognizing textual entailment (RTE) (Dagan et al., 2013). In this task, two sentences, a premise (p) and a hypothesis (h) are given, and the goal is to recognize whether the p entails, contradicts, or is neutral with h. For instance, consider the following entailment example: p: “Donald Trump spoke in Huntsville, criticizing the NFL players who protested during the anthem.”, and h: “In his Alabama speech, the president attacked the NFL players who protested during the hymn.”.

Previous generation models for this task leveraged an abundance of lexical resources to identify lexical inferences (Androutsopoulos and Malakasiotis, 2010), e.g. identifying that criticize and attack refer to the same action and that Huntsville is in Alabama. In recent years, with the availability of large-scale datasets, the task shifted from explicitly recognizing lexical inferences to training end-to-end neural models that rely solely on word embeddings for modeling lexical inferences. While these neural models achieve impressive performance on the new datasets, various recent works showed that this is achieved thanks to artifacts in the data rather than solving the actual task (Gururangan et al., 2018; Poliak et al., 2018). Specifically, Glockner et al. (2018), Naik et al. (2018), and Sanchez et al. (2018) all showed that the models are limited in their ability to recognize lexical inferences.

It is imperative, therefore, to incorporate external lexical knowledge into
these models, however, it is not trivial to do so. One promising approach is to incorporate this knowledge into the word embeddings themselves, and then use the embeddings in downstream tasks as one would use standard distributional embeddings. There is a growing research interest in embedding semantic relations from WordNet (Fellbaum, 1998) into vectors (e.g. Faruqui et al., 2015; Vendrov et al., 2016; Nguyen et al., 2017; Nickel and Kiela, 2017; Vulic and Mrksic, 2018), but these vectors have only been evaluated intrinsically on their ability to complete missing edges from the WordNet taxonomy. Using them in downstream applications is not trivial since they encode the information differently from distributional embeddings.

A promising research direction is to learn word pair embeddings that capture the semantic relation between the words, and use them instead of or in addition to the embeddings of each single word. Joshi et al. (2019) learned such vectors in an unsupervised manner from the lexico-syntactic paths connecting two words in the corpus. I expect that in the future, such knowledge will be incorporated into general word representations by adding auxiliary objectives to their training.
Bibliography


Max Glockner, Vered Shwartz, and Yoav Goldberg. Breaking nli systems with sentences that require simple lexical inferences. In The 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia, July 2018.


Vered Shwartz and Ido Dagan. Adding context to semantic data-driven paraphrasing. In *SEM, Berlin, Germany, August 2016a*.

Vered Shwartz and Ido Dagan. Path-based vs. distributional information in recognizing lexical semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex-V), in COLING, Osaka, Japan, December 2016b*.


Vered Shwartz, Omer Levy, Ido Dagan, and Jacob Goldberger. Learning to exploit structured resources for lexical inference. In *CoNLL, Beijing, China, July 2015*.


Tim Van de Cruys, Stergos Afantenos, and Philippe Muller. Melodi: A supervised distributional approach for free paraphrasing of noun compounds. In


תקציר

יישומי רבים בתחום הבנת שפה טבעית נאלצים להתמודד עם שיקשים עיקריים בشهدות האנושיות. אחד ההשיקשים הוא גיוון לקסיקלי: ניתן לבטא את אותה ה뜻ות בדרכים שונות. השיקשון הרוב־حصرיות הוא למילה иметь מספר מושגים שונים בהתאם להקשר בו היא נאמרה. המשמעויות של האלמנטים המתחום של הס🦚ת מתחום. שיקשים סמנטיים רבים הם יולקים של שישולש חטיף.

hisks: מילים דרומתיות, יסרי הכללה (חותול/בעל חיים), "חתול מ” (צלדדו/אנגלית), וכד.’

היהוו היסק לקסיקליים אשר תחרויות לפיצול של שימיום רב בענוד שפה בטבע מתא aupot קשטיים, לEMPLATEים יסרים שימוש בلزمאות ובשותה של שיקשים, בחוף הקיסרי והשיג יסרוים כ 노력 המילים, היא נאמרה לימים של המשמעותים של מילים וחקלאיים במילים והם ידועים המדוע בין שניים, והם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים המדוע בין שניים. בחרובים יישומים אחר ומילים Humanities שניהם ידועים-med2022/09/0061-0.jpg
The model can only be in trouble coding the problems in previous work. First, previous work demonstrated models based on context could infer hypotheses about distributional properties to generate information on each word separately, but without any connection between them. For example, they could answer whether the word "life" belongs to the category of "living creature" or not, while the word "life" is a category of the word "living creature" and "cat". As the result, the model demonstrated a clear connection between the words, as well as the context in the text. Second, the model represents a new inventory of the methods based on the distributional properties of the common words, which are more effective between similar sentences. The method achieved the best results in the task, which is also the case in another study, where the results were compared between the methods based on parallel sentences, such as questions and answers from the same domain, where the model of the previous work achieved the best results, due to its ability to generalize.

The model also created a new method for generating random sentences between a pair of sentences, such as questions and answers, where the model generated a new method for generating random sentences based on the distributional properties of the common words, which are more effective between similar sentences. The method achieved the best results in the task, which is also the case in another study, where the results were compared between the methods based on parallel sentences, such as questions and answers from the same domain, where the model of the previous work achieved the best results, due to its ability to generalize.

The model also created a new method for generating random sentences between a pair of sentences, such as questions and answers, where the model generated a new method for generating random sentences based on the distributional properties of the common words, which are more effective between similar sentences. The method achieved the best results in the task, which is also the case in another study, where the results were compared between the methods based on parallel sentences, such as questions and answers from the same domain, where the model of the previous work achieved the best results, due to its ability to generalize.

The model also created a new method for generating random sentences between a pair of sentences, such as questions and answers, where the model generated a new method for generating random sentences based on the distributional properties of the common words, which are more effective between similar sentences. The method achieved the best results in the task, which is also the case in another study, where the results were compared between the methods based on parallel sentences, such as questions and answers from the same domain, where the model of the previous work achieved the best results, due to its ability to generalize.
ตอนי עניינים

tכזר

1 הקדמה

1 סיווג ליחסיים סמנטיים

1.1 הפרידקטים

1.2 פארארפרזות של פרידקטים

1.3 פורוש של צירופים שמניים

1.4 היסקים לקסיקליים תלויים קשר

1.5 תרומות למגננה העונבה

2 רקע

2 סמנטיקה התפלגתית

2.1 טקסונומיות ידועות מובנה

2.2 מנתונות של היסקים לקסיקליים

3 מדידים לא מצבים שליחי היסקים

4 יוזרו ייחסיים יבשות משולבות מבוסсты מסלולים והתחנות

5 יוזרו מבוסсты מסלולים לעומת יוזרו התפלגותית בשליחי היסקים

6 רכישת פארארפרזות של פרידקטים מפורשים חדשות בתוכי

7 סווג צירופים שמניים ליחסיים סמנטיים

8 פורוש צירופים שמניים באמחאות פארארפרזות
על החשיבות של הקושר בדיקות לקסיקליים

9

סיום

10

סמותו קמעה דרך התוכן והtypically

10.1

פער של הקושר לקסיקליים ת��יידקוש

10.2

מידל של הקושר לקסיקליים ת��יידקוש

10.3

ייצוג ביטויים מורכבים מתאורה מילים

10.4

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצوعי של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה

92

רשימה מקורות

90

הדרך ההלא: שיפור ביצועים של משימות קיצה
עבודה והנחיות של פרופ’ עידו דר מוסטקה למדעי המחשב, אוניברסיטת בר אילן.
למידע החדש לקסיקליים בדינום גובה

חובר לסך קבליות על תואר "דוקטור לפילוסופיה"

נכתב:

ורד שוורץ

המחלקה למידע המחשב
הפקולטה למידעים מדוייקים

הנושאים של אוניברסיטת בר-אילן

ניסן תשע"ט

רموت נז