Natural Language Engineering **21** (3): 437–476. © Her Majesty the Queen in Right of Canada, as represented by the National Research Council Canada 2014

doi:10.1017/S1351324913000387

Experiments with three approaches to recognizing lexical entailment

P. D. TURNEY and S. M. MOHAMMAD

National Research Council Canada, Ottawa, Ontario K1A 0R6, Canada
e-mail: {peter.turney, saif.mohammad}@nrc-cnrc.gc.ca

(*Received 27 November 2012; revised 2 December 2013; accepted 4 December 2013; first published online 28 January 2014*)

Abstract

Inference in natural language often involves recognizing lexical entailment (RLE), that is, identifying whether one word entails another. For example, *buy* entails *own*. Two general strategies for RLE have been proposed: One strategy is to manually construct an asymmetric similarity measure for context vectors (*directional similarity*) and another is to treat RLE as a problem of learning to recognize semantic relations using supervised machine-learning techniques (*relation classification*). In this paper, we experiment with two recent state-of-the-art representatives of the two general strategies. The first approach is an asymmetric similarity measure (an instance of the *directional similarity* strategy), designed to capture the degree to which the contexts of a word, *a*, form a subset of the contexts of another word, *b*. The second approach (an instance of the *relation classification* strategy) represents a word pair, *a* : *b*, with a feature vector that is the concatenation of the context vectors of *a* and *b*, and then applies supervised learning to a training set of labeled feature vectors. In addition, we introduce a third approach that is a new instance of the *relation classification* strategy. The third approach represents a word pair, *a* : *b*, with a feature vector in which the features are the differences in the similarities of *a* and *b* to a set of reference words. All three approaches use vector space models of semantics, based on word–context matrices. We perform an extensive evaluation of the three approaches using three different datasets. The proposed new approach (similarity differences) performs significantly better than the other two approaches on some datasets and there is no dataset for which it is significantly worse. Along the way, we address some of the concerns raised in past research, regarding the treatment of RLE as a problem of semantic relation classification, and we suggest, it is beneficial to make connections between the research in lexical entailment and the research in semantic relation classification.

1 Introduction

Recognizing textual entailment (RTE) is a popular task in natural language processing research becuase of its relevance for text summarization, information retrieval, information extraction, question answering, machine translation, paraphrasing, and other applications (Androutsopoulos and Malakasiotis 2010). RTE involves pairs of sentences, such as the following (Dagan *et al.* 2009):

Text: *Norway's most famous painting, 'The Scream' by Edvard Munch, was recovered Saturday, almost three months after it was stolen from an*

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Oslo museum.
Hypothesis: *Edvard Munch painted 'The Scream'.*

The objective is to develop algorithms that can determine whether the text sentence entails the hypothesis sentence. In RTE, the gold standard for entailment is established by common sense, rather than formal logic. The text entails the hypothesis if the meaning of the hypothesis can be inferred from the meaning of the text, according to typical human interpretations of the text and the hypothesis (Dagan *et al.* 2009). In the above example, the text entails the hypothesis. In recent years, the RTE pairs have grown richer and more challenging. The text may now be a whole paragraph.

In many cases, to recognize when one *sentence* entails another, we must first be able to recognize when one *word* entails another (Geffet and Dagan 2005). Consider this sentence pair (our example):

Text: *George was bitten by a dog.*
Hypothesis: *George was attacked by an animal.*

To recognize that the text entails the hypothesis, we must first recognize that *bitten* entails *attacked* and *dog* entails *animal*. This is the problem of recognizing lexical entailment (RLE). (We discuss the definition of lexical entailment in Section 2.)

Vector space models (VSMs) of semantics have been particularly useful for lexical semantics (Turney and Pantel 2010), hence it is natural to apply them to RLE. In this paper, we experiment with three vector space model algorithms for lexical entailment. All three use word–context matrices, in which a word corresponds to a row vector, called a *context vector*. For a given word, the corresponding context vector represents the distribution of the word over various contexts. The contexts consist of the words that occur near the given word in a large corpus of text. These models are inspired by the distributional hypothesis (Harris 1954; Firth 1957):

Distributional hypothesis: Words that occur in similar contexts tend to have similar meanings.

The first of the three algorithms, *balAPinc* (balanced average precision (AP) for distributional inclusion), attempts to address the problem of RLE with an asymmetric similarity measure for context vectors (Kotlerman *et al.* 2010). The idea is to design a measure that captures the context inclusion hypothesis (Geffet and Dagan 2005):

Context inclusion hypothesis: If a word *a* tends to occur in a subset of the contexts in which a word *b* occurs (*b* contextually includes *a*), then *a* (the narrower term) tends to entail *b* (the broader term).

This is our paraphrase of what Geffet and Dagan (2005) call the *distributional inclusion hypothesis*. We prefer to call it *context inclusion* rather than *distributional inclusion*, to be clear about what is included. In the text–hypothesis example above, the narrower terms are *bitten* and *dog* and the broader terms are *attacked* and *animal*.

The intent of balAPinc is to take the context vectors **a** and **b** for the words *a* and *b* and calculate a numerical score that measures the degree to which *b* contextually

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

includes *a*. The context inclusion hypothesis is inspired by model theory in formal logic (Hunter 1996). Let *a* and *b* be assertions in formal logic. In model theory, '$a \models b$' means *a* entails *b*. If $a \models b$, then the set of models in which *a* is true is a subset of the set of models in which *b* is true. That is, the models of *b* include the models of *a*.

The second and third algorithms approach the task of RLE by using techniques from research in semantic relation classification. Semantic relation classification is the task of learning to recognize when a word pair is an instance of a given semantic relation class (Rosario and Hearst 2001; Rosario, Hearst, and Fillmore 2002; Nastase and Szpakowicz 2003; Turney 2006; Girju *et al.* 2007).

An important subclass of lexical entailment is covered by the hyponymy–hypernymy semantic relation. If a word pair $a:b$ is an instance of the hyponym–hypernym relation (*dog:animal*), then $a \models b$. There is a relatively large body of work on semantic relation classification in general, with good results on the hyponym–hypernym relation in particular (Hearst 1992; Snow, Jurafsky and Ng 2006). Since semantic relation classification algorithms have worked well for this important subclass of lexical entailment, it seems plausible that this approach can be expanded to cover other subclasses of lexical entailment, and perhaps all subclasses of lexical entailment. (We say more about this in Section 3.)

The second of the three algorithms represents a word pair, $a:b$, with a feature vector that is the concatenation of the context vector **a** for *a* and the context vector **b** for *b* (Baroni *et al.* 2012). For example, the concatenation of the two three-dimensional vectors $\langle 1, 2, 3\rangle$ and $\langle 4, 5, 6\rangle$ is the six-dimensional vector $\langle 1, 2, 3, 4, 5, 6\rangle$. This algorithm was not given a name by Baroni *et al.* (2012). For ease of reference, we will call it concatenated vectors (*ConVecs*).

ConVecs is based on the context combination hypothesis (Baroni *et al.* 2012).

> *Context combination hypothesis*: The tendency of *a* to entail *b* is correlated with some learnable function of the contexts in which *a* occurs and the contexts in which *b* occurs; some conjunctions of contexts tend to indicate entailment and others tend to indicate a lack of entailment.

This hypothesis implies that the contexts of *a* (the elements in the context vector **a**) and the contexts of *b* (elements in **b**) are suitable features for a feature vector representation of the word pair $a:b$. That is, if this hypothesis is correct, concatenated context vectors are an appropriate representation of word pairs for supervised machine learning of lexical entailment. This hypothesis was not explicitly stated by Baroni *et al.* (2012) but it is implicit in their approach.

In the semantic relation classification literature, vector concatentation (but not necessarily with context vectors) is a common way to construct feature vectors for supervised learning with word pairs (Rosario and Hearst 2001; Rosario *et al.* 2002; Nastase and Szpakowicz 2003). Context concatentation is a first-order feature vector representation of word pairs. We call it *first-order* because the features are directly based on the elements of the context vectors.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

This paper introduces a new algorithm, *SimDiffs* (similarity differences), as the third of the three algorithms we evaluate. SimDiffs uses a second-order feature vector representation of $a:b$, in which the features are differences in the similarities of a and b to a set of reference words, R. The similarities are given by cosines of the first-order context vectors for a, b, and the reference words, $r \in R$. (We use a set of common words for R, as described in Section 6.3. We do not experiment with other choices for R.)

SimDiffs is dependent on the similarity differences hypothesis (introduced here):

> *Similarity differences hypothesis*: The tendency of a to entail b is correlated with some learnable function of the differences in their similarities, $\text{sim}(a,r) - \text{sim}(b,r)$, to a set of reference words, $r \in R$; some differences tend to indicate entailment and others tend to indicate a lack of entailment.

For example, consider *dog* $\models$ *animal* versus *table* $\not\models$ *animal*. Suppose that *life* is one of the reference words. We see that *dog* and *animal* are similar with respect to the reference word *life*; the difference in their similarities is small. On the other hand, *table* and *animal* are dissimilar with respect to *life*; there is a large difference in their similarities. Some differences are important for entailment (such as whether something is animate or inanimate) and others usually have little effect (such as the color of a thing). Given labeled training data, we may be able to learn how differences in similarities affect lexical entailment.

We empirically evaluate the three algorithms, balAPinc, ConVecs, and SimDiffs, using three different datasets. We find that SimDiffs performs significantly better than the other two algorithms in some cases and there is no case for which it is significantly worse. ConVecs is significantly worse than balAPinc and SimDiffs on one dataset, whereas balAPinc is significantly worse than ConVecs on one dataset and significantly worse than SimDiffs on two datasets.

Section 2 defines *lexical entailment* in terms of semantic relations between words. There is some disagreement about whether lexical entailment should be approached as a semantic relation classification task. We address this issue in Section 3. Past work on RLE is examined in Section 4. Performance measures for RLE algorithms are presented in Section 5. We describe the three algorithms in detail in Section 6. The three algorithms are evaluated using three datasets, which are presented in Section 7. We use the datasets of Kotlerman *et al.* (2010), Baroni *et al.* (2012), and Jurgens *et al.* (2012). The experimental results are reported in Section 8. We discuss some implications of the experiments in Section 9. Limitations of this work are considered in Section 10, and we conclude this paper in Section 11.

2 Defining lexical entailment

Let w and v be two words. Zhitomirsky-Geffet and Dagan (2009: 442) define *substitutable lexical entailment* as follows:

> ... w entails v, if the following two conditions are fulfilled:

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

(1) *Word meaning entailment*: The meaning of a possible sense of w implies a possible sense of v.
(2) *Substitutability*: w can substitute for v in some naturally occurring sentence such that the meaning of the modified sentence would entail the meaning of the original one.

We call this the *substitutional definition* of lexical entailment.

We present a different definition of lexical entailment here. The idea that whether one word entails another depends on the semantic relation between the words. We discuss some objections to this idea in Section 3.

Let x and y be two words. To be able to say that x entails y outside the context of a specific sentence, it must be the case that there is a strong semantic relation between x and y, and the entailment must follow from the nature of that semantic relation. We say that x entails y if the following three conditions are fulfilled:

(1) *Typical relation*: Given x and y, there should be a typical semantic relation $R(x, y)$ that comes to mind. Let $R(x, y)$ be the typical semantic relation between x and y. If there is no typical semantic relation between x and y, then x cannot entail y outside a specific context.
(2) *Semantic relation entailment*: If x and y typically have the semantic relation $R(x, y)$, then it should follow from the meaning of the semantic relation that x implies y.
(3) *Relation entailment agreement*: If x and y have two or more typical semantic relations and the relations do not agree on whether x implies y, then assume that x does not imply y.

We call this the *relational definition* of lexical entailment.

In the first condition of the relational definition, the *typical relation* between x and y is the relation that naturally comes to mind when x and y are presented together. If x and y have multiple senses, the juxtaposition of x and y may suggest a semantic relation and it may also constrain the possible senses of the words. The constrained senses of the words are not necessarily the most frequent or prototypical senses of the words.

For example, consider the words *lion* and *cat*. The word *cat* has the senses *house cat* (a specific type of cat) and *feline* (the general class of cats, including domestic cats and wild cats). When the words *lion* and *cat* are juxtaposed, the relation that naturally comes to mind (for us) is hyponym–hypernym (a *lion* is a type of *cat*) and the sense of *cat* is constrained to *feline*, although the *house cat* sense is more frequent and prototypical than the *feline* sense.

Context determines the sense of an ambiguous word, but lexical entailment considers word pairs outside the context of sentences. Since word senses can affect entailment, any approach to lexical entailment must decide how to handle ambiguous words. The substitutional definition of lexical entailment invites us to imagine a natural sentence that provides the missing context and constrains the possible senses of the two words. The relational definition of lexical entailment invites us to imagine a semantic relation that connects the two words and constrains their possible senses.

The second condition of the relational definition determines whether one word entails another, based on their semantic relation. Since a hyponym implies its

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

hypernym, *lion* entails *cat*. The second condition excludes semantic relations that do not imply entailment. For example, antonymy is excluded (e.g. *tall* does not imply *short*) and the hyponym–hypernym relation is only included when the direction is correct (e.g. *lion* implies *cat* but *cat* does not imply *lion*).

The first condition in the substitutional definition of lexical entailment (word meaning entailment) asks us to consider whether the sense of one word implies the sense of another word. We hypothesize that any such implication must depend on the semantic relation between the senses of the words. It seems to us that if there is no semantic relation between the words then it is not possible for one word to imply the other. If one word implies another, the implication must follow from the nature of their semantic relation. The idea of the second condition in the relational definition of lexical entailment is to make this connection between semantic relations and lexical entailment explicit.

The third condition of the relational definition handles ambiguous cases by erring on the side of non-entailment. Some people might feel that *lion* and *cat* suggest either the hyponym–hypernym relation (assuming *cat* means *feline*) or the coordinate relation (assuming that *cat* means *house cat*). Coordinates are words with a shared hypernym. *Lion* and *house cat* share the hypernym *feline*. If *cat* means *house cat*, then *lion* and *cat* are coordinates. A hyponym implies its hypernym, but coordinates do not imply each other. *Lion* implies *cat* in the *feline* sense but not in the *house cat* sense. Thus, these two relations (hyponym–hypernym and coordinate) do not agree on whether *lion* implies *cat*. In this case, we believe that the hyponym–hypernym is more natural, so we say that *lion* implies *cat*. For people who feel both semantic relations are natural, the third condition says that there is no entailment; for them *lion* does not imply *cat*.

The third condition could be modified for different uses. For our dataset (Section 7.3), we chose to err on the side of non-entailment, but ideally the choice would be made based on the downstream application. For some applications, it may be better to err on the side of entailment. One possibility is to give higher weight to some relations and use the weighting to choose between entailment and non-entailment when two or more relations disagree. The weighting could be based on the corpus frequency of the relations or the contexts in which the words appear.

To apply the relational definition of lexical entailment, it is helpful to have a taxonomy of semantic relations to provide options for *R*. In this paper, we use the taxonomy of Bejar, Chaffin and Embretson (1991), which includes seventy-nine subcategories of semantic relations, grouped into ten high-level categories. The taxonomy is given in Tables 2 and 3 in Section 7.3.

It might seem that the relational definition redefines *lexical entailment* in a way that makes our results incomparable with past results, but we believe that our results are indeed comparable with past work. Both the substitutional definition and the relational definition are *operational definitions*: They are tests used to determine the presence of entailment. They both require an understanding of the word *implies*, but *implies* is a synonym for *entails*; they are not *theoretical definitions* of entailment. They attempt to objectively capture the same underlying notion of implication, and hence they may be compared and contrasted in terms of how well they capture that notion.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Zhitomirsky-Geffet and Dagan's (2009) substitutional definition of lexical entailment was intended to capture only substitutional cases of entailment. They explicitly excluded non-substitutable lexical entailment. They argue that their two conditions yield good inter-annotator agreement and result in lexical entailment decisions that fit well with the needs of systems for RTE.

We believe that there is a tradeoff between inter-annotator agreement and coverage. The substitutional and relational definitions differ regarding this tradeoff. The substitutional definition leads to higher levels of inter-annotator agreement than the relational definition, but the substitutional definition excludes (by design) important cases of lexical entailment (see Section 7.3.2).

Consider the following example:

Text: *Jane accidentally broke a glass.*
Hypothesis: *Jane accidentally broke something fragile.*

For the word pair *glass:fragile*, the typical relation that comes to mind is item: attribute, 'an *x* has the attribute *y*' (ID 5a in the semantic relation taxonomy); thus the first condition of the relational definition is fulfilled. An item entails its attributes; *glass* entails *fragile*; thus the second condition is fulfilled. There are exceptions, such as bulletproof glass, but bulletproof glass is not typical glass. There is no other typical relation between *glass* and *fragile*, so the third condition is fulfilled.

One limitation of *substitutability* as defined by Zhitomirsky-Geffet and Dagan (2009) is that it does not allow lexical entailment from one part of speech to another. For example, *glass* entails *fragile*, but *glass* is a noun and *fragile* is an adjective, so we cannot substitute one for the other in a sentence. However, in spite of the difference in their parts of speech, it seems reasonable to say that *glass* entails *fragile*. In a typical situation that involves *glass*, the situation also involves something *fragile*.

As another example of a case where the substitutional definition excludes a lexical entailment that the relational definition captures, consider *bequeath:heir*, an instance of the act:recipient relation (ID 7e in the relation taxonomy):

Text: *George bequeathed his estate to Jane.*
Hypothesis: *Jane was the heir to George's estate.*

It is reasonable to say that the act of bequeathing entails that there is an heir, although the verb *bequeathed* cannot be substituted for the noun *heir*.

To address this limitation of the substitutional definition, one possibility would be to relax the definition of *substitutability* to cope with different parts of speech. For example, given a noun *x* and an adjective *y*, we could allow 'an *x*' (*a glass*) to be replaced with 'something *y*' (*something fragile*). Perhaps a relatively small list of *substitutional patterns* could handle most parts of speech substitution cases. However, we do not pursue this option here, because it does not address a fundamental limitation of the substitutional definition, which is the absence of semantic relations. We believe that semantic relations and lexical entailment are intimately connected (see Section 3).

The idea of substitutional patterns suggests the generalization of lexical entailment to phrasal entailment. For example, the phrase '*x* bequeathed *y* to *z*' entails the phrase

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

'*z* was the heir to *x*'s *y*'. Patterns like this have been learned from corpora (Lin and Pantel 2001) and applied successfully to RTE (Mirkin *et al.* 2009a). However, our focus here is lexical entailment, not phrasal entailment. We believe that a good algorithm for lexical entailment should be useful as a component in an algorithm for phrasal entailment.

In our experiments, we use three different datasets. All three consist of word pairs that have been labeled *entails* or *does not entail*. One dataset (Section 7.1) was labeled using Zhitomirsky-Geffet and Dagan's (2009) substitutional definition. On preliminary inspection, it seems that the semantic relations in this dataset are often part–whole and hyponym–hypernym relations, but the word pairs have not been systematically labeled with relation categories. In another dataset (Section 7.2), all of the pairs that are labeled *entails* are instances of the hyponym–hypernym relation. In the third dataset (Section 7.3), the pairs were generated from Bejar, Chaffin, and Embretson's (1991) taxonomy. This dataset includes pairs sampled from all seventy-nine of the subcategories in the taxonomy. Each pair was labeled *entails* or *does not entail* based on the subcategory it came from. Tables 2 and 3 in Section 7.3 list all of the subcategories of relations and their entailment labels.

Lexical entailment is sometimes asymmetric (e.g. for word pairs that are instances of the hyponym–hypernym relation) and sometimes symmetric (e.g. for synonyms) (Geffet and Dagan 2005; Kotlerman *et al.* 2010). Both the substitutional and relational definitions allow this blend of symmetry and asymmetry.

In the semantic relation classification literature (discussed in Section 4), supervised learning algorithms are applied to the task of classifying word pairs. In general, these algorithms are capable of classifying both symmetric and asymmetric relations. In particular, both ConVecs and SimDiffs approach lexical entailment as a problem of supervised relation classification, and both are capable of learning symmetric and asymmetric relations. They should be able to learn when lexical entailment behaves asymmetrically (e.g. with cases like *glass:fragile*) and when it behaves symmetrically (e.g. with cases like *car:automobile*).

The balAPinc measure is designed to capture asymmetry, but it is likely to give approximately equal scores to *car:automobile* and *automobile:car*. This can be seen by considering the details of its definition (see Section 6.1).

3 Semantic relations and lexical entailment

Some researchers have applied semantic relation classification to lexical entailment (Akhmatova and Dras 2009; Baroni *et al.* 2012), but Zhitomirsky-Geffet and Dagan (2009: 443) have argued against this as follows:

> ... lexical entailment is not just a superset of other known relations, but it is rather designed to select those sub-cases of other lexical relations that are needed for applied entailment inference. For example, lexical entailment does not cover all cases of meronyms (e.g. division does not entail company), but only some sub-cases of part–whole relationship mentioned herein. In addition, some other relations are also covered by lexical entailment, like ocean and water and murder and death, which do not seem to directly correspond to meronymy or hyponymy relations.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Note also that whereas lexical entailment is a directional relation that specifies which word of the pair entails the other, the relation may hold in both directions for a pair of words, as is the case for synonyms.

We agree with Zhitomirsky-Geffet and Dagan (2009) that some sub-cases of part–whole involve lexical entailment and other sub-cases do not. However, this issue can be addressed by breaking the part–whole category into subcategories.

One of the high-level categories in Bejar *et al.*'s (1991) taxonomy is part–whole (ID 2 in the taxonomy), which has ten subcategories. We claim that eight of the ten subcategories involve entailment and two do not, which is consistent with the claim that 'lexical entailment does not cover all cases of meronyms' (in the above quotation).

Regarding 'ocean and water and murder and death' (in the above quotation), the word pair *ocean:water* is an instance of Bejar *et al.*'s (1991) object:stuff subcategory (ID 2g in the taxonomy) and *murder:death* is an instance of the cause:effect subcategory (ID 8a in the taxonomy). Regarding relations for which there is lexical entailment in both the directions, synonymy (ID 3a in the taxonomy) is readily handled by marking it as entailing in both the directions (see Tables 2 and 3 in Section 7.3).

We believe that Zhitomirsky-Geffet and Dagan's (2009) argument is correct for high-level categories but incorrect for subcategories. We offer the following hypothesis (introduced here):

> *Semantic relation subcategories hypothesis*: Lexical entailment is *not* a superset of high-level categories of semantic relations, but it *is* a superset of lower level subcategories of semantic relations.

This hypothesis implies a tight connection between research in RLE and research in semantic relation classification.

ConVecs and SimDiffs treat RLE as a semantic relation classification problem. These algorithms do not require the semantic relation subcategories hypothesis: It is possible that it may be fruitful to use ideas from research in semantic relation classification even if the hypothesis is wrong. However, if the semantic relation subcategories hypothesis is correct, then there is even more reason to treat RLE as a semantic relation classification problem.

We use the semantic relation subcategories hypothesis in Section 7.3 as a new way of generating a dataset for evaluating RLE algorithms. In our experiments (Section 8), we train the algorithms using data based on Bejar *et al.*'s (1991) taxonomy and then test them on previous lexical entailment datasets.

We do not claim that Bejar *et al.*'s (1991) taxonomy handles all cases of lexical entailment, but our results suggest that it covers enough cases to be effective. Future work may discover lexical entailments that do not fit readily in Bejar *et al.*'s (1991) taxonomy, but we believe that the taxonomy can be expanded to handle exceptions as they are discovered.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

4 Related works

The first RTE Challenge took place in 2005 (Dagan, Glickman and Magnini 2006) and it has been a regular event since then.[1] Since the beginning, many RTE systems have included a module for RLE (Herrera, Penas and Verdejo 2006; Hickl *et al.* 2006). The early RLE modules typically used a symmetric similarity measure, such as the cosine measure (Salton and McGill 1983), the LIN measure (Lin 1998), or a measure based on WordNet (Pedersen, Patwardhan and Michelizzi 2004), but it was understood that entailment is inherently asymmetric and any symmetric measure can only be a rough approximation (Geffet and Dagan 2005).

Lee (1999) proposed an asymmetric similarity measure for the degree to which a word *a* can be replaced by a word *b* in a sentence, without substantially changing the meaning of the sentence. Weeds and Weir (2003) introduced an asymmetric similarity measure for the degree to which a specific term *a* is subsumed by a more general term *b* (see also Weeds, Weir and McCarthy 2004). This idea was developed further, specifically for application to lexical entailment, in a series of papers that culminated in the balAPinc measure of the degree to which *a* entails *b* (Geffet and Dagan 2005; Szpektor and Dagan 2008; Zhitomirsky-Geffet and Dagan 2009; Kotlerman *et al.* 2010). We describe balAPinc in detail in Section 6.1.

Glickman, Dagan and Shnarch (2006) define *lexical reference*, which is somewhat similar to lexical entailment, but it is defined relative to a specific text, such as a sentence. Mirkin, Dagan and Shnarch (2009b) define *entailment between lexical elements*, which includes entailment between words and non-compositional elements. Their definition is not based on substitutability; they accept many kinds of lexical entailment that are excluded by substitutability. Their definition involves what can be inferred from a lexical element in the context of some natural text.

Compared with the number of papers on lexical entailment, there is a relatively large body of literature on semantic relation classification (Rosario and Hearst 2001; Rosario *et al.* 2002; Nastase and Szpakowicz 2003; Turney 2006; Girju *et al.* 2007). Semantic relation classification has been a part of several Semantic Evaluation (SemEval)) exercises[2]:

- SemEval-2007 Task 4: Classification of Semantic Relations Between Nominals (Girju *et al.* 2007)
  – seven semantic relation classes.
- SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals (Hendrickx *et al.* 2010)
  – nine semantic relation classes.
- SemEval-2012 Task 2: Measuring Degrees of Relational Similarity (Jurgens *et al.* 2012)
  – seventy-nine semantic relation classes.

Only a few papers apply semantic relation classification to lexical entailment (Akhmatova and Dras 2009; Do and Roth 2010; Baroni *et al.* 2012; Do and Roth 2012).

[1] The RTE Challenge usually takes place once a year. For more information, see the Textual Entailment Portal at `http://aclweb.org/aclwiki`.
[2] For more information, see the SemEval Portal at `http://aclweb.org/aclwiki`.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

All these papers emphasize the hyponym–hypernym semantic relation, which is important for lexical entailment, but it is not the only relation that involves entailment.

Baroni *et al.* (2012) compared their ConVecs algorithm with the balAPinc measure and found no significant difference in their performance. They also consider how quantifiers (e.g. *some, all*) affect entailment.

Most algorithms for semantic relation classification are supervised (Rosario and Hearst 2001; Rosario *et al.* 2002; Nastase and Szpakowicz 2003; Turney 2006; Girju *et al.* 2007), although some are not (Hearst 1992). One objection to supervised learning for lexical entailment is that it can require a large quantity of labeled training data.

Baroni *et al.* (2012) offer an elegant solution to the training data issue, based on the observation that, in adjective–noun phrases, the adjective–noun pair generally entails the head noun. For example, *big cat* entails *cat*. This observation allows them to label a large quantity of training data with relatively little effort. However, their technique does not seem to be applicable to many of the relevant subcategories in Bejar *et al.*'s (1991) taxonomy. Our solution is to use word pairs that were labeled with Bejar *et al.*'s (1991) classes using Amazon's Mechanical Turk (Jurgens *et al.* 2012). The JMTH dataset given in Section 7.3 covers a much wider range of semantic relations than Baroni *et al.*'s (2012) dataset.

5 Performance measures

One difference between an asymmetric similarity measure (such as balAPinc) and a classification model based on supervised machine learning (such as ConVecs or SimDiffs) is that the former yields a real-valued score whereas the latter gives a binary-valued classification (0 = *does not entail* and 1 = *entails*). However, this difference is superficial. Many supervised learning algorithms (including the algorithms we use here) are able to generate a real-valued probability score (the probability that the given example belongs in class 1). Likewise, it is easy to generate a binary-valued class from a real-valued score by setting a threshold on the score.

In our experiments (Section 8), we evaluate all three algorithms as both real-valued asymmetric similarity measures and binary-valued classifiers. We use average precision as a performance measure for real-valued scores, following Kotlerman *et al.* (2010). We use precision, recall, F-measure, and accuracy as performance measures for binary-valued classification, following Baroni *et al.* (2012). The balAPinc measure (balanced average precision for distributional inclusion) is partly inspired by the average precision measure, thus it is useful to discuss average precision now, before we discuss balAPinc (in Section 6.1).

5.1 Average precision

Average precision was originally designed as a performance measure for information retrieval systems. Suppose we have issued a query to a search engine and it has returned a ranked list of N documents, sorted in descending order of their automatically estimated degree of relevance for our query. Assume that human judges have manually labeled all of the documents as either *relevant* or *irrelevant* for

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

the given query. Let P(r) be the fraction of the top r highest ranked documents that have the label *relevant*. That is, P(r) is the *precision* of the ranked list if we cut off the list after the rth document. Let rel(r) be 1 if the rth document is labeled *relevant*, 0 otherwise. Average precision is defined as follows (Buckley and Voorhees 2000):

$$\mathrm{AP} = \frac{\sum_{r=1}^{N} [\mathrm{P}(r) \cdot \mathrm{rel}(r)]}{\text{Total number of relevant documents}} \tag{1}$$

Average precision ranges from 0 (very poor performance) to 1 (perfect performance). Buckley and Voorhees (2000) demonstrate that average precision is more stable and more discriminating than several alternative performance measures for information retrieval systems.

The definition of average precision reflects a bias in information retrieval. For a typical query and a typical document collection, most documents are irrelevant and the emphasis is on finding the few relevant documents. In machine learning, if we have two classes, 0 and 1, they are usually considered equally important. Kotlerman *et al.* (2010) emphasize class 1 (*entails*), but we believe class 0 (*does not entail*) is also important. For example, the scoring of the RTE Challenge gives an equal reward for recognizing when a text sentence entails a hypothesis sentence and when it does not. Therefore, we report two variations of average precision, which we call AP_0 (average precision with respect to class 0) and AP_1 (average precision with respect to class 1), which we define in the next paragraph.

Suppose we have a dataset of word pairs manually labeled 0 and 1. Let N be the number of word pairs in the dataset. Let $\mathrm{M}(a, b) \in \Re$ be a measure that assigns a real-valued score to each word pair, $a:b$. Sort the pairs in descending order of their $\mathrm{M}(a, b)$ scores. Let $\mathrm{P}_1(r)$ be the fraction of the top r highest ranked pairs that have the label 1. Let $\mathrm{P}_0(r)$ be the fraction of the bottom r lowest ranked pairs that have the label 0. Let $\mathrm{C}_1(r)$ be 1 if the rth document from the top is labeled 1, 0 otherwise. Let $\mathrm{C}_0(r)$ be 1 if the rth document from the bottom is labeled 0, 0 otherwise. Let N_0 be the total number of pairs labeled 0 and let N_1 be the total number of pairs labeled 1. We define AP_0 and AP_1 as follows:

$$\mathrm{AP}_0 = \frac{\sum_{r=1}^{N} [\mathrm{P}_0(r) \cdot \mathrm{C}_0(r)]}{N_0} \tag{2}$$

$$\mathrm{AP}_1 = \frac{\sum_{r=1}^{N} [\mathrm{P}_1(r) \cdot \mathrm{C}_1(r)]}{N_1} \tag{3}$$

In their experiments, Kotlerman *et al.* (2010) report only AP_1. It is possible to increase a system's performance according to AP_1 at the cost of lower AP_0 performance. The formula for AP_1 is more sensitive to the labels in the top of the list. What happens at the bottom of the list has little impact on AP_1, because $\mathrm{P}_1(r)$ gives a low weight to labels at the bottom of the list. On the other hand, the formula for AP_0 is more sensitive to labels at the bottom of the list. If we focus on AP_1 and ignore

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

AP_0, we will prefer algorithms that get the top of the list right, even if they do poorly with the bottom of the list. Therefore, it is important to report both AP_0 and AP_1.

5.2 Precision, recall, F-measure, and accuracy

Like average precision, precision and recall were originally designed as performance measures for information retrieval systems. The precision of a system is an estimate of the conditional probability that a document is truly relevant to a query if the system says it is relevant. The recall of a system is an estimate of the conditional probability that the system will say that a document is relevant to a query if it is truly relevant.

There is a tradeoff between precision and recall; one may be optimized at the cost of the other. The F-measure is the harmonic mean of precision and recall. It is designed to reward a balance of precision and recall.

Accuracy is a natural and intuitive performance measure, but it is sensitive to the relative sizes of the classes. It is easy to interpret accuracy when we have two equal-sized classes, but it is difficult to interpret when one class is much larger than the other. The F-measure is a better measure when the classes are not balanced.

As with average precision, there are two variations of precision, recall, and F-measure, depending on whether we focus on class 0 or class 1. Let $\mathbf{C}$ be a 2×2 confusion matrix, where c_{ij} is the number of word pairs that are actually in class i and the algorithm has predicted that they are in class j (here $i, j \in \{0, 1\}$). We define precision, recall, and F-measure as follows:

$$\mathrm{Pre}_0 = c_{00}/(c_{00} + c_{10}) \tag{4}$$

$$\mathrm{Pre}_1 = c_{11}/(c_{11} + c_{01}) \tag{5}$$

$$\mathrm{Rec}_0 = c_{00}/(c_{00} + c_{01}) \tag{6}$$

$$\mathrm{Rec}_1 = c_{11}/(c_{11} + c_{10}) \tag{7}$$

$$\mathrm{F}_0 = 2 \cdot \mathrm{Pre}_0 \cdot \mathrm{Rec}_0/(\mathrm{Pre}_0 + \mathrm{Rec}_0) \tag{8}$$

$$\mathrm{F}_1 = 2 \cdot \mathrm{Pre}_1 \cdot \mathrm{Rec}_1/(\mathrm{Pre}_1 + \mathrm{Rec}_1) \tag{9}$$

Following standard practice (Witten, Frank and Hall 2011), we merge the two variations of each measure by taking their weighted averages, where the weights are determined by the class sizes:

$$w_0 = (c_{00} + c_{01})/(c_{00} + c_{01} + c_{10} + c_{11}) \tag{10}$$

$$w_1 = (c_{11} + c_{10})/(c_{00} + c_{01} + c_{10} + c_{11}) \tag{11}$$

$$\mathrm{Pre} = w_0 \cdot \mathrm{Pre}_0 + w_1 \cdot \mathrm{Pre}_1 \tag{12}$$

$$\mathrm{Rec} = w_0 \cdot \mathrm{Rec}_0 + w_1 \cdot \mathrm{Rec}_1 \tag{13}$$

$$\mathrm{F} = w_0 \cdot \mathrm{F}_0 + w_1 \cdot \mathrm{F}_1 \tag{14}$$

Finally, we define accuracy as usual:

$$\mathrm{Acc} = 100 \cdot (c_{00} + c_{11})/(c_{00} + c_{01} + c_{10} + c_{11}) \tag{15}$$

The factor of 100 converts the accuracy from a fraction to a percentage score.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

6 Three approaches to lexical entailment

In this section we discuss the three approaches to RLE and describe the algorithms for each approach in detail. All three approaches are based on word–context matrices. For an introduction to the concepts behind word–context matrices, see the survey paper by Turney and Pantel (2010).

In preliminary experiments with our development datasets, Dev1 and Dev2, we tuned the three approaches to optimize their performance. We describe how Dev1 and Dev2 were generated in Section 8.1.1. For each algorithm, we selected the matrix or matrices that were most accurate with the development data. For both balAPinc and ConVecs, we chose the word–context matrix from Turney *et al.* (2011). For SimDiffs, we chose two word–context matrices from Turney (2012).[3]

ConVecs and SimDiffs use support vector machines (SVMs) for supervised learning. We used the development datasets to select the best kernels for the SVMs. The best kernel for ConVecs was a second-degree polynomial kernel and the best kernel for SimDiffs was a radial basis function (RBF) kernel.

6.1 The context inclusion hypothesis: balAPinc

We include balAPinc in our experiments because Kotlerman *et al.* (2010) experimentally compared it with a wide range of asymmetric similarity measures and found that balAPinc had the best performance. The balAPinc asymmetric similarity measure is a balanced combination of the asymmetric APinc measure (Kotlerman *et al.* 2010) with the symmetric LIN measure (Lin 1998). Balance is achieved by using the geometric mean:

$$\text{balAPinc}(u,v) = \sqrt{\text{APinc}(u,v) \cdot \text{LIN}(u,v)} \tag{16}$$

To define APinc and LIN, we must first introduce some terminology. Kotlerman *et al.* (2010) define balAPinc with terminology from set theory, whereas ConVecs and SimDiffs are more naturally defined with terminology from linear algebra. We will use the set theoretical terminology of Kotlerman *et al.* (2010) and the linear algebraic terminology of Turney and Pantel (2010) so that the reader can easily see both perspectives. This leads to a small amount of redundancy, but we believe it is helpful to connect the two points of view.[4]

First, some linear algebraic notation: Suppose that we have a word–context matrix in which each row vector corresponds to a word and each column vector corresponds to a context. Let $\mathbf{F}$ be the matrix of raw co-occurrence frequencies. If w is the word corresponding to the ith row vector, $\mathbf{f}_{i:}$, and c is the context corresponding to the jth column vector, $\mathbf{f}_{:j}$, then f_{ij} is the number of times w occurs in the context c in the given corpus.

Let the matrix $\mathbf{X}$ be the result of calculating the positive point-wise mutual information (PPMI) between the word w and the context c for each element f_{ij} in $\mathbf{F}$ (Bullinaria and Levy 2007; Turney and Pantel 2010). PPMI takes the raw co-occurrence frequencies and transforms them to weights that represent the importance

[3] Copies of all three matrices used here are available from the first author on request.

[4] ConVecs and SimDiffs are fundamentally linear algebraic in conception, whereas balAPinc is fundamentally set theoretic. We cannot readily describe all three systems with only one kind of notation.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

of a given context for a given word. The PPMI matrix $\mathbf{X}$ is typically sparse (most cells are zero) and no cells are negative.[5]

The matrix $\mathbf{X}$ has the same number of rows (n_r) and columns (n_c) as the raw frequency matrix $\mathbf{F}$. The value of an element x_{ij} in $\mathbf{X}$ is defined as follows (Turney and Pantel 2010):

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{ij}} \tag{17}$$

$$p_{i*} = \frac{\sum_{j=1}^{n_c} f_{ij}}{\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{ij}} \tag{18}$$

$$p_{*j} = \frac{\sum_{i=1}^{n_r} f_{ij}}{\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} f_{ij}} \tag{19}$$

$$\text{pmi}_{ij} = \log\left(\frac{p_{ij}}{p_{i*}p_{*j}}\right) \tag{20}$$

$$x_{ij} = \begin{cases} \text{pmi}_{ij} & \text{if } \text{pmi}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Now, some set theoretical notation: Given a word w corresponding to the ith row in $\mathbf{X}$, let F_w be the set of contexts for which x_{ij} is nonzero. That is, $c \in F_w$ if and only if $x_{ij} \neq 0$, where w corresponds to row i and c corresponds to column j. We may think of the contexts in the set F_w as *features* that characterize the word w. Let $|F_w|$ be the number of features in F_w. If w corresponds to the ith row in $\mathbf{X}$, then $|F_w|$ is the number of nonzero cells in the ith row vector, $\mathbf{x}_{i:}$.

Each feature f in F_w corresponds to a PPMI value x_{ij}. Let us rank the features in F_w in descending order of their corresponding PPMI values. Let f_{wr} be the rth feature in the ranking of F_w, where r ranges from 1 to $|F_w|$. Let $\text{rank}(f, F_w)$ be the rank of f in F_w. Thus, $\text{rank}(f_{wr}, F_w) = r$. We want to normalize this rank so that it ranges between 0 and 1, where higher PPMI values are closer to 1 and lower PPMI values are closer to 0. The function $\text{rel}(f, F_w)$ provides this normalization:

$$\text{rel}(f, F_w) = \begin{cases} 1 - \frac{\text{rank}(f, F_w)}{|F_w|+1} & \text{if } f \in F_w \\ 0 & \text{if } f \notin F_w \end{cases} \tag{22}$$

We may interpret $\text{rel}(f, F_w)$ as a measure of the importance of the feature f for characterizing the word w. This function is called *rel* because it is somewhat analogous to *relevance* in information retrieval.

[5] Other measures of word association may be used instead of PPMI. See Chapter 5 of Manning and Schütze (1999) for a good survey of association measures.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Recall the *context inclusion hypothesis*: If a word u tends to occur in a subset of the contexts in which a word v occurs (v contextually includes u), then u (the narrower term) tends to entail v (the broader term). Suppose we test the features of u, $f \in F_u$, in order of their rank, r, to see which features of u are *contextually included* in v. Let $\text{inc}(r, F_u, F_v)$ be the set consisting of those features, among the first r features in F_u, that are included in F_v:

$$\text{inc}(r, F_u, F_v) = \{f \mid \text{rank}(f, F_u) \leq r \text{ and } f \in (F_u \cap F_v)\} \tag{23}$$

The size of this set, $|\text{inc}(r, F_u, F_v)|$, ranges from 0 to r, where $r \leq |F_u|$. The function $\text{P}(r, F_u, F_v)$ normalizes the size to range from 0 to 1:

$$\text{P}(r, F_u, F_v) = \frac{|\text{inc}(r, F_u, F_v)|}{r} \tag{24}$$

We may interpret $\text{P}(r, F_u, F_v)$ as a measure of the density of F_v features among the top r features of F_u. This function is called P because it is somewhat analogous to *precision* in information retrieval.

Now we are ready to define APinc:

$$\text{APinc}(u, v) = \frac{\sum_{r=1}^{|F_u|} [\text{P}(r, F_u, F_v) \cdot \text{rel}(f_{ur}, F_v)]}{|F_u|} \tag{25}$$

APinc is a variation of the average precision measure, originally developed for measuring the performance of information retrieval systems (see Section 5.1). Consider the first term in the sum, $r = 1$. If f_{u1}, the highest ranking feature in F_u, is included in F_v, then $\text{P}(1, F_u, F_v)$ will be 1, otherwise it will be 0. If f_{u1} is in F_v, then the product $\text{P}(1, F_u, F_v) \cdot \text{rel}(f_{u1}, F_v)$ reduces to $\text{rel}(f_{u1}, F_v)$, the importance of the feature f_{u1} for the word v. APinc will have a high score when the most important features of u are also important features of v. APinc is asymmetric because it does not require that the most important features of v are important features of u.

Let $w_u(f)$ be the weight of the feature f in the word u. The weight is given by the PPMI value in $\mathbf{X}$. If u corresponds to the ith row and f corresponds to the jth column, then $w_u(f) = x_{ij}$. (It may seem redundant to have both $w_u(f)$ and x_{ij}. The first is set theoretical and the second is linear algebraic.) LIN is defined as follows (Lin 1998):

$$\text{LIN}(u, v) = \frac{\sum_{f \in F_u \cap Fv} [w_u(f) + w_v(f)]}{\sum_{f \in F_u} w_u(f) + \sum_{f \in F_v} w_v(f)} \tag{26}$$

In balAPinc (16), the LIN measure is combined with the APinc measure because the APinc measure by itself tends to be sensitive to cases where $|F_u|$ or $|F_v|$ are unusually small (Kotlerman *et al.* 2010).

There are two parameters, $\max_F$ and T, that can be varied to control the performance of balAPinc. The parameter $\max_F$ sets the maximum number of features for each word. For a given word w, we calculate all of the features, F_w. If $|F_w| > \max_F$, then we remove the lowest ranking features until $|F_w| = \max_F$. This reduces the

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

impact of low-ranking features on the APinc score. The parameter T is a threshold for classification. If balAPinc$(u,v) < T$, then the word pair $u:v$ is classified as 0 (*does not entail*); otherwise $u:v$ is classified as 1 (*entails*). We describe how these parameters are tuned in Section 8.

Kotlerman *et al.* (2010) do not use the threshold T, since they do not evaluate balAPinc as a classifier. They also do not use the parameter $\max_F$, although their analysis supports the utility of this parameter; see Section 5.4.4 of Kotlerman *et al.* (2010).

In the experiments with balAPinc in Section 8, the PPMI matrix **X** is the same matrix as used by Turney *et al.* (2011). The matrix has 114,501 rows and 139,246 columns. The rows correspond to single and multi-word entries (n-grams) in WordNet and the columns correspond to unigrams in WordNet, distinguished according to whether they appear in the left or right context of the given n-gram. The window size for context is four words to the left and four words to the right of the n-gram. The matrix has a density (percentage of nonzero values) of 1.22%.

The PPMI matrix is based on a corpus of 5×10^{10} words, collected from university websites by a webcrawler.[6] The corpus was indexed with the Wumpus search engine (Büttcher and Clarke 2005), which is designed for passage retrieval rather than document retrieval.[7] Suppose f_{ij} is an element in the matrix of raw co-occurrence frequencies **F**. The ith row of the matrix corresponds to an n-gram w in WordNet and the jth column of the matrix corresponds to a unigram c. The value of f_{ij} was calculated by sending the query w to Wumpus and counting the frequency of c in the retrieved passages. The matrix is described in detail in Section 2.1 of Turney *et al.* (2011).

It is common to smooth the PPMI matrix by applying a truncated singular value decomposition (SVD) (Turney and Pantel 2010). On the development datasets, we experimented with smoothing the matrix, but the results were poor. The problem is that the truncated SVD yields a matrix with a density of 100%, but balAPinc is designed for highly sparse matrices.

Consider (23) for example. If the matrix has a density of 100%, then all of the contexts (all of the matrix columns) are nonzero, so F_u and F_v are simply the entire set of features, and $(F_u \cap F_v)$ is also the entire set of features. Likewise, in (26) all of the sums, $\sum_f$, range over the entire set of features. The equations behind balAPinc are based on the assumption that most of the elements in the matrix are zero (i.e. the matrix is sparse), but this assumption is false if we apply a truncated SVD.

In the experiments in Section 8, we use the raw PPMI matrix with no SVD smoothing. Baroni *et al.* (2012) also found that balAPinc works better without SVD smoothing (see their Footnote 3).

6.2 The context combination hypothesis: ConVecs

With the ConVecs algorithm, Baroni *et al.* (2012) were able to match the performance of balAPinc. In ConVecs, we represent a word pair $a:b$ by the concatentation of the context vectors **a** for a and **b** for b. We apply a supervised learning algorithm to

[6] The corpus was collected by Charles Clarke at the University of Waterloo.
[7] Wumpus is available at `http://www.wumpus-search.org/`.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

a training set of word pairs, where each word pair is represented by concatenated context vectors that are labeled *entails* or *does not entail*. The supervised learning algorithm generates a classification model which enables us to assign labels to new word pairs not present in the training data.

Let $\mathbf{X}$ be a word–context matrix where the value of the cell x_{ij} in $\mathbf{X}$ is given by the PPMI between the ith word w and the jth context c. In our experiments we use the word–context matrix $\mathbf{X}$ from Turney *et al.* (2011), as in Section 6.1, but now we smooth $\mathbf{X}$ with a truncated SVD.

Singular value decomposition decomposes $\mathbf{X}$ into the product of three matrices $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are in column orthonormal form (i.e. the columns are orthogonal and have unit length, $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$) and $\mathbf{\Sigma}$ is a diagonal matrix of singular values (Golub and Van Loan 1996). If $\mathbf{X}$ is of rank r, then $\mathbf{\Sigma}$ is also of rank r. Let $\mathbf{\Sigma}_k$, where $k < r$, be the diagonal matrix formed from the top k singular values, and let $\mathbf{U}_k$ and $\mathbf{V}_k$ be the matrices produced by selecting the corresponding columns from $\mathbf{U}$ and $\mathbf{V}$. The matrix $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$ is the matrix of rank k that best approximates the original matrix $\mathbf{X}$, in that it minimizes the approximation errors. That is, $\hat{\mathbf{X}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$ minimizes $\|\hat{\mathbf{X}} - \mathbf{X}\|_F$ over all matrices $\hat{\mathbf{X}}$ of rank k, where $\|\dots\|_F$ denotes the Frobenius norm (Golub and Van Loan 1996).

We represent a word pair $a:b$ using row vectors from the matrix $\mathbf{U}_k\mathbf{\Sigma}_k^p$. If a and b correspond to row vectors $\mathbf{a}$ and $\mathbf{b}$ in $\mathbf{U}_k\mathbf{\Sigma}_k^p$, then $a:b$ is represented by the $2k$-dimensional vector that is the concatenation of $\mathbf{a}$ and $\mathbf{b}$. We normalize $\mathbf{a}$ and $\mathbf{b}$ to unit length before we concatenate them.

There are two parameters in $\mathbf{U}_k\mathbf{\Sigma}_k^p$ that need to be set. The parameter k controls the number of latent factors and the parameter p adjusts the weights of the factors, by raising the corresponding singular values in $\mathbf{\Sigma}_k^p$ to the power p.

The parameter k is well known in the literature (Landauer *et al.* 2007), but p is less familiar. Caron (2001) introduced p for improving the performance of truncated SVD with term–document matrices in information retrieval. The use of p to improve the performance with word–context matrices in lexical semantics is supported by the empirical evaluations of Bullinaria and Levy (2012) and Turney (2012). In the following experiments (Section 8), we explore a range of values for p and k. Baroni *et al.* (2012) use $k = 300$ and $p = 1$.[8]

Recall the *context combination hypothesis*: The tendency of a to entail b is correlated with some learnable function of the contexts in which a occurs and the contexts in which b occurs; some conjunctions of contexts tend to indicate entailment and others tend to indicate a lack of entailment. Given the context combination hypothesis, vector concatenation is a natural way to represent $a:b$ for learning lexical entailment.

For their supervised learning algorithm, Baroni *et al.* (2012) used Weka with LIBSVM.[9] They used a polynomial kernel for SVM. We also use Weka and a

[8] Baroni *et al.* (2012) mention $k = 300$ in their Footnote 3. In personal communication in November 2012, they said they used $p = 1$.

[9] Weka is available at `http://www.cs.waikato.ac.nz/ml/weka/` and LIBSVM is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

polynomial kernel, but we use the sequential minimal optimization (SMO) SVM in Weka (Platt 1998) because it can generate real-valued probability estimates, as well as binary-valued classes. The probability estimates are based on fitting the outputs of the SVM with logistic regression models (Witten, Frank, and Hall 2011).

We tried various kernels with ConVecs on the development datasets (Dev1 and Dev2; see Section 8.1.1), and found that a second-degree polynomial kernel had the best performance. We use default settings for the polynomial kernel SMO SVM in Weka, except we disable normalization, because the vectors are already normalized to the same length.

It seems to us that ConVecs is a good algorithm for a generic semantic relation, but a representation that takes advantage of some background knowledge about lexical entailment might require less training data. One thing we know about lexical entailment is $a \models a$, for any a. ConVecs can only reliably recognize that $a \models a$ if a is similar to some x such that the word pair $x:x$ appears in the training data and has been labeled *entails*. To cover a broad range of possible values for a, there must be many different $x:x$ pairs in the training data. The ConVecs representation does not make efficient use of the training data.

6.3 The similarity differences hypothesis: SimDiffs

SimDiffs uses two different word–context matrices, a *domain* matrix, **D**, and a *function* matrix, **F** (Turney 2012). The domain matrix is designed for measuring the domain similarity between two words (similarity of topic, subject, or field). For example, *carpenter* and *wood* have a high degree of domain similarity; they both come from the domain of *carpentry*. The function matrix is designed for measuring function similarity (similarity of role, relationship, or usage). For example, *carpenter* and *mason* have a high degree of function similarity; they both function as *artisans*.

The two matrices use different types of context. The domain matrix uses the *nouns* that occur near a given word as the context for the word, whereas the function matrix uses the *verbs* that occur near the given word. The part-of-speech information was generated with the OpenNLP tagger.[10] Our motivation for using two matrices in SimDiffs is to generate a larger and more varied set of features for the supervised learning algorithm. Turney (2012) demonstrated that domain and function matrices work together synergetically when applied to semantic relations.

In experiments with the development datasets (Dev1 and Dev2), we tried using the domain and function matrices with balAPinc and ConVecs, but both algorithms worked better with the word–context matrix from Turney *et al.* (2011). For SimDiffs, the combination of the domain and function matrices from Turney (2012) had the best performance on the development datasets.

Both **D** and **F** use PPMI and SVD, as in Section 6.2. This results in a total of four parameters that need to be tuned, k_d and p_d for domain space and k_f and p_f for function space. In the following experiments (Section 8), to simplify the search through parameter space, we make $k_d = k_f$ and $p_d = p_f$.

[10] OpenNLP is available at `http://opennlp.apache.org/`.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

The domain and function matrices are based on the same corpus as the word–context matrix from Turney *et al.* (2011). Wumpus was used to index the corpus and search for passages in the same way as described in Section 6.1. **D** has 114,297 rows and 50,000 columns. The PPMI matrix has a density of 2.62%. **F** has 114,101 rows and 50,000 columns. The PPMI matrix has a density of 1.21%. For both matrices, truncated SVD results in a density of 100%.

The rows for both the matrices correspond to single and multi-word entries (*n*-grams) in WordNet. The columns are more complex; Turney (2012) provides a detailed description of the columns and other aspects of the matrices. The matrices have different numbers of rows because, before applying SVD, we removed rows that were entirely zero. The function matrix, with its lower density, had more zero-valued rows than the domain matrix.

Suppose that the words a and b correspond to the row vectors $\mathbf{a}_\mathrm{d}$ and $\mathbf{b}_\mathrm{d}$ in the domain matrix, **D**. Let $\mathrm{sim}_\mathrm{d}(a,b)$ be the similarity of a and b in domain space, as measured by the cosine of the angle between $\mathbf{a}_\mathrm{d}$ and $\mathbf{b}_\mathrm{d}$, $\cos(\mathbf{a}_\mathrm{d}, \mathbf{b}_\mathrm{d})$. Likewise, let $\mathrm{sim}_\mathrm{f}(a,b)$ be $\cos(\mathbf{a}_\mathrm{f}, \mathbf{b}_\mathrm{f})$, the cosine in function space.

Let R be a set of reference words. Recall the *similarity differences hypothesis*: The tendency of a to entail b is correlated with some learnable function of the differences in their similarities, $\mathrm{sim}(a,r) - \mathrm{sim}(b,r)$, to a set of reference words, $r \in R$; some differences tend to indicate entailment and others tend to indicate a lack of entailment. In SimDiffs, we represent a word pair $a:b$ with a feature vector comprising four sets of features, S_1, S_2, S_3, and S_4, defined as follows:

$$S_1 = \{\mathrm{sim}_\mathrm{d}(a,r) - \mathrm{sim}_\mathrm{d}(b,r) \mid r \in R\} \tag{27}$$

$$S_2 = \{\mathrm{sim}_\mathrm{f}(a,r) - \mathrm{sim}_\mathrm{f}(b,r) \mid r \in R\} \tag{28}$$

$$S_3 = \{\mathrm{sim}_\mathrm{d}(a,r) - \mathrm{sim}_\mathrm{f}(b,r) \mid r \in R\} \tag{29}$$

$$S_4 = \{\mathrm{sim}_\mathrm{f}(a,r) - \mathrm{sim}_\mathrm{d}(b,r) \mid r \in R\} \tag{30}$$

S_1 is the difference between a and b in domain space, with respect to their similarities to the reference words, R. S_2 is the difference between a and b in function space. S_1 and S_2 are based on differences in the same spaces, whereas S_3 and S_4 are based on differences in different spaces.

The cross-spatial differences (S_3 and S_4) may seem counterintuitive. Consider the example *murder* $\models$ *death*, suggested by the quotation from Zhitomirsky-Geffet and Dagan (2009) in Section 3. Murder typically involves two people, the victim and the aggressor, whereas death typically involves one person, the deceased. This suggests that there is a functional difference between the words, hence the function similarities of *murder* may be quite different from the function similarities of *death*. However, perhaps the domain similarities of *murder* are somewhat similar to the function similarities of *death* (S_3) or perhaps the function similarities of *murder* are somewhat similar to the domain similarities of *death* (S_4). We include these similarities here to see if the supervised learning algorithm can make use of these.

For R, the set of reference words, we use 2,086 words from Basic English (Ogden 1930).[11] Thus, a word pair $a:b$ is represented by $2{,}086 \times 4 = 8{,}344$ features. The

[11] This word list is available at `http://ogden.basic-english.org/word2000.html`.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

words of Basic English were selected by Ogden (1930) to form a core vocabulary, sufficient to represent most other English words by paraphrasing. We chose this set of words because it is small enough to keep the number of features manageable yet broad enough to cover a wide range of concepts. Other reference words may also be suitable; this is a topic for future work.

We mentioned in Section 6.2 that ConVecs may be inefficient for learning $a \models a$. On the other hand, consider how $a \models a$ is represented in SimDiffs. Looking at (27) and (28), we see that, given the word pair $a:a$, every feature in S_1 and S_2 will have the value zero. Therefore, it should not take many examples of $x:x$ in the training data to learn that $a \models a$, for any a.

For our supervised learning algorithm, we use the SMO SVM in Weka. Based on the experiments with the development datasets (Dev1 and Dev2), we use a radial basis function kernel. We use the default settings, except we disable normalization. We generate probability estimates for the classes.

7 Three datasets for lexical entailment

This section describes the three datasets we use in our experiments. The first two datasets have been used in the past for lexical entailment research. The third dataset has been used for semantic relation research; this is the first time it has been used for lexical entailment. We refer to each dataset by the initials of the authors of the paper in which it was first reported.

7.1 The KDSZ dataset

The KDSZ dataset was introduced by Kotlerman *et al.* (2010) to evaluate balAPinc. The dataset contains 3,772 word pairs, 1,068 labeled *entails*, and 2,704 labeled *does not entail*. It was created by taking a dataset of 3,200 labeled word pairs from Zhitomirsky-Geffet and Dagan (2009) and adding 572 more labeled pairs.[12] The labeling of the original subset of 3,200 pairs is described in detail by Zhitomirsky-Geffet and Dagan (2009). The definition of lexical entailment that the judges used was the *substitutional definition* given in Section 2. Three judges labeled the pairs with inter-annotator agreement between any two of the three judges varying from 90.0% to 93.5%.

This dataset has two properties that complicate the experiments. First, the class sizes are not balanced; 71.7% of the pairs are labeled *does not entail* and 28.3% are labeled *entails*. Second, although every word pair is unique, there are a few words that appear many times, in many different pairs. We address these points in our experiments.

The words in the word pairs are mainly unigrams, but there are a few bigrams (*central bank, higher education, state government*). Fortunately, all of the bigrams appear in WordNet, so they have corresponding row vectors in our matrices.

7.2 The BBDS dataset

The BBDS dataset was created by Baroni *et al.* (2012) and has been applied to evaluating both balAPinc and ConVecs. In their paper, Baroni *et al.* (2012) discuss

[12] Personal communication with Zhitomirsky-Geffet in March 2012.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 1. *Types of information about the classes in the relation taxonomy*

	Type	Example
1.	ID	1a
2.	Category	class-inclusion
3.	Subcategory	taxonomic
4.	Paradigmatic examples ($x:y$)	flower:tulip, emotion:rage, poem:sonnet
5.	Relational schema	y is a kind/type/instance of x
6.	Turker examples (Phase 1)	fruit:grape, song:opera, rodent:mouse, ...
7.	Turker ratings (Phase 2)	fruit:grape: 24.0, song:opera: -18.0, ...

several different datasets. We use the dataset they call $N_1 \models N_2$, described in their Section 3.3. The dataset contains 2,770 word pairs, 1,385 labeled *entails*, and 1,385 labeled *does not entail*. All of the 1,385 pairs labeled *entails* are hyponym–hypernym noun–noun pairs, such as *pope* $\models$ *leader*. The pairs were generated automatically from WordNet and then validated manually.

Although the class sizes are balanced, 50% *entails* and 50% *does not entail*, the BBDS dataset is not representative of the variety of semantic relations that involve entailment as we will see in Section 7.3. Also, although every word pair is unique, there are a few words that appear many times. All of the word pairs comprise unigrams and all the unigrams appear in WordNet, so they have corresponding row vectors in our matrices.

7.3 The JMTH dataset

Jurgens *et al.* (2012) created a semantic relation dataset for SemEval-2012 Task 2: Measuring Degrees of Relational Similarity.[13] This dataset contains 3,218 word pairs labeled with seventy-nine types of semantic relations. In this section, we describe the original SemEval-2012 dataset and the process we used to convert the dataset into 2,308 word pairs, 1,154 labeled *entails*, and 1,154 labeled *does not entail*.

The original dataset consists of word pairs labeled using the relation classification scheme of Bejar *et al.* (1991). This is a hierarchical classification system with ten high-level categories, each of which has between five and ten subcategories, for a total of seventy-nine distinct subcategories.

For each subcategory in Bejar *et al.*'s (1991) relation taxonomy, we have several types of information, as shown in Table 1. The first four types of information come from Bejar *et al.* (1991) and the rest were added by Jurgens *et al.* (2012).[14]

The original SemEval-2012 dataset was generated in two phases, using Amazon's Mechanical Turk (Jurgens *et al.* 2012).[15] We refer to Mechanical Turk workers as Turkers. In the first phase, for each of the seventy-nine subcategories, Turkers were

[13] The dataset is available at `https://sites.google.com/site/semeval2012task2/`. We used the package called SemEval-2012-Gold-Ratings.

[14] All of this information is provided in the file SemEval-2012-Complete-Data-Package at `https://sites.google.com/site/semeval2012task2/download`.

[15] See `https://www.mturk.com/`.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

shown paradigmatic examples of word pairs in the given subcategory, and they were asked to generate more word pairs of the same semantic relation type. In the second phase, for each of the seventy-nine subcategories, Turkers were shown word pairs that were generated in the first phase, and they were asked to rate the pairs according to their degree of prototypicality for the given semantic relation type. (See Table 1 for examples of the results of the two phases.)

We transformed the original SemEval-2012 semantic relation dataset to the new lexical entailment dataset in the following four steps:

(1) *Cleaning*: To improve the quality of the dataset, we removed the ten lowest rated word pairs from each subcategory. Since the original dataset has 3,218 word pairs, the average subcategory has 40.7 word pairs. Our cleaning operation reduced this to 30.7 pairs per subcategory, a total of 2,428 word pairs ($3218 - 79 \times 10 = 2,428$).
(2) *Doubling*: For each word pair $a:b$ labeled with a subcategory X, we generated a new word pair $b:a$ and labeled it X^{-1}. For example, *car:engine* is labeled object:component, so we created the pair *engine:car* and labeled it object:component^{-1}. This increased the number of pairs to 4,856 and the number of subcategories to 158.
(3) *Mapping*: We then mapped the 158 subcategory labels to the labels 0 (*does not entail*) and 1 (*entails*). The mapping is given in Tables 2 and 3. We assume that all word pairs within a subcategory belong to the same class (either all entail or none entail). (This assumption is tested in Section 7.3.2.) The result of mapping was 4,856 word pairs with two labels. There were 1,154 pairs labeled 1 and 3,702 pairs labeled 0.
(4) *Balancing*: To make a balanced dataset, we randomly removed pairs labeled 0 until there were 1,154 pairs labeled 0 and 1,154 pairs labeled 1, a total of 2,308 word pairs.

Here is how to interpret Tables 2 and 3. Given the pair *anesthetic:numbness* with the label instrument:goal, we see from Table 3 (ID 8f in the semantic relation taxonomy) that $a \models b$ has the value 1, so we map the label instrument:goal to the label 1 (*entails*). Given the pair *numbness:anesthetic* labeled instrument:goal^{-1}, we see from the table (ID 8f in the semantic relation taxonomy) that $b \models a$ has the value 0, so we map the label instrument:goal^{-1} to the label 0 (*does not entail*). In other words, *anesthetic* $\models$ *numbness*:

Text: *Due to the anesthetic, Jane felt no pain.*
Hypothesis: *Due to the numbness, Jane felt no pain.*
Text $\models$ Hypothesis

However, *numbness* $\not\models$ *anesthetic*; for example, the numbness might be caused by cold temperature, not anesthetic:

Text: *George had frostbite. Due to the numbness, he felt no pain.*
Hypothesis: *Due to the anesthetic, he felt no pain.*
Text $\not\models$ Hypothesis

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 2. *Semantic relation categories 1 to 5, based on Bejar* et al. *(1991)*

ID	Category	Subcategory	Example (*a*:*b*)	$a \models b$	$b \models a$
1a	Class-inclusion	taxonomic	flower:tulip	0	1
1b	Class-inclusion	functional	weapon:knife	0	1
1c	Class-inclusion	singular:collective	cutlery:spoon	0	1
1d	Class-inclusion	plural:collective	dishes:saucers	0	1
1e	Class-inclusion	class:individual	mountain:Everest	0	1
2a	Part-whole	object:component	car:engine	1	0
2b	Part-whole	collection:member	forest:tree	1	0
2c	Part-whole	mass:potion	time:moment	1	0
2d	Part-whole	event:feature	banquet:food	1	0
2e	Part-whole	stage:activity	kickoff:football	0	1
2f	Part-whole	item:topological part	room:corner	1	0
2g	Part-whole	object:stuff	glacier:ice	1	0
2h	Part-whole	creature:possession	millionaire:money	1	0
2i	Part-whole	item:nonpart	horse:wings	0	0
2j	Part-whole	item:ex-part	prisoner:freedom	0	0
3a	Similar	synonymity	car:auto	1	1
3b	Similar	dimension similarity	simmer:boil	0	0
3c	Similar	dimension excessive	concerned:obsessed	0	1
3d	Similar	dimension naughty	listen:eavesdrop	0	1
3e	Similar	conversion	grape:wine	0	0
3f	Similar	attribute similarity	rake:fork	0	0
3g	Similar	coordinates	son:daughter	0	0
3h	Similar	change	crescendo:sound	1	0
4a	Contrast	contradictory	alive:dead	0	0
4b	Contrast	contrary	happy:sad	0	0
4c	Contrast	reverse	buy:sell	0	0
4d	Contrast	directional	left:right	0	0
4e	Contrast	incompatible	slow:stationary	0	0
4f	Contrast	asymmetric contrary	hot:cool	0	0
4g	Contrast	pseudoantonym	popular:shy	0	0
4h	Contrast	defective	limp:walk	1	0
5a	Attribute	item:attribute	glass:fragile	1	0
5b	Attribute	attribute:condition	edible:eaten	0	0
5c	Attribute	object:state	beggar:poverty	1	0
5d	Attribute	attribute:state	contentious:conflict	1	0
5e	Attribute	object:typical act	soldier:fight	0	0
5f	Attribute	attribute:typical act	viable:live	0	0
5g	Attribute	act:act attribute	creep:slow	1	0
5h	Attribute	act:object attribute	sterilize:infectious	0	0
5i	Attribute	act:resultant	rain:wet	1	0

Of the seventy-nine subcategories, twenty-five were labeled with class 1 for $a \models b$ and twelve were labeled with class 1 for $b \models a$. Note that two of the subcategories are labeled with class 1 for both $a \models b$ and $b \models a$ (IDs 3a and 9e). This shows that our transformation handles both symmetric and asymmetric semantic relations.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 3. *Semantic relation categories 6 to 10, based on Bejar* et al. *(1991)*

ID	Category	Subcategory	Example (*a*:*b*)	$a \models b$	$b \models a$
6a	Non-attribute	item:nonattribute	harmony:discordant	0	0
6b	Non-attribute	attr.:noncondition	brittle:molded	0	0
6c	Non-attribute	object:nonstate	laureate:dishonor	0	0
6d	Non-attribute	attr.:nonstate	dull:cunning	0	0
6e	Non-attribute	obj.:atypical act	recluse:socialize	0	0
6f	Non-attribute	attr.:atypical act	reticent:talk	0	0
6g	Non-attribute	act:act nonattr.	creep:fast	0	0
6h	Non-attribute	act:object nonattr.	embellish:austere	0	0
7a	Case relations	agent:object	tailor:suit	0	0
7b	Case relations	agent:recipient	doctor:patient	0	0
7c	Case relations	agent:instrument	farmer:tractor	0	0
7d	Case relations	act:object	plow:earth	0	0
7e	Case relations	act:recipient	bequeath:heir	1	0
7f	Case relations	object:recipient	speech:audience	0	0
7g	Case relations	object:instrument	pipe:wrench	0	0
7h	Case relations	recipient:instr.	graduate:diploma	0	0
8a	Cause-purpose	cause:effect	enigma:puzzlement	1	0
8b	Cause-purpose	cause:counteract	hunger:eat	0	0
8c	Cause-purpose	enabler:object	match:candle	0	0
8d	Cause-purpose	act:goal	flee:escape	0	0
8e	Cause-purpose	agent:goal	climber:peak	0	0
8f	Cause-purpose	instrument:goal	anesthetic:numbness	1	0
8g	Cause-purpose	instrument:use	abacus:calculate	0	0
8h	Cause-purpose	prevention	pesticide:vermin	0	0
9a	Space-time	location:item	arsenal:weapon	1	0
9b	Space-time	location:product	bakery:bread	1	0
9c	Space-time	location:activity	highway:driving	0	0
9d	Space-time	location:instr.	beach:swimsuit	0	0
9e	Space-time	contiguity	coast:ocean	1	1
9f	Space-time	time:activity	childhood:play	0	0
9g	Space-time	time:associated	retirement:pension	0	0
9h	Space-time	sequence	prologue:narrative	0	0
9i	Space-time	attachment	belt:waist	0	0
10a	Reference	sign:significant	siren:danger	1	0
10b	Reference	expression	hug:affection	1	0
10c	Reference	representation	person:portrait	0	1
10d	Reference	plan	blueprint:building	0	1
10e	Reference	knowledge	psychology:minds	1	0
10f	Reference	concealment	disguise:identity	1	0

7.3.1 Making the mapping table

We (authors of this paper) each independently created a mapping like Tables 2 and 3. We disagreed on twelve of the 158 (79×2) mappings (92.4% agreement). We compared our tables and discussed them until we arrived at a consensus. For all twelve disagreements, our consensus was to label them 0. Tables 2 and 3 are the result of our consensus.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 4. *Agreement and entailment for the two definitions of lexical entailment*

Measure	Annotation	Relational	Substitutional
Percent agreement	Turney versus Mohammad	81	89
Percent agreement	Turney versus SemEval	81	68
Percent agreement	Mohammad versus SemEval	70	65
Percent entailing	Turney	51	22
Percent entailing	Mohammad	48	25
Percent entailing	SemEval	50	50

We used the first five types of information in Table 1 to decide how to map relation classes to entailment classes. Before we each independently created a mapping table, we agreed to approach the task as follows:

Procedure for annotation:

(1) The relational schemas have more weight than the paradigmatic examples when deciding whether x entails y or y entails x.
(2) Consider each of the paradigm pairs as instances of the given relational schema, that is, interpret the pairs in the light of the schema. If the three paradigmatic pairs are such that x entails y, when interpreted this way, then annotate the given category as 'x entails y', and likewise for y entails x. If two out of three paradigmatic pairs are such that x entails y, and the pair that is the exception seems unusual in some way, make a note about the exceptional pair, for later discussion.
(3) If any of the paradigmatic pairs are in the wrong order, correct their order before proceeding. Make a note of the correction.

We then compared our tables and combined them to form the final Tables 2 and 3.

7.3.2 *Inter-annotator agreement*

As we have mentioned above, we assume that all word pairs within a subcategory belong to the same class (either all entail or none entail). To test this assumption, we randomly selected one hundred word pairs, fifty labeled *entails* and fifty labeled *does not entail*. We hid the labels and then we each independently labeled the pairs manually, first using the relational definition of lexical entailment and then a second time using the substitutional definition of lexical entailment (see Section 2). Table 4 shows the percentage agreement between our manual labels and automatic labeling, generated from the SemEval-2012 dataset by the mapping in Tables 2 and 3.

With the relational definition of lexical entailment, we agreed on 81% of the labels. The agreement between our manual labels and the labels that were generated automatically, by applying the mapping in Tables 2 and 3 to the SemEval dataset, varied from 70% to 81%. These numbers suggest that our assumption that all word pairs within a subcategory belong to the same class is reasonable. The assumption yields levels of agreement that are comparable with the agreement in our manual labels.

We mentioned in Section 7.1 that Zhitomirsky-Geffet and Dagan (2009) had inter-annotator agreements in the 90% range, whereas our agreement is 81%. We

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

hypothesize that substitutability is a relatively objective test that leads to higher levels of agreement but excludes important cases of lexical entailment. We discussed some examples of cases that are missed by the substitutional definition in Section 2.

Table 4 shows that the agreement in our manual labels is 81% for the relational definition and 89% for the substitutional definition. This supports our hypothesis that substitutability is more objective. The agreement of 89% is close to the levels reported by Zhitomirsky-Geffet and Dagan (2009). On the other hand, the number of pairs labeled *entails* drops from 48%–51% for the relational definition to 22%–25% for the substitutional definition. This supports our hypothesis that substitutability excludes many cases of entailment. The relational definition yields approximately twice the number of lexical entailments that are captured by the substitutional definition.

As expected, the automated labeling using SemEval corresponds more closely to manual labeling with the relational definition (70%–81%) than manual labeling with the substitional definition (65%–68%). This confirms that the construction of the dataset is in accordance with the intention of our relational definition.

8 Experiments

In this section we evaluate the three approaches to lexical entailment (balAPinc, ConVecs, and SimDiffs) on the three datasets.

8.1 Experiments with the JMTH dataset

For the first set of experiments, we used the JMTH dataset (Section 7.3). This dataset has 2,308 word pairs, 1,154 in class 0 and 1,154 in class 1.

8.1.1 Experimental setup

For the experiments, we split the dataset into three (approximately) equal parts, two development sets (Dev1 and Dev2) and one test set (Test). The splits were random, except that the balance of the class sizes was maintained in all the three subsets. Both Dev1 and Dev2 contain 768 pairs and Test contains 772 pairs.

Table 5 shows the number of word pairs in the Test set for each of the ten high-level categories. In Tables 2 and 3, we see that $a \models b$ is 0 for all subcategories of the category class-inclusion, hence there are zero pairs for $a \models b$ in the row for class-inclusion in Table 5 but there are twelve pairs for $a \not\models b$. On the other hand, in Tables 2 and 3, $b \models a$ is 1 for all subcategories of the category class-inclusion, so it is not surprising to see that there are fifty-five pairs for $b \models a$ in the row for class-inclusion in Table 5 and zero pairs for $b \not\models a$. The number of pairs labeled *entails* is $261 + 125 = 386$ and the number of pairs labeled *does not entail* is $176 + 210 = 386$.

The balAPinc measure has two parameters to tune, $\max_F$ for the maximum number of features and T as a threshold for classification. On Dev1, we calculated balAPinc five times, using five different values for $\max_F$, 1,000, 2,000, 3,000, 4,000, and 5,000. For each given value of $\max_F$, we set T to the value that optimized

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 5. *The distribution of the ten high-level categories in the Test set*

ID	Category	$a \models b$	$a \not\models b$	$b \models a$	$b \not\models a$	Total
1	Class-inclusion	0	12	55	0	67
2	Part-whole	72	7	12	27	118
3	Similar	28	15	28	15	86
4	Contrast	9	25	0	26	60
5	Attribute	56	13	0	26	95
6	Non-attribute	0	40	0	31	71
7	Case relations	6	24	0	27	57
8	Cause-purpose	21	15	0	29	65
9	Space-time	35	17	10	17	79
10	Reference	34	8	20	12	74
—	Total	261	176	125	210	772

the F-measure on Dev1. This gave us five pairs of values for $\max_F$ and T. We tested each of these five settings on Dev2 and chose the setting that maximized the F-measure, which was $\max_F = 1,000$. The balAPinc measure is robust with respect to the parameter settings. The accuracy on Dev2 ranged from 56.5% with $\max_F = 1,000$ to 52.5% with $\max_F = 5,000$. We kept the best $\max_F$ setting, but we tuned T again on the union of Dev1 and Dev2. With these parameter settings, we then applied balAPinc to the Test set.

ConVecs has two parameters to tune, k and p for $\mathbf{U}_k\mathbf{\Sigma}_k^p$. For k, we tried 100, 200, 300, 400, and 500. For p, we tried ten values, from 0.1 to 1.0 in increments of 0.1. For each of the fifty pairs of values for k and p, we ran Weka, using Dev1 as training data and Dev2 as testing data. The maximum F-measure on Dev2 was achieved with $k = 100$ and $p = 0.4$. ConVecs is robust with respect to the parameter settings. The accuracy on Dev2 ranged from a high of 70.1% to a low of 64.6%. We then ran Weka for one more time, using $k = 100$ and $p = 0.4$, with the union of Dev1 and Dev2 as training data and Test as testing data.

SimDiffs has four parameters to tune, k_d and p_d for domain space and k_f and p_f for function space, but we reduced this to two parameters, k and p, by setting $k = k_\mathrm{d} = k_\mathrm{f}$ and $p = p_\mathrm{d} = p_\mathrm{f}$. We tried the same fifty pairs of values for k and p as we did with ConVecs. The maximum F-measure on Dev2 was achieved with $k = 200$ and $p = 0.6$. SimDiffs is robust with respect to the parameter settings. The accuracy on Dev2 ranged from 74.1% to 68.2%. We then ran Weka one more time, with the union of Dev1 and Dev2 as training data and Test as testing data.

8.1.2 Results

Table 6 shows the performance of all three algorithms on the Test set. According to Fisher's Exact Test (Agresti 1996), the accuracy of ConVecs (70.2%) is not significantly different from the accuracy of SimDiffs (72.4%). However, both ConVecs and SimDiffs are more accurate than balAPinc (57.3%) at the 95% confidence level. The other performance measures (AP_0, AP_1, Pre, Rec, and F) follow the same general

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 6. *Comparison of the three algorithms on the JMTH dataset*

Algorithm	AP_0	AP_1	Pre	Rec	F	Acc	95% CI
balAPinc	0.57	0.56	0.573	0.573	0.573	57.3	53.8–60.7
ConVecs	0.76	0.77	0.703	0.702	0.702	70.2	66.9–73.3
SimDiffs	0.80	0.79	0.724	0.724	0.724	72.4	69.1–75.4

Table 7. *Accuracy for each of the ten high-level categories in the Test set*

ID	Category	balAPinc	ConVecs	SimDiffs	Size
1	Class-inclusion	79.1	76.1	88.1	67
2	Part-whole	61.0	69.5	66.9	118
3	Similar	46.5	44.2	54.7	86
4	Contrast	65.0	76.7	80.0	60
5	Attribute	44.2	68.4	70.5	95
6	Non-attribute	77.5	74.6	76.1	71
7	Case relations	43.9	64.9	71.9	57
8	Cause-purpose	49.2	73.8	75.4	65
9	Space-time	68.4	83.5	77.2	79
10	Reference	40.5	75.7	73.0	74
—	Average	57.3	70.2	72.4	77

pattern as accuracy, which is what we would usually expect for a balanced dataset. The final column in Table 6 shows the 95% confidence interval (CI) for accuracy, calculated using the Wilson method.

Table 7 shows how the accuracies of the three algorithms vary over the ten high-level categories in the Test set. ConVecs and SimDiffs have roughly similar profiles but balAPinc is substantially different from the other two. This is what we would expect, given that both ConVecs and SimDiffs approach lexical entailment as a semantic relation classification problem, whereas balAPinc approaches it as a problem of designing an asymmetric similarity measure. The approach of balAPinc is near the level of the other two for some relation categories (e.g. class-inclusion, non-attribute) but substantially below for others (e.g. attribute, case relations, reference).

We explore the contribution of each set of features to the performance of SimDiffs in Table 8. In the columns for S_1 to S_4, a value of 1 indicates that the set is included in the feature vector and 0 indicates that the set is excluded (see Section 6.3). S_1 is the difference between *a* and *b* in domain space with respect to their similarities to the reference words, *R*. S_2 is the difference between *a* and *b* in function space. S_1 and S_2 are based on differences in the same spaces, whereas S_3 and S_4 are based on differences in different spaces. The parameters are tuned individually for each row in Table 8 in the same way as they are tuned for SimDiffs in Table 6. The results are based on the Test set.

Most of the differences in the accuracies in Table 8 are not significant, but according to Fisher's Exact Test at the 95% confidence level, the accuracy of all of

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 8. *Experiments with subsets of SimDiffs features*

S_1	S_2	S_3	S_4	AP_0	AP_1	Pre	Rec	F	Acc	95% CI
1	1	1	1	0.80	0.79	0.724	0.724	0.724	72.4	69.1–75.4
1	1	0	0	0.76	0.75	0.680	0.680	0.680	68.0	64.6–71.2
0	0	1	1	0.79	0.79	0.717	0.716	0.716	71.6	68.3–74.7
1	0	0	0	0.71	0.69	0.663	0.663	0.663	66.3	62.9–69.6
0	1	0	0	0.75	0.72	0.684	0.684	0.684	68.4	65.0–71.6
0	0	1	0	0.76	0.74	0.690	0.690	0.690	69.0	65.7–72.2
0	0	0	1	0.75	0.73	0.701	0.701	0.701	70.1	66.8–73.2

the features together (72.4%) is significantly higher than the accuracy of S_1 and S_2 without the help of S_3 and S_4 (68.0%). This supports the view that working with two different spaces has a synergetic effect, since each feature in S_3 and S_4 is based on two different spaces, whereas each feature in S_1 and S_2 is based on one space. (See the discussion of this in Section 6.3.)

Let *Gen* (general) refer to the matrix from Turney *et al.* (2011), and let *Dom* and *Fun* refer to the domain and function matrices from Turney (2012). In Section 6, we mentioned that we performed experiments on the development datasets (Dev1 and Dev2) in order to select the matrices for each algorithm. Based on these experiments, we chose the Gen matrix for both balAPinc and ConVecs, and chose the Dom and Fun matrices for SimDiffs.

In Table 9, we vary the matrices and evaluate the performance on the Test set to see whether the development datasets were a reliable guide for choosing the matrices. The matrices that were chosen based on the development datasets are in bold font. For balAPinc, Gen (57.3%) is indeed the best matrix. For ConVecs, it seems that Fun (71.9%) might be a better choice than Gen (70.2%), but the difference in their accuracy is not statistically significant. For SimDiffs, Dom and Fun (72.4%) are slightly less accurate than Gen and Fun (72.8%), but again the difference is not significant. As expected, no matrices are significantly better on the Test set than the matrices that were chosen based on the development datasets.

8.2 Experiments with the KDSZ dataset

The second set of experiments used the KDSZ dataset (Section 7.1). This dataset has 3,772 word pairs, 2,704 in class 0 and 1,068 in class 1.

8.2.1 Experimental setup

We experimented with four different ways of splitting the dataset. The *Evaluation* column in Table 10 indicates the experimental setup (dataset splitting).

The *standard* evaluation is ten-fold cross-validation in which the folds are random. This evaluation yields relatively high scores, because, although every pair in the KDSZ dataset is unique, many pairs share a common term. This makes supervised

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 9. *Effect of the matrices on the performance of the algorithms*

Algorithm	Matrices	AP_0	AP_1	Pre	Rec	F	Acc	95% CI
balAPinc	**Gen**	0.57	0.56	0.573	0.573	0.573	57.3	53.8–60.7
	Dom	0.53	0.54	0.532	0.532	0.532	53.2	49.7–56.7
	Fun	0.54	0.53	0.530	0.530	0.530	53.0	49.5–56.5
ConVecs	**Gen**	0.76	0.77	0.703	0.702	0.702	70.2	66.9–73.3
	Dom	0.72	0.75	0.676	0.675	0.674	67.5	64.1–70.7
	Fun	0.79	0.78	0.719	0.719	0.719	71.9	68.6–75.0
SimDiffs	**Dom, Fun**	0.80	0.79	0.724	0.724	0.724	72.4	69.1–75.4
	Gen, Fun	0.79	0.79	0.728	0.728	0.728	72.8	69.5–75.9
	Dom, Gen	0.77	0.77	0.702	0.702	0.702	70.2	66.9–73.3
	Gen, Gen	0.75	0.76	0.689	0.689	0.689	68.9	65.6–72.1

learning easier, because a pair in the testing fold will often share a term with several pairs in the training folds.

The *clustered* evaluation is designed to be more challenging than the *standard* evaluation. The *clustered* evaluation is ten-fold cross-validation with non-random folds. We put pairs that share a common term into the same fold. Due to the large number of pairs with shared terms, it is not possible to construct ten folds such that there are absolutely no terms that are shared by any two folds. Therefore, we gave a high priority to isolating the most common shared words to single folds, but we allowed a few less common shared words to appear in more than one fold. Thus, a pair in the testing fold will only rarely share a term with pairs in the training folds.

The *standard* and *clustered* evaluations have more examples in class 0 (*does not entail*) than in class 1 (*entails*). The *balanced* dataset takes the *clustered* evaluation a step further, by first clustering folds and then randomly removing pairs labeled as class 0 until all the folds have an equal number of pairs in both the classes.

For the *different* evaluation, instead of cross-validation, the algorithms are trained on the JMTH dataset and tested on the KDSZ dataset after the KDSZ dataset has been balanced by randomly removing pairs labeled as class 0.

The balAPinc measure has two parameters, $\max_F$ for the maximum number of features and T as a threshold for classification. In all the four experimental setups, we used the setting $\max_F = 1,000$, based on the tuning experiments with the JMTH dataset (Section 8.1). For T, we used the training split in each of the four experimental setups. For the *standard*, *clustered*, and *balanced* setups, the training split is the nine folds used for training in each step of the ten-fold cross-validation. For the *different* setup, the training split is the whole JMTH dataset. For all four setups, we set T to the value that optimized the F-measure on the training split.

ConVecs has two parameters to tune, k and p for $\mathbf{U}_k\mathbf{\Sigma}_k^p$. In all the four experimental setups, we used $k = 100$ and $p = 0.4$, based on the experiments with the JMTH dataset. The training splits were used to teach the supervised learning algorithm (the polynomial kernel SMO SVM in Weka).

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 10. *Comparison of the three algorithms on the KDSZ dataset*

Algorithm	Evaluation	AP_0	AP_1	Pre	Rec	F	Acc	95% CI
balAPinc	Standard	0.79	0.37	0.645	0.645	0.645	64.5	63.0–66.0
	Clustered	0.79	0.37	0.644	0.643	0.644	64.3	62.8–65.8
	Balanced	0.60	0.59	0.583	0.583	0.583	58.3	56.2–60.4
	Different	0.61	0.60	0.582	0.582	0.582	58.2	56.1–60.3
ConVecs	Standard	0.87	0.56	0.731	0.747	0.735	74.7	73.3–76.1
	Clustered	0.78	0.36	0.636	0.690	0.645	69.0	67.5–70.5
	Balanced	0.60	0.59	0.567	0.554	0.531	55.4	53.3–57.5
	Different	0.57	0.62	0.569	0.561	0.547	56.1	54.0–58.2
SimDiffs	Standard	0.88	0.60	0.749	0.757	0.752	75.7	74.3–77.0
	Clustered	0.80	0.40	0.664	0.684	0.671	68.4	66.9–69.9
	Balanced	0.63	0.64	0.596	0.592	0.588	59.2	57.1–61.3
	Different	0.58	0.61	0.581	0.574	0.564	57.4	55.3–59.5

SimDiffs has four parameters to tune. We used $k_d = k_f = 200$ and $p_d = p_f = 0.6$, based on the experiments with the JMTH dataset. The training splits were used to teach the supervised learning algorithm (the radial basis function kernel SMO SVM in Weka).

8.2.2 Results

In Table 10, the four experimental setups (standard, clustered, balanced, and different) are given in order of increasing challenge and increasing realism. Of the four experimental setups, we believe that the *different* evaluation is the most challenging and most realistic. If an RLE module is part of a commercial RTE system, the module will inevitably encounter word pairs in the field that are quite different from the pairs it saw during training. The *different* evaluation comes closest to approximating field usage.

On the *different* evaluations, balAPinc achieves an accuracy of 58.2%, ConVecs has an accuracy of 56.1%, and SimDiffs reaches 57.4% accuracy. According to Fisher's Exact Test at the 95% confidence level, there is no statistically significant difference between any of these accuracies.

With ConVecs and SimDiffs, compared with balAPinc, there is a relatively large gap between the *standard* performance and the *different* performance. This is because ConVecs and SimDiffs use supervised learning and thus they benefit from the *standard* setup, where the training data is highly similar to the testing data. In balAPinc, the training data is used only to tune the threshold, T, which limits the benefit of the training data.

Note that the gap between the *standard* performance and the *different* performance is not simply a question of the quantity of data. In the *different* setup, there is a qualitative difference between the training and testing data. Increasing the size of the training dataset with more data of the same type will not be helpful. The goal of the *different* setup is to test the ability of the algorithms to bridge the qualitative

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 11. *Comparison of the three algorithms on the BBDS dataset*

Algorithm	Evaluation	AP_0	AP_1	Pre	Rec	F	Acc	95% CI
balAPinc	Standard	0.79	0.73	0.722	0.722	0.722	72.2	70.5–73.8
	Clustered	0.79	0.73	0.722	0.722	0.722	72.2	70.5–73.8
	Different	0.79	0.73	0.701	0.687	0.682	68.7	67.0–70.4
ConVecs	Standard	0.95	0.95	0.876	0.876	0.876	87.6	86.3–88.8
	Clustered	0.92	0.91	0.829	0.821	0.819	82.1	80.6–83.5
	Different	0.72	0.71	0.652	0.651	0.650	65.1	63.3–66.9
SimDiffs	Standard	0.97	0.97	0.913	0.913	0.913	91.3	90.2–92.3
	Clustered	0.96	0.96	0.883	0.881	0.881	88.1	86.8–89.3
	Different	0.84	0.82	0.751	0.745	0.743	74.5	72.8–76.1

gap between the training and testing data. This qualitative gap is more challenging for supervised learning than a quantitative gap. It is a gap that learning algorithms inevitably face in real applications (Pan and Yang 2010).

The KDSZ dataset has been used in previous research, but the past results are not comparable with our results. Kotlerman *et al.* (2010) reported AP_1 without AP_0, but there is a tradeoff between AP_1 and AP_0. Kotlerman *et al.* (2010) did not attempt to evaluate balAPinc as a classifier, so they did not report precision, recall, F-measure, or accuracy.

8.3 Experiments with the BBDS dataset

The final set of experiments used the BBDS dataset (Section 7.2). The dataset has 2,770 word pairs, 1,385 in class 0 and 1,385 in class 1.

8.3.1 Experimental setup

We experimented with three different ways of splitting the dataset. In Table 11 the evaluations follow the same setups as in Table 10. However, there is no *balanced* setup, since the BBDS dataset is already balanced. In the *different* evaluation, the algorithms are trained on the JMTH dataset and evaluated on the BBDS dataset. This is the most realistic evaluation setup.

8.3.2 Results

In Table 11, on the *different* evaluations, balAPinc achieves an accuracy of 68.7%, ConVecs has an accuracy of 65.1%, and SimDiffs reaches 74.5% accuracy. According to Fisher's Exact Test at the 95% confidence level, all these accuracies are significantly different.

The BBDS dataset was used by Baroni *et al.* (2012) to compare balAPinc with ConVecs. They used two different evaluation setups similar to our *standard* and *different* setups. For balAPinc using a *standard* setup, they obtained an accuracy

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 12. *Summary of accuracy for the three algorithms with the three datasets*

Algorithm	JMTH accuracy	KDSZ accuracy	BBDS accuracy
balAPinc	**57.3**	58.2	**68.7**
ConVecs	70.2	56.1	**65.1**
SimDiffs	72.4	57.4	74.5

of 70.1%, slighly below our result of 72.2%. The difference is likely due to minor differences in the word–context matrices that we used. For balAPinc using a *different* setup, their accuracy was 70.4%, compared with our 68.7%. They used their own independent dataset to tune balAPinc, whereas we used the JMTH dataset. Given that our word–context matrices and our training data are different from theirs, the accuracies are closer than might be expected.[16]

For ConVecs using a *standard* setup, Baroni *et al.* (2012) report an accuracy of 88.6%, whereas we achived 87.6% accuracy. Using a *different* setup, they obtained 69.3% accuracy, whereas our accuracy was 65.1%. It seems likely that our training data (the JMTH dataset) was less similar to the BBDS dataset than their own independent dataset, which made our *different* setup more challenging than theirs. Nonetheless, the accuracies are closer than might be expected, given the differences in the setups.

9 Discussion of results

Table 12 summarizes the accuracy results from the experiments. For the KDSZ and BBDS experiments, only the *different* evaluation is shown. Bold font is used to mark the cases where the accuracy is significantly less than the accuracy of SimDiffs. In no case is the accuracy significantly greater than the accuracy of SimDiffs.

The JMTH dataset is based on seventy-nine types of semantic relations. The pairs in this dataset were labeled in accordance with the relational definition of lexical entailment (see Section 2). This explains why balAPinc, which was designed with the substitutional definition in mind, performs poorly on the JMTH dataset. ConVecs and SimDiffs were designed for semantic relation classification, so it is not surprising that they perform much better than balAPinc.

The KDSZ dataset was labeled using the substitutional definition of lexical entailment (see Section 2). On this dataset, there is no statistically significant difference between any of the algorithms. This is the ideal dataset for balAPinc, the dataset for which it was designed, so it is natural that balAPinc has the highest accuracy. On the other hand, we see that the two learning algorithms handle this dataset well, although they were trained on the JMTH dataset (recall that this is the *different* setup), which is quite different from the KDSZ dataset. It is good that

[16] These accuracy numbers and the numbers reported in the next paragraph are taken from Table 2 in Baroni *et al.* (2012).

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Table 13. *Summary of* AP_1 *for the three algorithms with the three datasets*

Algorithm	JMTH AP_1	KDSZ AP_1	BBDS AP_1
balAPinc	0.56	0.37	0.73
ConVecs	0.77	0.36	0.91
SimDiffs	0.79	0.40	0.96

they both are able to cope with the qualitative difference between the training data and the testing data.

All of the positive pairs in the BBDS dataset are instances of the hyponym–hypernym semantic relation. Instances of this relation are substitutable, so balAPinc is designed to handle them. ConVecs was also designed specifically for this dataset, and we see from Table 11 that ConVecs reaches an accuracy of 87.6% when the training data are similar to the testing data. However, ConVecs has trouble bridging the qualitative gap between the training data (the JMTH dataset) and the testing data with the *different* setup. On the other hand, SimDiffs is able to bridge this gap.

We have argued that the *different* evaluation is the most realistic scenario, but it could be argued that the *entails* class is more important than the *does not entail class*, and *entails* is also more scarce in natural settings. Therefore, Table 13 presents an alternative summary of the results. The table reports AP_1 instead of accuracy; this puts the emphasis on the *entails* class. For the KDSZ and BBDS datasets, we report the *clustered* setup. This is closer to the evaluation setup of Kotlerman *et al.* (2010). In Table 13, we do not use bold font to mark significant differences because there is no agreement on the appropriate statistical test for AP_1.

Although Tables 12 and 13 are based on different scores and experimental setups, both support SimDiffs and the similarity differences hypothesis. More generally, they suggest that second-order features are useful for modeling lexical entailment. They also suggest that it is beneficial to use two different spaces when constructing features for lexical entailment.

Manually designing an asymmetric similarity measure is a difficult task, as we can see from the equations in Section 6.1. We believe that lexical entailment is more tractable when it is approached as a supervised learning problem. The effort involved in manually designing feature vectors is less than that required for designing similarity measures. The performance of SimDiffs indicates that supervised learning can yield better results than manually designing measures.

10 Limitations and future work

We have evaluated RLE directly, but most applications would use RLE as a module inside a larger system. Future work will be needed to demonstrate that our results with a direct evaluation can predict how an RLE module will perform as a component of a larger system.

Although SimDiffs performs better than the competition, there is much room for improved performance. However, when SimDiffs is used as a component in

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

a larger RTE system, words will be given in the contexts of sentences. With the support of this contextual information and help from other modules in the system, SimDiffs might yield substantial improvements in RTE performance. Related to this proposed future work, Shnarch, Barak and Dagan (2009) evaluated *lexical reference* rules (Glickman, Dagan, and Shnarch 2006) derived from Wikipedia on the RTE-4 dataset. Used as a component in an RTE system, the rules improved the RTE-4 score by 1%.

Most of the past work on RLE has been based on the context inclusion hypothesis, but ConVecs and SimDiffs show that other approaches, based on novel hypotheses, can achieve competitive results. We believe that progress on the problem will come from exploring a wide range of hypotheses and approaches. It is too early at this stage of research to commit the field to a single hypothesis.

Recall the semantic relation subcategories hypothesis. Lexical entailment is not a superset of high-level categories of semantic relations, but it is a superset of lower level subcategories of semantic relations. The experiments lend some support to this hypothesis, but more research is needed. Any counterexamples for the hypothesis could be handled by revising the taxonomy. However, if the required revisions become onerous, then the hypothesis should be rejected.

The three algorithms here are based on three different hypotheses, but all three achieve some degree of success on the task of RLE. This suggests that it would be fruitful to combine the three approaches. One simple way to combine them would be to average their real-valued outputs or apply voting to their binary-valued outputs. This could be a useful direction for the future research.

We have focused here on individual words, but the natural next step is to extend these ideas to phrases. Baroni *et al.* (2012) have achieved promising results with quantifier phrases, such as *all dogs* $\models$ *some dogs*.

Looking at Tables 2 and 3 in Section 7.3, we see a high density of 1s (*entails*) for class-inclusion and part-whole. The strong connection between these two categories and lexical entailment may explain why Morris and Hirst (2004) call hypernymy and meronymy *classical relations*, whereas the relation in *chapel:funeral* (space-time, location:activity, ID 9c in the semantic relation taxonomy) is *non-classical* (this is one of their examples of a *non-classical* relation). For instance, WordNet contains information about hypernymy and meronymy, but not space-time relations. Particular relations might be considered *classical* because we find them particularly useful for making inferences. This connection is another topic for the future work.

11 Conclusions

In this paper, we have evaluated three different algorithms for RLE on three different datasets. Each algorithm relies on a different hypothesis about lexical entailment. We find that SimDiffs has the best performance on two of the three datasets. On the third dataset, there is no significant difference in the three algorithms. The performance of SimDiffs suggests that similarity differences make useful features for learning to recognize lexical entailment.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

We have approached lexical entailment as a supervised learning problem of semantic relation classification. The results indicate that this is a promising approach to lexical entailment. This builds a bridge between research in lexical entailment and research in semantic relation classification. We hope that this connection will strengthen research in both the fields.

Acknowledgments

Thanks to Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet for providing a copy of the KDSZ dataset and answering questions. Thanks to Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan for providing the BBDS dataset and answering questions. Thanks to the reviewers of *Natural Language Engineering* for their very helpful comments.

References

Agresti, A. 1996. *An Introduction to Categorical Data Analysis*. New York, NY: Wiley.

Akhmatova, E., and Dras, M. 2009. Using hypernymy acquisition to tackle (part of) textual entailment. In *Proceedings of the 2009 Workshop on Applied Textual Inference at ACL-IJCNLP 2009*, Suntec, Singapore, pp. 52–60.

Androutsopoulos, I., and Malakasiotis, P. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research* **38**: 135–87.

Baroni, M., Bernardi, R., Do, N.-Q., and Shan, C. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, Avignon, France, pp. 23–32.

Bejar, I. I., Chaffin, R., and Embretson, S. E. 1991. *Cognitive and Psychometric Analysis of Analogical Problem Solving*. New York, NY: Springer-Verlag.

Buckley, C., and Voorhees, E. 2000. Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, pp. 33–40. New York, NY: ACM.

Bullinaria, J., and Levy, J. 2007. Extracting semantic representations from word co-occurrence statistics: a computational study. *Behavior Research Methods* **39**(3): 510–26.

Bullinaria, J., and Levy, J. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behavior Research Methods* **44**, 890–907.

Büttcher, S., and Clarke, C. 2005. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD.

Caron, J. 2001. Experiments with LSA scoring: optimal rank and basis. In *Proceedings of the SIAM Computational Information Retrieval Workshop*, Raleigh, NC, pp. 157–69.

Dagan, I., Dolan, B., Magnini, B., and Roth, D. 2009. Recognizing textual entailment: rational, evaluation and approaches. *Natural Language Engineering* **15**(4): i–xvii.

Dagan, I., Glickman, O., and Magnini, B. 2006. The PASCAL recognising textual entailment challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché-Buc (eds.), *Machine Learning Challenges: Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pp. 177–90, New York, NY: Springer.

Do, Q. X., and Roth, D. 2010. Constraints-based taxonomic relation classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, MIT, Cambridge, MA, pp. 1099–109.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Do, Q. X., and Roth, D. 2012. Exploiting the Wikipedia structure in local and global classification of taxonomic relations. *Natural Language Engineering* **18**(2): 235–62.

Firth, J. R. 1957. A synopsis of linguistic theory 1930–1955. In F. Palmer (ed.), *Studies in Linguistic Analysis*, pp. 1–32. Oxford, UK: Blackwell.

Geffet, M., and Dagan, I. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005)*, Ann Arbor, MI, pp. 107–14.

Girju, R., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., and Yuret, D. 2007. SemEval-2007 Task 4: classification of semantic relations between nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval 2007)*, Prague, Czech Republic, pp. 13–8.

Glickman, O., Dagan, I., and Shnarch, E. 2006. Lexical reference: a semantic matching subtask. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia, pp. 172–9.

Golub, G. H., and Van Loan, C. F. 1996. *Matrix Computations*, 3rd edn. Baltimore, MD: Johns Hopkins University Press.

Harris, Z. 1954. Distributional structure. *Word* **10**(23): 146–62.

Hearst, M. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING-92)*, Nantes, France, pp. 539–45.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Séaghdha, D. O., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. 2010. Semeval-2010 task 8: multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, Uppsala, Sweden, pp. 33–8.

Herrera, J., Peñas, A., and Verdejo, F. 2006. Textual entailment recognition based on dependency analysis and WordNet. In *Machine Learning Challenges: Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, Lecture Notes in Computer Science, vol. 3944, pp. 231–9. New York, NY: Springer.

Hickl, A., Bensley, J., Williams, J., Roberts, K., Rink, B., and Shi, Y. 2006. Recognizing textual entailment with LCC's GROUNDHOG system. In *Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment*, Venice, Italy.

Hunter, G. 1996. *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*. Berkeley, CA: University of California Press.

Jurgens, D. A., Mohammad, S. M., Turney, P. D., and Holyoak, K. J. 2012. SemEval-2012 Task 2: measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, Montréal, Canada, pp. 356–64.

Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-Geffet, M. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* **16**(4): 359–89.

Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W. 2007. *Handbook of Latent Semantic Analysis*. Mahwah, NJ: Lawrence Erlbaum.

Lee, L. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, College Park, MD, pp. 25–32.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada, pp. 768–74. Ann Arbor, MI: Association for Computational Linguistics.

Lin, D., and Pantel, P. 2001. DIRT – discovery of inference rules from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, San Francisco, CA, pp. 323–8.

Manning, C., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Mirkin, S., Bar-Haim, R., Berant, J., Dagan, I., Shnarch, E., Stern, A., and Szpektor, I. 2009a. Bar-ilan University's submission to RTE-5. In *Proceedings of the Second Text Analysis Conference (TAC 2009)*, Gaithersburg, MD.

Mirkin, S., Dagan, I., and Shnarch, E. 2009b. Evaluating the inferential utility of lexical-semantic resources. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, Athens, Greece, pp. 558–66.

Morris, J., and Hirst, G. 2004. Non-classical lexical semantic relations. In *Workshop on Computational Lexical Semantics, HLT-NAACL-04*, Boston, MA.

Nastase, V., and Szpakowicz, S. 2003. Exploring noun-modifier semantic relations. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, Tilburg, Netherlands, pp. 285–301.

Ogden, C. K. 1930. *Basic English: A General Introduction with Rules and Grammar*. London: Kegan Paul, Trench, Trubner.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* **22**: 1345–59.

Pedersen, T., Patwardhan, S., and Michelizzi, J. 2004. WordNet::Similarity – measuring the relatedness of concepts. In D. Palmer, J. Polifroni, and D. Roy (eds.), *Demonstration Papers at HLT-NAACL 2004*, Boston, MA, pp. 38–41.

Platt, J. C. 1998. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola (eds.), *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208, Cambridge, MA: MIT Press.

Rosario, B., and Hearst, M. 2001. Classifying the semantic relations in noun-compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*, Pittsburgh, PA, pp. 82–90.

Rosario, B., Hearst, M., and Fillmore, C. 2002. The descent of hierarchy, and selection in relational semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA, pp. 247–54.

Salton, G., and McGill, M. 1983. *Introduction to Modern Information Retrieval*. New York, NY: McGraw-Hill.

Shnarch, E., Barak, L., and Dagan, I. 2009. Extracting lexical reference rules from Wikipedia. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, Suntec, Singapore, pp. 450–8.

Snow, R., Jurafsky, D., and Ng, A. Y. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, Sydney, NSW, Australia, pp. 801–8.

Szpektor, I., and Dagan, I. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, Manchester, UK, pp. 849–56.

Turney, P. D. 2006. Similarity of semantic relations. *Computational Linguistics* **32**(3): 379–416.

Turney, P. D. 2012. Domain and function: a dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research* **44**: 533–85.

Turney, P. D., Neuman, Y., Assaf, D., and Cohen, Y. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, UK, pp. 680–90.

Turney, P. D., and Pantel, P. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research* **37**: 141–88.

Weeds, J., and Weir, D. 2003. A general framework for distributional similarity. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2003)*, Sapporo, Japan, pp. 81–8.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press

Weeds, J., Weir, D., and McCarthy, D. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, Geneva, Switzerland, pp. 1015–21.

Witten, I. H., Frank, E., and Hall, M. A. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd edn. San Francisco, CA: Morgan Kaufmann.

Zhitomirsky-Geffet, M., and Dagan, I. 2009. Bootstrapping distributional feature vector quality. *Computational Linguistics* **35**(3): 435–61.

https://doi.org/10.1017/S1351324913000387 Published online by Cambridge University Press