Analogy perception applied to seven tests of word comprehension
Turney, Peter D

Publisher's version / Version de l'éditeur:
https://doi.org/10.1080/0952813X.2010.502311
Journal of Experimental & Theoretical Artificial Intelligence, 23, 3, pp. 343-362, 2011-07-01

Analogy Perception Applied to Seven Tests of Word Comprehension

Peter D. Turney
Institute for Information Technology
National Research Council of Canada
M-50 Montreal Road
Ottawa, Ontario, Canada
K1A 0R6

Phone: (613) 993-8564
Fax: (613) 952-7151
peter.turney@nrc-cnrc.gc.ca

February 23, 2009

Analogy Perception Applied to Seven Tests of Word Comprehension

Abstract

It has been argued that analogy is the core of cognition. In AI research, algorithms for analogy are often limited by the need for hand-coded high-level representations as input. An alternative approach is to use high-level perception, in which high-level representations are automatically generated from raw data. Analogy perception is the process of recognizing analogies using high-level perception. We present PairClass, an algorithm for analogy perception that recognizes lexical proportional analogies using representations that are automatically generated from a large corpus of raw textual data. A proportional analogy is an analogy of the form A:B::C:D, meaning "A is to B as C is to D". A lexical proportional analogy is a proportional analogy with words, such as carpenter:wood::mason:stone. PairClass represents the semantic relations between two words using a high-dimensional feature vector, in which the elements are based on frequencies of patterns in the corpus. PairClass recognizes analogies by applying standard supervised machine learning techniques to the feature vectors. We show how seven different tests of word comprehension can be framed as problems of analogy perception and we then apply PairClass to the seven resulting sets of analogy perception problems. We achieve competitive results on all seven tests. This is the first time a uniform approach has handled such a range of tests of word comprehension.

Keywords: analogies, word comprehension, test-based AI, semantic relations, synonyms, antonyms.

1 Introduction

Many AI researchers and cognitive scientists believe that analogy is "the core of cognition" (Hofstadter, 2001):

- "How do we ever understand anything? Almost always, I think, by using one or another kind of analogy." – Marvin Minsky (1986)
- "My thesis is this: what makes humans smart is (1) our exceptional ability to learn by analogy, (2) the possession of symbol systems such as language and mathematics, and (3) a relation of mutual causation between them

whereby our analogical prowess is multiplied by the possession of relational language." – Dedre Gentner (2003)

- "We have repeatedly seen how analogies and mappings give rise to secondary meanings that ride on the backs of primary meanings. We have seen that even primary meanings depend on unspoken mappings, and so in the end, we have seen that all meaning is mapping-mediated, which is to say, all meaning comes from analogies." – Douglas Hofstadter (2007)

These quotes connect analogy with understanding, learning, language, and meaning. Our research in natural language processing for word comprehension (lexical semantics) has been guided by this view of the importance of analogy.

The best-known approach to analogy-making is the Structure-Mapping Engine (SME) (Falkenhainer *et al.*, 1989), which is able to process scientific analogies. SME constructs a mapping between two high-level conceptual representations. These kinds of high-level analogies are sometimes called *conceptual analogies*. For example, SME is able to build a mapping between a high-level representation of Rutherford's model of the atom and a high-level representation of the solar system (Falkenhainer *et al.*, 1989). The input to SME consists of hand-coded high-level representations, written in LISP. (See Appendix B of Falkenhainer *et al.* (1989) for examples of the input LISP code.)

The SME approach to analogy-making has been criticized because it assumes that hand-coded representations are available as the basic building blocks for analogy-making (Chalmers *et al.*, 1992). The process of forming high-level conceptual representations from raw data (without hand-coding) is called *high-level perception* (Chalmers *et al.*, 1992). Turney (2008a) introduced the Latent Relation Mapping Engine (LRME), which combines ideas from SME and Latent Relational Analysis (LRA) (Turney, 2006). LRME is able to construct mappings without hand-coded high-level representations. Using a kind of high-level perception, LRME builds conceptual representations from raw data, consisting of a large corpus of plain text, gathered by a web crawler.

In this paper, we use ideas from LRA and LRME to solve word comprehension tests. We focus on a kind of lower-level analogy, called *proportional analogy*, which has the form $A{:}B{::}C{:}D$, meaning "A is to B as C is to D". Each component mapping in a high-level conceptual analogy is essentially a lower-level proportional analogy. For example, in the analogy between the solar system and Rutherford's model of the atom, the component mappings include the proportional analogies sun:planet::nucleus:electron and mass:sun::charge:nucleus (Turney, 2008a).

Proportional analogies are common in psychometric tests, such as the Miller Analogies Test (MAT) and the Graduate Record Examination (GRE). In these tests, the items in the analogies are usually either geometric figures or words. An early AI system for proportional analogies with geometric figures was ANALOGY (Evans, 1964) and an early system for words was Argus (Reitman, 1965). Both of these systems used hand-coded representations to solve simple proportional analogy questions.

In Section 2, we present an algorithm we call PairClass, designed for recognizing proportional analogies with words. PairClass performs high-level perception (Chalmers *et al.*, 1992), forming conceptual representations of semantic relations between words, by analysis of raw textual data, without hand-coding. The representations are high-dimensional vectors, in which the values of the elements are derived from the frequencies of patterns in textual data. This form of representation is similar to latent semantic analysis (LSA) (Landauer and Dumais, 1997), but vectors in LSA represent the meaning of individual words, whereas vectors in PairClass represent the relations between two words. The use of frequency vectors to represent semantic relations was introduced in Turney *et al.* (2003).

PairClass uses a standard supervised machine learning algorithm (Platt, 1998; Witten and Frank, 1999) to classify word pairs according to their semantic relations. A proportional analogy such as sun:planet::nucleus:electron asserts that the semantic relations between sun and planet are similar to the semantic relations between nucleus and electron. The planet orbits the sun; the electron orbits the nucleus. The sun's gravity attracts the planet; the nucleus's charge attracts the electron. The task of perceiving this proportional analogy can be framed as the task of learning to classify sun:planet and nucleus:electron into the same class, which we might call orbited:orbiter. Thus our approach to analogy perception is to frame it as a problem of classification of word pairs (hence the name PairClass).

To evaluate PairClass, we use seven word comprehension tests. This could be seen as a return to the 1960's psychometric test-based approach of ANALOGY (Evans, 1964) and Argus (Reitman, 1965), but the difference is that PairClass achieves human-level scores on the tests without using hand-coded representations. We believe that word comprehension tests serve as an excellent benchmark for evaluating progress in computational linguistics. More generally, we support test-based AI research (Bringsjord and Schimanski, 2003).

In Section 3, we present our experiments with seven tests:

- 374 multiple-choice analogy questions from the SAT college entrance test (Turney *et al.*, 2003),
- 80 multiple-choice synonym questions from the TOEFL (test of English as a foreign language) (Landauer and Dumais, 1997),
- 50 multiple-choice synonym questions from an ESL (English as a second language) practice test (Turney, 2001),

- 136 synonym-antonym questions collected from several ESL practice tests (introduced here),
- 160 synonym-antonym questions from research in computational linguistics (Lin *et al.*, 2003),
- 144 similar-associated-both questions that were used for research in cognitive psychology (Chiarello *et al.*, 1990), and
- 600 noun-modifier relation classification problems from research in computational linguistics (Nastase and Szpakowicz, 2003).

We discuss the results of the experiments in Section 4. For five of the seven tests, there are past results that we can compare with the performance of PairClass. In general, PairClass is competitive, but not the best system. However, the strength of PairClass is that it is able to handle seven different tests. As far as we know, no other system can handle this range of tests. PairClass performs well, although it is competing against specialized algorithms, developed for single tasks. We believe that this illustrates the power of analogy perception as a unified approach to lexical semantics.

Related work is examined in Section 5. PairClass is similar to past work on semantic relation classification (Rosario and Hearst, 2001; Nastase and Szpakowicz, 2003; Turney and Littman, 2005; Girju *et al.*, 2007). For example, with noun-modifier classification, the task is to classify a noun-modifier pair, such as *laser printer*, according to the semantic relation between the head noun, *printer*, and the modifier, *laser*. In this case, the relation is *instrument:agency*: the laser is an instrument that is used by the printer. The standard approach to semantic relation classification is to use supervised machine learning techniques to classify feature vectors that represent relations. We demonstrate in this paper that the paradigm of semantic relation classification can be extended beyond the usual relations, such as *instrument:agency*, to include analogy, synonymy, antonymy, similarity, and association.

Limitations and future work are considered in Section 6. Limitations of PairClass are the need for a large corpus and the time required to run the algorithm. We conclude in Section 7.

PairClass was briefly introduced in Turney (2008b). The current paper describes PairClass in more detail, provides more background information and discussion, and brings the number of tests up from four to seven.

2 Analogy Perception

A lexical analogy, $A{:}B{::}C{:}D$, asserts that A is to B as C is to D; for example, carpenter:wood::mason:stone asserts that carpenter is to wood as mason is to stone;

that is, the semantic relations between carpenter and wood are highly similar to the semantic relations between mason and stone. In this paper, we frame the task of recognizing lexical analogies as a problem of classifying word pairs (see Table 1).

Word pair	Class label
carpenter:wood	artisan:material
mason:stone	artisan:material
potter:clay	artisan:material
glassblower:glass	artisan:material
sun:planet	orbited:orbiter
nucleus:electron	orbited:orbiter
earth:moon	orbited:orbiter
starlet:paparazzo	orbited:orbiter

Table 1: Examples of how the task of recognizing lexical analogies may be viewed as a problem of classifying word pairs.

We approach this task as a standard classification problem for supervised machine learning (Witten and Frank, 1999). PairClass takes as input a training set of word pairs with class labels and a testing set of word pairs without labels. Each word pair is represented as a vector in a feature space and a supervised learning algorithm is used to classify the feature vectors. The elements in the feature vectors are based on the frequencies of automatically defined patterns in a large corpus. The output of the algorithm is an assignment of labels to the word pairs in the testing set. For some of the following experiments, we select a unique label for each word pair; for other experiments, we assign probabilities to each possible label for each word pair.

For a given word pair, such as mason:stone, the first step is to generate morphological variations, such as masons:stones. In the following experiments, we use *morpha* (morphological analyzer) and *morphg* (morphological generator) for morphological processing (Minnen *et al.*, 2001).[1]

The second step is to search in a large corpus for phrases of the following forms:

- "[0 to 1 words] X [0 to 3 words] Y [0 to 1 words]"
- "[0 to 1 words] Y [0 to 3 words] X [0 to 1 words]"

In these templates, X:Y consists of morphological variations of the given word pair; for example, mason:stone, mason:stones, masons:stones, and so on. Typical phrases for mason:stone would be "the mason cut the stone with" and "the stones

[1] http://www.informatics.susx.ac.uk/research/groups/nlp/carroll/morph.html.

that the mason used". We then normalize all of the phrases that are found, by using *morpha* to remove suffixes.

The templates we use here are similar to those in Turney (2006), but we have added extra context words before the first variable (X in the first template and Y in the second) and after the second variable. Our morphological processing also differs from Turney (2006). In the following experiments, we search in a corpus of 5×10^{10} words (about 280 GB of plain text), consisting of web pages gathered by a web crawler.[2] To retrieve phrases from the corpus, we use Wumpus (Büttcher and Clarke, 2005), an efficient search engine for passage retrieval from large corpora.[3]

The next step is to generate patterns from all of the phrases that were found for all of the input word pairs (from both the training and testing sets). To generate patterns from a phrase, we replace the given word pairs with variables, X and Y, and we replace the remaining words with a wild card symbol (an asterisk) or leave them as they are. For example, the phrase "the mason cut the stone with" yields the patterns "the X cut * Y with", "* X * the Y *", and so on. If a phrase contains n words, then it yields $2^{(n-2)}$ patterns.

Each pattern corresponds to a feature in the feature vectors that we will generate. Since a typical input set of word pairs yields millions of patterns, we need to use feature selection, to reduce the number of patterns to a manageable quantity. For each pattern, we count the number of input word pairs that generated the pattern. For example, "* X cut * Y *" is generated by both mason:stone and carpenter:wood. We then sort the patterns in descending order of the number of word pairs that generated them. If there are N input word pairs (and thus N feature vectors, including both the training and testing sets), then we select the top kN patterns and drop the remainder. In the following experiments, k is set to 20. The algorithm is not sensitive to the precise value of k.

The reasoning behind the feature selection algorithm is that shared patterns make more useful features than rare patterns. The number of features (kN) depends on the number of word pairs (N), because, if we have more feature vectors, then we need more features to distinguish them. Turney (2006) also selects patterns based on the number of pairs that generate them, but the number of selected patterns is a constant (8000), independent of the number of input word pairs.

The next step is to generate feature vectors, one vector for each input word pair. Each of the N feature vectors has kN elements, one element for each selected pattern. The value of an element in a vector is given by the logarithm of the frequency in the corpus of the corresponding pattern for the given word pair. For

[2]The corpus was collected by Charles Clarke at the University of Waterloo. We can provide copies of the corpus on request.

[3]http://www.wumpus-search.org/.

example, suppose the given pair is mason:stone and the pattern is "* X cut * Y *". We look at the normalized phrases that we collected for mason:stone and we count how many match this pattern. If f phrases match the pattern, then the value of this element in the feature vector is $\log(f+1)$ (we add 1 because $\log(0)$ is undefined). Each feature vector is then normalized to unit length. The normalization ensures that features in vectors for high-frequency word pairs are comparable to features in vectors for low-frequency word pairs.

Table 2 shows the first and last ten features (excluding zero-valued features) and the corresponding feature values for the word pair audacious:boldness, taken from the SAT analogy questions. The features are in descending order of the number of word pairs that generate them; that is, they are ordered from common to rare. Thus the first features typically involve patterns with many wild cards and high-frequency words, and the first feature values are usually nonzero. The last features often have few wild cards and contain low-frequency words, with feature values that are usually zero. The feature vectors are generally highly sparse (i.e., they are mainly zeros; if $f = 0$, then $\log(f+1) = 0$).

Now that we have a feature vector for each input word pair, we can apply a standard supervised learning algorithm. In the following experiments, we use a sequential minimal optimization (SMO) support vector machine (SVM) with a radial basis function (RBF) kernel (Platt, 1998), as implemented in Weka (Waikato Environment for Knowledge Analysis) (Witten and Frank, 1999).[4] The algorithm generates probability estimates for each class by fitting logistic regression models to the outputs of the SVM. We disable the normalization option in Weka, since the vectors are already normalized to unit length. We chose the SMO RBF algorithm because it is fast, robust, and it easily handles large numbers of features.

In the following experiments, PairClass is applied to each of the seven tests with no adjustments or tuning of the learning parameters to the specific problems. Some work is required to fit each problem into the general framework of PairClass (analogy perception: supervised classification of word pairs), but the core algorithm is the same in each case.

It might be objected that what PairClass does should not be considered as high-level perception, in the sense given by Chalmers *et al.*. (1992). They define high-level perception as follows:

> Perceptual processes form a spectrum, which for convenience we can divide into two components. ... [We] have low-level perception, which involves the early processing of information from the various sensory modalities. High-level perception, on the other hand, involves taking

[4] http://www.cs.waikato.ac.nz/ml/weka/.

Feature number	Feature (pattern)	Value (normalized log)
1	"* X * * Y *"	0.090
2	"* Y * * X *"	0.150
3	"* X * Y *"	0.198
4	"* Y * X *"	0.221
5	"* X * * * Y *"	0.045
7	"* X Y *"	0.233
8	"* Y X *"	0.167
10	"* Y * the X *"	0.071
12	"* Y and * X *"	0.116
13	"* X and Y *"	0.135
27,591	"define X * Y *"	0.045
28,524	"what Y and X *"	0.045
28,804	"for Y and * X and"	0.045
29,017	"very X and Y *"	0.045
32,028	"s Y and X and"	0.045
34,893	"understand X * Y *"	0.071
35,027	"* X be not * Y but"	0.045
39,410	"* Y and X cause"	0.045
41,303	"* X but Y and"	0.105
43,511	"be X not Y *"	0.105

Table 2: The first and last ten features, excluding zero-valued features, for the pair X:Y = audacious:boldness. (The "s" in the pattern for feature 32,028 is part of a possessive noun. The "be" in the patterns for features 35,027 and 43,511 is the result of normalizing "is" and "was" with *morpha*.)

> a more global view of this information, extracting *meaning* from the raw material by accessing concepts, and making sense of situations at a conceptual level. This ranges from the recognition of objects to the grasping of abstract relations, and on to understanding entire situations as coherent wholes. ... The study of high-level perception leads us directly to the problem of mental *representation*. Representations are the fruits of perception.

Spoken or written language can be converted to electronic text by speech recognition software or optical character recognition software. It seems reasonable to call this low-level perception. PairClass takes electronic text as input and generates high-dimensional feature vectors from the text. These feature vectors represent abstract semantic relations and they can be used to classify semantic relations into various semantic classes. It seems reasonable to call this high-level perception. We do not claim that PairClass has the richness and complexity of human high-level perception, but it is nonetheless a (simple, restricted) form of high-level perception.

3 Experiments

This section presents seven sets of experiments. We explain how each of the seven tests is treated as a problem of analogy perception, we give the experimental results, and we discuss past work with each test.

3.1 SAT Analogies

In this section, we apply PairClass to the task of recognizing lexical analogies. To evaluate the performance, we use a set of 374 multiple-choice questions from the SAT college entrance exam. Table 3 shows a typical question. The target pair is called the *stem*. The task is to select the choice pair that is most analogous to the stem pair.

Stem:		mason:stone
Choices:	(a)	teacher:chalk
	(b)	carpenter:wood
	(c)	soldier:gun
	(d)	photograph:camera
	(e)	book:word
Solution:	(b)	carpenter:wood

Table 3: An example of a question from the 374 SAT analogy questions.

The problem of recognizing lexical analogies was first attempted with a system called Argus (Reitman, 1965), using a small hand-built semantic network with a spreading activation algorithm. Turney *et al.* (2003) used a combination of 13 independent modules. Veale (2004) used a spreading activation algorithm with WordNet (in effect, treating WordNet as a semantic network). Turney (2005) used a corpus-based algorithm.

We may view Table 3 as a binary classification problem, in which mason:stone and carpenter:wood are positive examples and the remaining word pairs are negative examples. The difficulty is that the labels of the choice pairs must be hidden from the learning algorithm. That is, the training set consists of one positive example (the stem pair) and the testing set consists of five unlabeled examples (the five choice pairs). To make this task more tractable, we randomly choose a stem pair from one of the 373 other SAT analogy questions, and we assume that this new stem pair is a negative example, as shown in Table 4.

Word pair	Train or test	Class label
mason:stone	train	positive
tutor:pupil	train	negative
teacher:chalk	test	hidden
carpenter:wood	test	hidden
soldier:gun	test	hidden
photograph:camera	test	hidden
book:word	test	hidden

Table 4: How to fit a SAT analogy question into the framework of supervised classification of word pairs. The randomly chosen stem pair is tutor:pupil.

To answer a SAT question, we use PairClass to estimate the probability that each testing example is positive, and we guess the testing example with the highest probability. Learning from a training set with only one positive example and one negative example is difficult, since the learned model can be highly unstable. To increase the stability, we repeat the learning process 10 times, using a different randomly chosen negative training example each time. For each testing word pair, the 10 probability estimates are averaged together. This is a form of bagging (Breiman, 1996). Table 5 shows an example of an analogy that has been correctly solved by PairClass.

Stem:		insubordination:punishment	Probability
Choices:	(a)	evening:night	0.236
	(b)	earthquake:tornado	0.260
	(c)	candor:falsehood	0.391
	(d)	heroism:praise	0.757
	(e)	fine:penalty	0.265
Solution:	(d)	heroism:praise	0.757

Table 5: An example of a correctly solved SAT analogy question.

PairClass attains an accuracy of 52.1% on the 374 SAT analogy questions. The best previous result is an accuracy of 56.1% (Turney, 2005). Random guessing would yield an accuracy of 20% (five choices per question). The average senior high school student achieves 57% correct (Turney, 2006). The *ACL Wiki* lists 12 previously published results with the 374 SAT analogy questions.[5] Adding PairClass to the list, we have 13 results. PairClass has the third highest accuracy of the 13 systems.

3.2 TOEFL Synonyms

Now we apply PairClass to the task of recognizing synonyms, using a set of 80 multiple-choice synonym questions from the TOEFL (test of English as a foreign language). A sample question is shown in Table 6. The task is to select the choice word that is most similar in meaning to the stem word.

Stem:		levied
Choices:	(a)	imposed
	(b)	believed
	(c)	requested
	(d)	correlated
Solution:	(a)	imposed

Table 6: An example of a question from the 80 TOEFL synonym questions.

Synonymy can be viewed as a high degree of semantic similarity. The most common way to measure semantic similarity is to measure the distance between words in WordNet (Resnik, 1995; Jiang and Conrath, 1997; Hirst and St-Onge, 1998; Budanitsky and Hirst, 2001). Corpus-based measures of word similarity are also common (Lesk, 1969; Landauer and Dumais, 1997; Turney, 2001).

We may view Table 6 as a binary classification problem, in which the pair levied:imposed is a positive example of the class *synonymous* and the other possible pairings are negative examples, as shown in Table 7.

Word pair	Class label
levied:imposed	positive
levied:believed	negative
levied:requested	negative
levied:correlated	negative

Table 7: How to fit a TOEFL synonym question into the framework of supervised classification of word pairs.

[5]For more information, see *SAT Analogy Questions (State of the art)* at http://aclweb.org/aclwiki/. There were 12 previous results at the time of writing, but the list is likely to grow.

The 80 TOEFL questions yield 320 (80×4) word pairs, 80 labeled positive and 240 labeled negative. We apply PairClass to the word pairs using ten-fold cross-validation. In each random fold, 90% of the pairs are used for training and 10% are used for testing. For each fold, we use the learned model to assign probabilities to the testing pairs. Our guess for each TOEFL question is the choice that has the highest probability of being positive, when paired with the corresponding stem. Table 8 gives an example of a correctly solved question.

Stem:		prominent	Probability
Choices:	(a)	battered	0.005
	(b)	ancient	0.114
	(c)	mysterious	0.010
	(d)	conspicuous	0.998
Solution:	(d)	conspicuous	0.998

Table 8: An example of a correctly solved TOEFL synonym question.

PairClass attains an accuracy of 76.2%. For comparison, the *ACL Wiki* lists 15 previously published results with the 80 TOEFL synonym questions.[6] Adding PairClass to the list, we have 16 algorithms. PairClass has the ninth highest accuracy of the 16 systems. The best previous result is an accuracy of 97.5% (Turney *et al.*, 2003), obtained using a hybrid of four different algorithms. Random guessing would yield an accuracy of 25% (four choices per question). The average foreign applicant to a US university achieves 64.5% correct (Landauer and Dumais, 1997).

3.3 ESL Synonyms

The 50 ESL synonym questions are similar to the TOEFL synonym questions, except that each question includes a sentence that shows the stem word in context. Table 9 gives an example. In our experiments, we ignore the sentence context and treat the ESL synonym questions the same way as we treated the TOEFL synonym questions (see Table 10).

The 50 ESL questions yield 200 (50×4) word pairs, 50 labeled positive and 150 labeled negative. We apply PairClass to the word pairs using ten-fold cross-validation. Our guess for each question is the choice word that has the highest probability of being positive, when paired with the corresponding stem word.

PairClass attains an accuracy of 78.0%. The best previous result is 82.0% (Jarmasz and Szpakowicz, 2003). The *ACL Wiki* lists 8 previously published results

[6] See *TOEFL Synonym Questions (State of the art)* at http://aclweb.org/aclwiki/. There were 15 systems at the time of writing, but the list is likely to grow.

Stem:		"A *rusty* nail is not as strong as a clean, new one."
Choices:	(a)	corroded
	(b)	black
	(c)	dirty
	(d)	painted
Solution:	(a)	corroded

Table 9: An example of a question from the 50 ESL synonym questions.

Word pair	Class label
rusty:corroded	positive
rusty:black	negative
rusty:dirty	negative
rusty:painted	negative

Table 10: How to fit an ESL synonym question into the framework of supervised classification of word pairs.

for the 50 ESL synonym questions.[7] Adding PairClass to the list, we have 9 algorithms. PairClass has the third highest accuracy of the 9 systems. The average human score is unknown. Random guessing would yield an accuracy of 25% (four choices per question).

3.4 ESL Synonyms and Antonyms

The task of classifying word pairs as either synonyms or antonyms readily fits into the framework of supervised classification of word pairs. Table 11 shows some examples from a set of 136 ESL (English as a second language) practice questions that we collected from various ESL websites.

Hatzivassiloglou and McKeown (1997) propose that antonyms and synonyms can be distinguished by their semantic orientation. A word that suggests praise has a positive semantic orientation, whereas criticism is negative semantic orientation. Antonyms tend to have opposite semantic orientation (fast:slow is positive:negative) and synonyms tend to have the same semantic orientation (fast:quick is positive:positive). However, this proposal has not been evaluated, and it is not difficult to find counter-examples (simple:simplistic is positive:negative, yet the words are synonyms, rather than antonyms).

[7] See *ESL Synonym Questions (State of the art)* at http://aclweb.org/aclwiki/. There were 8 systems at the time of writing, but the list is likely to grow.

Word pair	Class label
galling:irksome	synonyms
yield:bend	synonyms
naive:callow	synonyms
advise:suggest	synonyms
dissimilarity:resemblance	antonyms
commend:denounce	antonyms
expose:camouflage	antonyms
unveil:veil	antonyms

Table 11: Examples of synonyms and antonyms from 136 ESL practice questions.

Lin *et al.* (2003) distinguish synonyms from antonyms using two patterns, "from X to Y" and "either X or Y". When X and Y are antonyms, they occasionally appear in a large corpus in one of these two patterns, but it is very rare for synonyms to appear in these patterns. Our approach is similar to Lin *et al.* (2003), but we do not rely on hand-coded patterns; instead, PairClass patterns are generated automatically.

Using ten-fold cross-validation, PairClass attains an accuracy of 75.0%. Always guessing the majority class would result in an accuracy of 65.4%. The average human score is unknown and there are no previous results for comparison.

3.5 CL Synonyms and Antonyms

To compare PairClass with the algorithm of Lin *et al.* (2003), this experiment uses their set of 160 word pairs, 80 labeled *synonym* and 80 labeled *antonym*. These 160 pairs were chosen by Lin *et al.* (2003) for their high frequency; thus they are somewhat easier to classify than the 136 ESL practice questions. Some examples are given in Table 12.

Lin *et al.* (2003) report their performance using precision (86.4%) and recall (95.0%), instead of accuracy, but an accuracy of 90.0% can be derived from their figures, with some minor algebraic manipulation. Using ten-fold cross-validation, PairClass has an accuracy of 81.9%. Random guessing would yield an accuracy of 50%. The average human score is unknown.

3.6 Similar, Associated, and Both

A common criticism of corpus-based measures of word similarity (as opposed to lexicon-based measures) is that they are merely detecting associations (co-occurrences), rather than actual semantic similarity (Lund *et al.*, 1995). To address this

Word pair	Class label
audit:review	synonyms
education:tuition	synonyms
location:position	synonyms
material:stuff	synonyms
ability:inability	antonyms
balance:imbalance	antonyms
exaggeration:understatement	antonyms
inferiority:superiority	antonyms

Table 12: Examples of synonyms and antonyms from 160 labeled pairs for experiments in computational linguistics (CL).

criticism, Lund *et al.* (1995) evaluated their algorithm for measuring word similarity with word pairs that were labeled *similar*, *associated*, or *both*. These labeled pairs were originally created for cognitive psychology experiments with human subjects (Chiarello *et al.*, 1990). Table 13 shows some examples from this collection of 144 word pairs (48 pairs in each of the three classes).

Word pair	Class label
table:bed	similar
music:art	similar
hair:fur	similar
house:cabin	similar
cradle:baby	associated
mug:beer	associated
camel:hump	associated
cheese:mouse	associated
ale:beer	both
uncle:aunt	both
pepper:salt	both
frown:smile	both

Table 13: Examples of word pairs labeled *similar*, *associated*, or *both*.

Lund *et al.* (1995) did not measure the accuracy of their algorithm on this three-class classification problem. Instead, following standard practice in cognitive psychology, they showed that their algorithm's similarity scores for the 144 word pairs were correlated with the response times of human subjects in priming tests. In a typical priming test, a human subject reads a *priming* word (*cradle*) and is then asked to complete a partial word (complete *bab* as *baby*) or to distinguish a word (*baby*) from a non-word (*baol*). The time required to perform the task is taken to indicate the strength of the cognitive link between the two words (*cradle* and *baby*).

Using ten-fold cross-validation, PairClass attains an accuracy of 77.1% on the 144 word pairs. Since the three classes are of equal size, guessing the majority class and random guessing both yield an accuracy of 33.3%. The average human score is unknown and there are no previous results for comparison.

3.7 Noun-Modifier Relations

A noun-modifier expression is a compound of two (or more) words, a head noun and a modifier of the head. The modifier is usually a noun or adjective. For example, in the noun-modifier expression *student discount*, the head noun *discount* is modified by the noun *student*.

Noun-modifier expressions are very common in English. There is wide variation in the types of semantic relations between heads and modifiers. A challenging task for natural language processing is to classify noun-modifier pairs according to their semantic relations. For example, in the noun-modifier expression *electron microscope*, the relation might be *theme:tool* (a microscope for electrons; perhaps for viewing electrons), *instrument:agency* (a microscope that uses electrons), or *material:artifact* (a microscope made out of electrons).[8] There are many potential applications for algorithms that can automatically classify noun-modifier pairs according to their semantic relations.

Nastase and Szpakowicz (2003) collected 600 noun-modifier pairs and hand-labeled them with 30 different classes of semantic relations. The 30 classes were organized into five groups: causality, temporality, spatial, participant, and quality. Due to the difficulty of distinguishing 30 classes, most researchers prefer to treat this as a five-class classification problem. Table 14 shows some examples of noun-modifier pairs with the five-class labels.

The design of the PairClass algorithm is closely related to past work on the problem of classifying noun-modifier semantic relations, so we will examine this past work in more detail than in our discussions of related work for the other six tests. Section 5 will focus on the relation between PairClass and past work on semantic relation classification.

Using ten-fold cross-validation, PairClass achieves an accuracy of 58.0% on the task of classifying the 600 noun-modifier pairs into five classes. The best previous result was also 58.0% (Turney, 2006). The *ACL Wiki* lists 5 previously pub-

[8]The correct answer is *instrument:agency*.

Word pair	Class label
cold:virus	causality
onion:tear	causality
morning:frost	temporality
late:supper	temporality
aquatic:mammal	spatial
west:coast	spatial
dream:analysis	participant
police:intervention	participant
copper:coin	quality
rice:paper	quality

Table 14: Examples of noun-modifier word pairs labeled with five semantic relations.

lished results with the 600 noun-modifier pairs.[9] Adding PairClass to the list, we have 6 algorithms. PairClass ties for first place in the set of 6 systems. Guessing the majority class would result in an accuracy of 43.3%. The average human score is unknown.

4 Discussion

The seven experiments are summarized in Tables 15 and 16. For the five experiments for which there are previous results, PairClass is not the best, but it performs competitively. For the other two experiments, PairClass performs significantly above the baselines. However, the strength of this approach is not its performance on any one task, but the range of tasks it can handle. No other algorithm has been applied to this range of lexical semantic problems.

Of the seven tests we use here, as far as we know, only the noun-modifier relations have been approached using a standard supervised learning algorithm. For the other six tests, PairClass is the first attempt to apply supervised learning.[10] The advantage of being able to cast these six problems in the framework of standard supervised learning problems is that we can now exploit the huge literature on supervised learning. Past work on these problems has required implicitly coding our

[9] See *Noun-Modifier Semantic Relations (State of the art)* at http://aclweb.org/aclwiki/. There were 5 systems at the time of writing, but the list is likely to grow.

[10] Turney *et al.* (2003) apply something like supervised learning to the SAT analogies and TOEFL synonyms, but it would be more accurate to call it reinforcement learning, rather than standard supervised learning.

Experiment	Vectors	Features	Classes
SAT Analogies	2,244	44,880	374
TOEFL Synonyms	320	6,400	2
ESL Synonyms	200	4,000	2
ESL Synonyms and Antonyms	136	2,720	2
CL Synonyms and Antonyms	160	3,200	2
Similar, Associated, and Both	144	2,880	3
Noun-Modifier Relations	600	12,000	5

Table 15: Summary of the seven tasks. See Section 3 for explanations. The number of features is 20 times the number of vectors, as mentioned in Section 2. For SAT Analogies, the number of vectors is 374×6. For TOEFL Synonyms, the number of vectors is 80×4. For ESL Synonyms, the number of vectors is 50×4.

Experiment	Accuracy	Best previous	Baseline	Rank
SAT Analogies	52.1%	56.1%	20.0%	3 of 13
TOEFL Synonyms	76.2%	97.5%	25.0%	9 of 16
ESL Synonyms	78.0%	82.0%	25.0%	3 of 9
ESL Synonyms and Antonyms	75.0%	-	65.4%	-
CL Synonyms and Antonyms	81.9%	90.0%	50.0%	2 of 2
Similar, Associated, and Both	77.1%	-	33.3%	-
Noun-Modifier Relations	58.0%	58.0%	43.3%	1 of 6

Table 16: Summary of experimental results. See Section 3 for explanations. For the Noun-Modifier Relations, PairClass is tied for first place.

knowledge of the nature of the task into the structure of the algorithm. For example, the structure of the algorithm for latent semantic analysis (LSA) implicitly contains a theory of synonymy (Landauer and Dumais, 1997). The problem with this approach is that it can be very difficult to work out how to modify the algorithm if it does not behave the way we want. On the other hand, with a supervised learning algorithm, we can put our knowledge into the labeling of the feature vectors, instead of putting it directly into the algorithm. This makes it easier to guide the system to the desired behaviour.

Humans are able to make analogies without supervised learning. It might be argued that the requirement for supervision is a major limitation of PairClass. However, with our approach to the SAT analogy questions (see Section 3.1), we are blurring the line between supervised and unsupervised learning, since the training set for a given SAT question consists of a single real positive example (and a single “virtual” or “simulated” negative example). In effect, a single example (such as mason:stone in Table 4) becomes a *sui generis*; it constitutes a class of its own. It may be possible to apply the machinery of supervised learning to other problems that apparently call for unsupervised learning (for example, clustering or measuring similarity), by using this *sui generis* device.

5 Related Work

One of the first papers using supervised machine learning to classify word pairs was Rosario and Hearst's (2001) paper on classifying noun-modifier pairs in the medical domain. For example, the noun-modifier expression *brain biopsy* was classified as *Procedure*. Rosario and Hearst (2001) constructed feature vectors for each noun-modifier pair using MeSH (Medical Subject Headings) and UMLS (Unified Medical Language System) as lexical resources. They then trained a neural network to distinguish 13 classes of semantic relations, such as *Cause*, *Location*, *Measure*, and *Instrument*. Nastase and Szpakowicz (2003) explored a similar approach to classifying general-domain noun-modifier pairs, using WordNet and Roget's Thesaurus as lexical resources.

Turney and Littman (2005) used corpus-based features for classifying noun-modifier pairs. Their features were based on 128 hand-coded patterns. They used a nearest-neighbour learning algorithm to classify general-domain noun-modifier pairs into 30 different classes of semantic relations. Turney (2005; 2006) later addressed the same problem using 8000 automatically generated patterns.

One of the tasks in SemEval 2007 was the classification of semantic relations between nominals (Girju *et al.*, 2007).[11] The problem is to classify semantic relations between nominals (nouns and noun compounds) in the context of a sentence. The task attracted 14 teams who created 15 systems, all of which used supervised machine learning with features that were lexicon-based, corpus-based, or both.

PairClass is most similar to the algorithm of Turney (2006), but it differs in the following ways:

- PairClass does not use a lexicon to find synonyms for the input word pairs. One of our goals in this paper is to show that a pure corpus-based algorithm can handle synonyms without a lexicon. This considerably simplifies the algorithm.
- PairClass uses a support vector machine (SVM) instead of a nearest neighbour (NN) learning algorithm.

[11] SemEval 2007 was the Fourth International Workshop on Semantic Evaluations. More information on Task 4, the classification of semantic relations between nominals, is available at http://purl.org/net/semeval/task4.

- PairClass does not use the singular value decomposition (SVD) to smooth the feature vectors. It has been our experience that SVD is not necessary with SVMs.
- PairClass generates probability estimates, whereas Turney (2006) uses a cosine measure of similarity. Probability estimates can be readily used in further downstream processing, but cosines are less useful.
- The automatically generated patterns in PairClass are slightly more general than the patterns of Turney (2006), as mentioned in Section 2.
- The morphological processing in PairClass (Minnen *et al.*, 2001) is more sophisticated than in Turney (2006).

However, we believe that the main contribution of this paper is not PairClass itself, but the extension of supervised word pair classification beyond the classification of noun-modifier pairs and semantic relations between nominals, to analogies, synonyms, antonyms, and associations. As far as we know, this has not been done before.

6 Limitations and Future Work

The main limitation of PairClass is the need for a large corpus. Phrases that contain a pair of words tend to be more rare than phrases that contain either of the members of the pair, thus a large corpus is needed to ensure that sufficient numbers of phrases are found for each input word pair. The size of the corpus has a cost in terms of disk space and processing time. In the future, as hardware improves, this will become less of an issue, but there may be ways to improve the algorithm, so that a smaller corpus is sufficient.

Human language can be creatively extended as needed. Given a newly-defined word, a human would be able to use it immediately in an analogy. Since PairClass requires a large number of phrases for each pair of words, it would be unable to handle a newly-defined word. A problem for future work is the extension of PairClass, so that it is able to work with definitions of words. One approach is a hybrid algorithm that combines a corpus-based algorithm with a lexicon-based algorithm. For example, Turney *et al.* (2003) describe an algorithm that combines 13 different modules for solving proportional analogies with words.

7 Conclusion

The PairClass algorithm classifies word pairs according to their semantic relations, using features generated from a large corpus of text. We describe PairClass as performing analogy perception, because it recognizes lexical proportional analogies using a form of high-level perception (Chalmers *et al.*, 1992). For given input training and testing sets of word pairs, it automatically generates patterns and constructs its own representations of the word pairs as high-dimensional feature vectors. No hand-coding of representations is involved.

We believe that analogy perception provides a unified approach to natural language processing for a wide variety of lexical semantic tasks. We support this by applying PairClass to seven different tests of word comprehension. It achieves competitive performance on the tests, although it is competing with algorithms that were developed for single tasks. More significant is the range of tasks that can be framed as problems of analogy perception.

The idea of subsuming a broad range of semantic phenomena under analogies has been suggested by many researchers (Minsky, 1986; Gentner, 2003; Hofstadter, 2007). In computational lingistics, analogical algorithms have been applied to machine translation (Lepage and Denoual, 2005), morphology (Lepage, 1998), and semantic relations (Turney and Littman, 2005). Analogy provides a framework that has the potential to unify the field of semantics. This paper is a small step towards that goal.

In this paper, we have used tests from educational testing (SAT analogies and TOEFL synonyms), second language practice (ESL synonyms and ESL synonym and antonyms), computational linguistics (CL synonyms and antonyms and noun-modifiers), and cognitive psychology (similar, associated, and both). Six of the tests have been used in previous research and four of the tests have associated performance results and bibliographies in the *ACL Wiki*. Shared tests make it possible for researchers to compare their algorithms and assess the progress of the field.

Applying human tests to machines is a natural way to evaluate progress in AI. Five of the seven tests were originally developed for humans. For the SAT and TOEFL tests, the average human scores are available. On the SAT test, PairClass has an accuracy of 52.1%, with a 95% confidence interval ranging from 46.9% to 57.3% (using the Binomial Exact test). The average senior high school student applying to a US university achieves 57% (Turney, 2006), which is within the 95% confidence interval for PairClass. On the TOEFL synonym test, PairClass has an accuracy of 76.2%, with a 95% confidence interval ranging from 65.4% to 85.1% (using the Binomial Exact test). The average foreign applicant to a US university achieves 64.5% (Landauer and Dumais, 1997), which is below the 95% confidence interval for PairClass. Thus PairClass performance on SAT is not significantly different from average human performance, and PairClass performance on TOEFL is significantly better than average human performance.

One criticism of AI as a field is that its success stories are limited to narrow domains, such as chess. Human intelligence has a generality and flexibility that

AI currently lacks. This paper is a tiny step towards the goal of performing competively on a wide range of tests, rather than performing very well on a single test.

Acknowledgements

Thanks to Michael Littman for the 374 SAT analogy questions, Thomas Landauer for the 80 TOEFL synonym questions, Donna Tatsuki for the 50 ESL synonym questions, Dekang Lin for the 160 synonym-antonym questions, Christine Chiarello for the 144 similar-associated-both questions, and Vivi Nastase and Stan Szpakowicz for the 600 labeled noun-modifiers. Thanks to Charles Clarke for the corpus, Stefan Büttcher for Wumpus, Guido Minnen, John Carroll, and Darren Pearce for *morpha* and *morphb*, and Ian Witten and Eibe Frank for Weka. Thanks to Selmer Bringsjord for inviting me to contribute to the special issue of *JETAI* on *Test-Based AI*. Thanks to Joel Martin for comments on an earlier version of this paper. I am especially thankful to Michael Littman for initiating my interest in analogies in 2001, by suggesting that a statistical approach might be able to solve multiple-choice analogy questions.

References

(Breiman, 1996) Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

(Bringsjord and Schimanski, 2003) Selmer Bringsjord and Bettina Schimanski. What is artificial intelligence? Psychometric AI as an answer. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 887–893, Acapulco, Mexico, 2003.

(Budanitsky and Hirst, 2001) Alexander Budanitsky and Graeme Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pages 29–24, Pittsburgh, PA, 2001.

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

(Chalmers *et al.*, 1992) David J. Chalmers, Robert M. French, and Douglas R. Hofstadter. High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental & Theoretical Artificial Intelligence*, 4(3):185–211, 1992.

(Chiarello *et al.*, 1990) Christine Chiarello, Curt Burgess, Lorie Richards, and Alma Pollock. Semantic and associative priming in the cerebral hemispheres: Some words do, some words don't ... sometimes, some places. *Brain and Language*, 38:75–104, 1990.

(Evans, 1964) Thomas Evans. A heuristic program to solve geometric-analogy problems. In *Proceedings of the Spring Joint Computer Conference*, pages 327–338, 1964.

(Falkenhainer *et al.*, 1989) Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1):1–63, 1989.

(Gentner, 2003) Dedre Gentner. Why we're so smart. In Dedre Gentner and Susan Goldin-Meadow, editors, *Language in Mind: Advances in the Study of Language and Thought*, pages 195–235. MIT Press, 2003.

(Girju *et al.*, 2007) Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval 2007)*, pages 13–18, Prague, Czech Republic, 2007.

(Hatzivassiloglou and McKeown, 1997) Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL (ACL/EACL-1997)*, pages 174–181, 1997.

(Hirst and St-Onge, 1998) Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–332. MIT Press, 1998.

(Hofstadter, 2001) Douglas Hofstadter. Epilogue: Analogy as the core of cognition. In Dedre Gentner, Keith J. Holyoak, and Boicho N. Kokinov, editors, *The Analogical Mind: Perspectives from Cognitive Science*, pages 499–538. MIT Press, 2001.

(Hofstadter, 2007) Douglas Hofstadter. *I Am a Srange Loop*. Basic Books, 2007.

(Jarmasz and Szpakowicz, 2003) Mario Jarmasz and Stan Szpakowicz. Roget's thesaurus and semantic similarity. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pages 212–219, Borovets, Bulgaria, 2003.

(Jiang and Conrath, 1997) Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X)*, pages 19–33, Tapei, Taiwan, 1997.

(Landauer and Dumais, 1997) Thomas K. Landauer and Susan T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

(Lepage and Denoual, 2005) Yves Lepage and Etienne Denoual. Purest ever example-based machine translation: Detailed presentation and assessment. *Machine Translation*, 19(3):251–282, 2005.

(Lepage, 1998) Yves Lepage. Solving analogies on words: An algorithm. In *Proceedings of the 36th Annual Conference of the Association for Computational Linguistics*, pages 728–735, 1998.

(Lesk, 1969) Michael E. Lesk. Word-word associations in document retrieval systems. *American Documentation*, 20(1):27–38, 1969.

(Lin *et al.*, 2003) Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. Identifying synonyms among distributionally similar words. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pages 1492–1493, 2003.

(Lund *et al.*, 1995) Kevin Lund, Curt Burgess, and Ruth Ann Atchley. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 660–665, 1995.

(Minnen *et al.*, 2001) Guido Minnen, John Carroll, and Darren Pearce. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223, 2001.

(Minsky, 1986) Marvin Minsky. *The Society of Mind*. Simon & Schuster, New York, NY, 1986.

(Nastase and Szpakowicz, 2003) Vivi Nastase and Stan Szpakowicz. Exploring noun-modifier semantic relations. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 285–301, Tilburg, The Netherlands, 2003.

(Platt, 1998) John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press Cambridge, MA, USA, 1998.

(Reitman, 1965) Walter R. Reitman. *Cognition and Thought: An Information Processing Approach.* John Wiley and Sons, New York, NY, 1965.

(Resnik, 1995) Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 448–453, San Mateo, CA, 1995. Morgan Kaufmann.

(Rosario and Hearst, 2001) Barbara Rosario and Marti Hearst. 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)*, pages 82–90, 2001.

(Turney and Littman, 2005) Peter D. Turney and Michael L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1–3):251–278, 2005.

(Turney *et al.*, 2003) Peter D. Turney, Michael L. Littman, Jeffrey Bigham, and Victor Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pages 482–489, Borovets, Bulgaria, 2003.

(Turney, 2001) Peter D. Turney. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-01)*, pages 491–502, Freiburg, Germany, 2001.

(Turney, 2005) Peter D. Turney. Measuring semantic similarity by latent relational analysis. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1136–1141, Edinburgh, Scotland, 2005.

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

(Turney, 2008a) Peter D. Turney. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33:615–655, 2008.

(Turney, 2008b) Peter D. Turney. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 905–912, Manchester, UK, 2008.

(Veale, 2004) Tony Veale. WordNet sits the SAT: A knowledge-based approach to lexical analogy. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 606–612, Valencia, Spain, 2004.

(Witten and Frank, 1999) Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, 1999.