From Atoms to the Solar System: Generating Lexical Analogies from Text

by

Pei-Wen Andy Chiu

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2006

Abstract

A *lexical analogy* is two pairs of words (w_1, w_2) and (w_3, w_4) such that the relation between w_1 and w_2 is identical or similar to the relation between w_3 and w_4. For example, (*abbreviation*, *word*) forms a lexical analogy with (*abstract*, *report*), because in both cases the former is a shortened version of the latter. Lexical analogies are of theoretic interest because they represent a second order similarity measure: *relational similarity*. Lexical analogies are also of practical importance in many applications, including text-understanding and learning ontological relations.

This thesis presents a novel system that generates lexical analogies from a corpus of text documents. The system is motivated by a well-established theory of analogy-making, and views lexical analogy generation as a series of three processes: identifying pairs of words that are semantically related, finding clues to characterize their relations, and generating lexical analogies by matching pairs of words with similar relations. The system uses a *dependency grammar* to characterize semantic relations, and applies machine learning techniques to determine their similarities. Empirical evaluation shows that the system performs remarkably well, generating lexical analogies at a precision of over 90%.

Acknowledgments

I would like to thank the following people, without whom this thesis would not have been possible.

... my supervisors Pascal Poupart and Chrysanne DiMarco for their guidance throughout this research.

... my readers Frank Tompa and Fakhri Karray for their invaluable comments and suggestions.

... my parents Louis and Shirley, sister Betty, brother-in-law Terry, and nephew Joshua, for their never-ending support and encouragement.

... my wife Cybill for her selfless sacrifice to always be there for me.

Above all, I would like to thank God, for all of the above.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Analogies Everywhere

After a few trips with his mother to the local dried food store, Little Joshua quickly learned to use colour to associate dried food with its original form. Purple raisins come from purple grapes, and yellow plantain chips are made from yellow bananas. As Joshua and his mother visited the store again one day, they came across a bucket of black prunes. Joshua's eyes widened as he pointed to the bucket and proclaimed excitedly: "Look Mom, chocolate apricots!"

This true story of my three-year-old nephew demonstrates how analogy plays a central role in the way we reason, learn, and describe the world. The analogy-making mind of Joshua recognizes the colour relation between grapes and raisins and between bananas and plantain chips, transfers this colour relation into the domain of chocolate and apricots, verifies that this transferring does not conflict with his existing knowledge base, and finally forms a new conclusion using this transferred relation. His verbal expression of his new discovery also involves an analogy. In this case, it is the transferring of the language construct from the likes of "toy car" and "strawberry cake" to form the new compound noun "chocolate apricots". Similar analogy-making processes occur in almost everything we do. Analogy allows us to reason that green apples are just as edible as red apples, to learn roller-blading by using what we know about skating , and to describe an atom concisely as a miniature solar system. In fact, analogy is such an integral component of our cognitive experience that it has been argued as indispensable to reasoning and learning [1].

The study of analogy spans a multitude of disciplines that include psychology, education, linguistics, cognitive science, and artificial intelligence. Within the artificial intelligence community, active analogy research exists in at least the following areas: computational models of analogy-making, analogical reasoning, and analogy discovery. Computational models of analogy-making are computer implementations of cognitive theories that explain and model the complex process of analogy-making. Analogical reasoning explores analogy as a reasoning and learning mechanism. For example, case-based reasoning is an especially well-studied branch of analogical reasoning. Lastly, analogy discovery involves the practical problems of identifying, extracting, comparing, and generating analogies automatically from real-world data.

In this thesis, we consider a particular problem in analogy discovery: lexical analogy generation from text.

1.2 Lexical Analogies

Lexical analogies, also called *verbal analogies*, are analogies between pairs of words, or *word-pairs*. A lexical analogy between word-pairs (w_1, w_2) and (w_3, w_4) signifies that the two word-pairs are *relationally similar*. In other words, the relations between w_1 and w_2 and those between w_3 and w_4 are identical or similar in some way. In this case, (w_1, w_2) is called a *lexical analogue* of (w_3, w_4), and vice versa. For a word-pair (w_1, w_2), the relations between w_1 and w_2 are called the word-pair's *underlying relations*. For example, one of the underlying relations of (*planet*, *star*) is *revolves-around*, which makes the word-pair a lexical analogue of (*electron*, *nucleus*). Lexical analogies are denoted either by simply listing the two word-pairs, or by using the specialized notation w_1:w_2::w_3:w_4, which is read "w_1 is to w_2 as w_3 is to w_4". Table 1.1 lists some lexical analogies, along with their underlying relations.

As shown in Table 1.1, the set of relations underlying lexical analogies is extremely large and varied. *Same-as*, *is-a*, *part-of*, *founder-of*, *plays-in*, *loves*, *produces*, *shortened-version-of*, *improves-function-of*, and *revolves-around* are but a small number of possibilities. In fact, lexical analogies can occur even with non-semantic relations. For example, given *write:written::bite::?*, any competent speaker of English can complete the analogy with *bitten*. The underlying relation in this case, however, is merely morphological. In the case of *pick:up::put:down*, the underlying relation is syntactical. Even more extreme examples such as *net:ten::top:pot* and *a:aaa::b:bbb* show that lexical analogies can occur even on word form alone. Indeed, despite the fact that lexical analogies are almost al-

Lexical Analogy	Relation
big:large::small:little	synonymy
car:vehicle::sword:weapon	is-a
finger:hand::toe:foot	part-of
gates:microsoft::jobs:apple	founder-of
pitcher:baseball::striker:soccer	plays-in
romeo:juliet::oliver:jenny	loves
composer:music::poet:poem	produces
abbreviation:word::abstract:report	shortened-version-of
amplifier:ear::telescope:eye	improves-function-of
electron:nucleus::planet:star	revolves-around

Table 1.1: Examples of Lexical Analogies

ways associated with semantic relations, they are a peculiar phenomenon that can occur across all levels of linguistics. In this thesis, however, only lexical analogies with semantic relations are considered.

Lexical analogies are used extensively in both formal writing and everyday dialogue. Explicit analogies such as "*the laser beam cuts through metal like a hot knife through butter*" allow complex situations to be described clearly, concisely, and creatively. Even more common are implicit analogies, in which some component words are implied instead of explicitly specified. For example, "*the printer died*" contains the implicit analogy *person:death::printer:malfunction*, and "*your words seem hollow*" makes use of the implicit analogy *object:hollowness::word:meaninglessness*. The second example comes from Lakoff and Johnson [2], who discuss in detail implicit analogies in the form of linguistic metaphors. Turney and Littman [3] elaborate further on Lakoff and Johnson's analysis in their work on lexical analogy comparison.

1.3 Lexical Analogy Generation

The problem of *lexical analogy generation* can be broadly stated as follows:

> Given a text resource such as a linguistic ontology or a corpus of text documents, return a list of lexical analogies that are explicitly or implicitly contained in the resource.

Lexical analogy generation is closely related to another problem in analogy discovery, namely lexical analogy comparison, which determines the degree of relational similarity between two given word-pairs. Indeed, the ultimate goal of lexical analogy generation is to construct a dictionary of word-pairs that have a high degree of relational similarity.

Lexical analogy generation is a difficult problem because it essentially requires the software agent to identify all possible relations between any two words. Such relations depend not only on the meaning of each word individually, but also on their interactions in different contexts. For example, considering their meanings alone, there is little similarity between *air* and *sound*. However, within the context of conductivity, the fact that air *conducts* sound waves means the two words can be used together to form lexical analogies such as *air:sound::metal:electricity*. Such sophisticated and creative relations are simply not supported by current natural language computing systems and resources. Existing linguistic ontologies such as WordNet [4], for example, are typically limited to a few classical relations such as *is-a* and *part-of*. For the purpose of lexical analogy generation, however, all possible relations must be considered [5].

Despite the difficulty, lexical analogy generation is an important research area because of its tremendous potential in many artificial intelligence applications. For example, text-understanding systems must be able to recognize lexical analogies to correctly interpret figurative sentences such as "*Leafs killed Senators in Game 7*". Text generation systems, as well, can use lexical analogies to generate more varied and creative text. Turney and Littman [3] further point out the importance of lexical analogies in language evolution, machine translation, and information retrieval. Finally, as Sections 1.4 and 6.3 will illustrate, lexical analogies can also serve as a key component in an emerging research area: automated acquisition of ontological relations from text.

1.4 Research Contributions

The product of our research is **GELATI**, a system for **GE**nerating **L**exical **A**nalogies from **T**ext **I**nformation. GELATI is a fully-automated system that generates lexical analogies from a given corpus of unannotated text documents. In this section we discuss some key contributions of our research.

The most important contribution of our research stems from the fact that, contrary to most existing research on lexical analogy generation, GELATI generates lexical analogies by discovering word relations directly from a corpus of unannotated natural language text, instead of relying on the relations defined in a linguistic ontology such as WordNet [4] or HowNet [6]. This independence from linguistic ontologies allows GELATI to generate lexical analogies using a much larger and richer set of relations than is available in linguistic ontologies. For example, GELATI is able to generate the lexical analogy *hostage:release::troops:withdrawal*, which is not possible for systems relying on WordNet as *hostage* and *troops* are two completely unrelated entities in WordNet. Even more importantly, the independence from standard linguistic ontologies means that GELATI can facilitate the automatic construction and enrichment of these ontologies. Current ontologies are largely constructed manually by human domain experts, which is both extremely time-consuming and expensive. As a consequence, current ontologies in general cannot afford to capture anything more than a few of the more common relations. For example, while almost all ontologies contain the *is-a* relation, very few, if any, capture relations such as *revolves-around* or *shortened-version-of*. This deficiency in representing relations is exactly where GELATI can help — the rich set of relations it discovers directly from corpus data can be used to expand an ontology's set of relations automatically. Section 6.3 will discuss this tremendously useful application of GELATI in detail.

The second key contribution of our research is GELATI's flexible and extensible architecture. Motivated by a well-established theory of analogy-making, GELATI divides the complex process of lexical analogy generation into a series of intuitive and manageable components. The componential nature of GELATI allows it to be extended with additional functionalities or adopted into new problem domains simply by changing some component implementations. In this regard, GELATI acts as a framework for analogy generation in general. Chapter 6 explores some possible extensions to GELATI.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews existing research in the artificial intelligence community on analogy, particularly lexical analogy comparison and generation. Chapter 3 discusses the theoretical motivations and intuitions behind GELATI. Chapter 4 describes GELATI and each of its components in detail. Chapter 5 demonstrates the performance of GELATI through an empirical evaluation. Finally, Chapter 6 concludes this thesis with a brief summary and a list of future directions.

Chapter 2

Related Work

In this chapter we review key related research on analogy within the field of artificial intelligence. Section 2.1 describes the most significant cognitive and computational models of analogy-making. Sections 2.2 and 2.3 survey previous research on lexical analogy comparison and generation, respectively.

2.1 Computational Models of Analogy-Making

Modern computer implementations of analogy-making began to appear in the early 1960s. Hall [7] provides an in-depth comparison of analogy-making models and programs up to around 1989, and French [8] offers a more recent but brief survey of over 40 computational analogy-making models.

Structure-Mapping Models

ANALOGY (Evans [9]), developed in 1968, was one of the first programs for analogy-making. ANALOGY solves simple geometric analogy questions: given three geometric figures A, B, and C, where B is obtained by simple transformation of A, select from a list of geometric figures the best match for C using the same or analogous transformation. Each geometric figure is described by some propositional sentences using a set of pre-defined predicates such as *ABOVE* and *INSIDE*. ANALOGY translates the transformation from A to B into a set of transformation rules by mapping each description sentence of A to one in B, then selects the answer by looking for a match of C that best preserves the transformation rules.

Despite its simplicity, the basic intuition behind ANALOGY, that analogy-making is a process of mapping relations, is a far-reaching insight that eventually led to the most influential theory of analogy-making: the Structure-Mapping Theory (Gentner [10]). Gentner's theory formalizes analogy-making as the process of mapping the *structure* of an object from one domain to another, where an object's structure is defined by its relations to other objects as well as the relations of its internal components. Gentner emphasizes that the mapping is structural, and not attributional. For example, although an atom is drastically different from the solar system in terms of size, shape, colour, and other attributes, they are nevertheless analogous because their internal structures are similar — a number of small bodies revolving around a central large body. Gentner also specifies a set of hard and soft rules for determining analogies that are satisfying, such as the preference of higher order relations over lower order ones. Lastly, Gentner justifies his theory with empirical support from a number of cognitive experiments. Falkenhainer et al. [11] turned Gentner's theory into a computer implementation called the Structure-Mapping Engine. To date, the Structure-Mapping Theory is still the basis of many, if not most, computational models of analogy-making.

Taxonomic Abstraction Models

In addition to structure-mapping, another popular approach to analogy-making is *taxonomic abstraction*. The taxonomic-abstraction view of analogy-making can be traced back to as early as 350 BC, when Aristotle offered an explanation of analogy-making and metaphor-making in his Poetics (excerpt from S.H. Butcher's translation [12]):

> Metaphor is the application of an alien name by transference either from genus to species, or from species to genus, or from species to species, or by analogy, that is, proportion.

In the taxonomic-abstraction point of view, an analogy is a match between two objects that share a similar path structure to a common ancestor in a taxonomy. In WordNet [4], for example, an analogy can be drawn between *Zeus* and *Varuna* as *Zeus* is a child of *Greek_deity* and *Varuna* is a child of *Hindu_deity*, and both *Greek_deity* and *Hindu_deity* are children of *deity*. In a way, taxonomic abstraction can be regarded as a highly specialized version of structure mapping in which the structure is entirely specified by the taxonomy. A recent analogy-making model based on the taxonomic-abstraction perspective is KNOW-BEST by Veale [13], which is described in the next section.

Connectionist Models

A completely different approach to implementing analogy-making is the connectionist approach, in which analogy-making is achieved by training a neural network or other similar connectionist structure. Compared to structure-mapping and taxonomic abstraction, connectionist models offer little explanation about the process of analogy-making — the network hides the actual mechanism that determines analogies. Blank's Analogator [14] is one of the most recent connectionist implementations of analogy-making.

Limitation of Computational Models

A common limitation of most of these computational models is that, in order to be general and applicable across a wide variety of problem domains, they typically assume the input data is already in some structured and formal format where analogies can be readily and precisely drawn — many, for example, use input in some form of predicate logic. Unfortunately, real-world data such as natural language text is anything but structured and formal. Drawing analogies from real-world data requires sophisticated and detailed mechanisms to deal with noise and uncertainty, which are beyond the scope of these models.

2.2 Lexical Analogy Comparison

Turney et al. [15] [3] [16] use a corpus-based, statistical approach to lexical analogy comparison. The actual problem that they try to solve is the verbal analogy question in the Scholastic Aptitude Test (SAT). Specifically, an SAT verbal analogy question consists of a word-pair, called the *stem*, and five other word-pairs, called the *candidates*, and the problem is to identify the candidate that forms the best verbal analogy with the stem. Clearly the SAT verbal analogy question is a direct application of lexical analogy comparison — compare the stem with each candidate and select the highest scoring candidate. Table 2.1 illustrates an example of an SAT verbal analogy question. In this example, the correct answer is (a), since *opaque:transparent* and *gaunt:rotund* are both related by the *opposite-of* relation.

The experimental data for this research is a set of 374 SAT verbal analogy questions, and the evaluation metrics include *skipping* (number of skipped questions), *precision* (number of correct answers ÷ number of questions answered), *recall* (number of cor-

Stem:	**opaque:transparent**
(a)	gaunt:rotund
(b)	wary:angry
(c)	thin:elongated
(d)	proud:arrogant
(e)	libel:free

Table 2.1: Example of SAT Verbal Analogy Question

rect answers ÷ total number of questions), and *F-measure* ((2 × precision × recall) ÷ (precision + recall)).

Turney: Ensemble Method

Turney et al.'s first approach [15] is an ensemble method that combines the results of 13 expert modules to form the final answer. The expert modules include a *phrase vector module*, a *thesaurus path module*, nine *lexical relations modules*, and two *similarity modules*. Given two word-pairs (A, B) and (C, D), each module operates as follows. The phrase vector module compares (A, B) and (C, D) by computing a *signature vector* for each word-pair, then uses the cosine angle between the signature vectors as the score. The signature vector for each (X, Y) is computed by counting the number of occurrences of X and Y in short phrases such as "*X of Y*", "*Y in the X*", and "*X * very Y*", across all web documents indexed by AltaVista. A total of 128 hand-crafted short phrases are used, resulting in 128 dimensions in the signature vectors. The thesaurus path module uses WordNet, and compares the edges of the shortest path from A to B against those of the shortest path from C to D. The lexical relation modules again use WordNet. Each lexical relation module checks if A and B are related by a particular lexical relation defined in WordNet such as *synonym*, and if so, checks if C and D are related by the same relation. The nine lexical relations checked by the lexical relation modules are *synonym*, *antonym*, *hypernym*, *hyponym*, *meronym:substance*, *meronym:part*, *meronym:member*, *holonym:substance*, and *holonym:member*. Finally, each similarity module uses a dictionary to compute a similarity score between A and C and one between B and D, then sum the two scores to get the final score. One of the two similarity modules uses *Dictionary.com* as its dictionary, while the other uses *Wordsmyth.net*[1].

[1]Both www.dictionary.com and www.wordsmyth.net are online lexical references that contain information from several dictionaries and thesauri.

As the merging rules used to combine the results require training, the experimental data is divided into a training set of 274 questions and a test set of 100 questions. The ensemble method achieves a precision of 45.0% on the test data under the best merging rule. The phrase vector module has by far the greatest contribution, scoring 38.2% on its own.

Turney: Vector Space Model

The success of the phrase vector module in the ensemble method leads to a more thorough exploration in which the module is cast as an application of the Vector Space Model (VSM) [3]. The VSM algorithm in essence operates identically to the phrase vector module in the ensemble method, but incorporates a mechanism to trade off precision for recall and vice versa. Specifically, the algorithm calculates a confidence over the answer it selects, then uses the confidence as a threshold to decide whether or not to answer the question. A high threshold results in the algorithm being more conservative, hence increasing precision while decreasing recall. Similarly, a low threshold results in increased recall but decreased precision. The performance of the VSM algorithm over the entire experiment data set ranges from about 35% precision / 62% recall, to about 60% precision / 11% recall. The highest F-measure occurs at 47% precision / 47% recall. In addition to using the algorithm to perform lexical analogy comparison, Turney et al. also show the same algorithm can be applied to classifying noun-modifier relations [3].

Turney: Latent Relational Analysis

Turney's most recent work on lexical analogy comparison is an algorithm called Latent Relational Analysis (LRA) [16]. The basic intuition behind LRA is inherited from the VSM algorithm, but the actual computation is substantially different. LRA's main improvements over the VSM algorithm are in four areas. First, for a word-pair (A, B), LRA considers not only (A, B), but also the most significant alternatives formed by the synonyms of A and the synonyms of B, as indicated by a thesaurus. Furthermore, instead of considering the stem word-pair and each candidate word-pair separately, LRA considers the stem word-pair, the candidate word-pairs, as well as their synonym alternatives all at the same time. This means that instead of forming two signature vectors, LRA forms a *signature matrix* that combines the signature vectors of all word-pairs in consideration. Second, LRA looks for actual phrases in which the word-pairs participate in a large corpus, and use those phrases instead of the hand-crafted short phrases in the VSM algorithm. Third, LRA favours phrases that are most differentiating – in other words, phrases whose counts are most different between word-pairs – by giving them higher weights. Finally, before computing the cosine angle, LRA applies singular value decomposition to the signature matrix to reduce its dimension as well as compressing the differences, a process inspired by Latent Semantic Analysis [17].

The precision of LRA on the entire experiment data set is a remarkable 56.1%, with only four questions skipped due to insufficient data. This result is on par with the average score of high-school students writing the SAT. However, due to the heavy computational requirement of singular value decomposition, LRA requires over nine days to run through the experiment data set on a 2.4 gigahertz processor.

Veale: KNOW-BEST

Veale [13] takes an entirely different approach to lexical analogy comparison. The algorithm here, called KNOW-BEST, is a knowledge-based method that depends heavily on WordNet. The problem that Veale considers is again the SAT verbal analogy problem. In fact, Veale uses the same experiment data set provided by Turney et al [15]. Given two word-pairs (A, B) and (C, D), KNOW-BEST computes a score for (A, C) and one for (B, D), then linearly combines them to form the final score. The score for each (X, Y) is computed from their relative positions in the WordNet taxonomy, as well as the position of their lowest common ancestor. In addition, KNOW-BEST augments the score in the following ways. First, the definition of common ancestor is relaxed to include ancestors who are not the same node but share a common lexical root or modifier. For example, in WordNet, the only common ancestor between {*seed*} and {*egg*} is the root node {*entity, something*}. However, the parent of {*seed*} is {*reproductive_structure*} and the parent of {*egg*} is {*reproductive_cell*}. As the two parents share a common lexical modifier *reproductive*, KNOW-BEST considers both nodes to be valid common ancestors of (*egg*, *seed*), and hence selects whichever is lower in the hierarchy as the lowest common ancestor. Secondly, KNOW-BEST augments each score with a count of the number of overlapping adjectival terms in the glosses of X and Y. For example, from the WordNet hierarchy alone, (*abbreviation*, *abridgement*) has a very low score as the two words are very far apart. However, as both words share *shortened* in their glosses, their score is increased to reflect this similarity.

As with LRA, KNOW-BEST solves an SAT verbal analogy question by comparing the stem with each candidate then selecting the highest scoring candidate as the answer. In addition, KNOW-BEST implements some filtering mechanisms to eliminate less likely candidates. For example, if in the stem (A, B), A is a direct ancestor of B, then all candidates that do not have this property are eliminated.

KNOW-BEST achieves a precision of 42% over the entire experiment data set, with no skipped questions. If KNOW-BEST is limited to only providing an answer when both words in the stem fall under {*entity*} in WordNet, it achieves a higher precision of 53% at the expense of skipping 76% of the questions.

2.3 Lexical Analogy Generation

Veale: Analogical Thesaurus

Veale [18] considers the problem of constructing an *analogical thesaurus*. A traditional thesaurus takes a word and returns a list of near-synonyms. An analogical thesaurus, on the other hand, takes two words W and D, and returns the analogical equivalents of W in the domain of D. For example, given *Bible* and *Muslim*, an analogical thesaurus would return *Koran* since the Koran is the analogical equivalent of the Bible in the Muslim domain. Clearly, lexical analogy is at the heart of an analogical thesaurus. The example above, for example, is derived directly from the lexical analogy *Bible:Christian::Koran:Muslim*. Indeed, constructing an analogical thesaurus is a restricted form of lexical analogy generation, in which only analogues within the specified domain are allowed to be generated.

Veale's approach makes use of the rich semantic information contained in WordNet [4], both in its formal taxonomy, and in the informal textual glosses that WordNet includes for each entry. The basic algorithm proceeds as follows. Given a source word W and a target domain D, the algorithm first retrieves the set of entities S in WordNet whose textual glosses contain D. S is considered the set of all domain elements of D. The algorithm then proceeds to select the best match of W in S, by computing the relative distance between W and each element of S in the WordNet taxonomy.

The basic algorithm has two important limitations, both of which Veale also addresses. The first limitation is due to the informal nature of the textual glosses. Matching D to the glosses often fails due to the glosses using synonyms of D instead of D itself. The gloss for {*Koran*}, for example, is "*sacred writings of Islam*", which would not match the domain *Muslim*. To counter this problem, Veale introduces an extended notion of synonymy called *symmetric association*, and uses it to expand the search space to include not only exact matches of D but also all entities related to D.

The second issue is that the WordNet taxonomy is often not sufficiently discriminating to support the selection process. For example, given *Zeus* and *Hindu*, the correct analogy to return is *Varuna* as both *Zeus* and *Varuna* are the supreme deity of their respective religion. Using just the WordNet taxonomy, however, it is impossible to select between *Varuna* and, for example, *Ganesh* (the Hindu deity of wisdom) as both {*Varuna*} and {*Ganesh*} are direct hyponyms of {*Hindu_deity*}. Even worse, there may be other entities in the set of candidates that are not even deities, but are nonetheless close to {*Zeus*} in the taxonomy.

The solution that Veale presents involves two mechanisms which are elaborated in the following subsections. First, candidates in S are filtered according to the taxonomic branches in which they are located. Second, new taxonomic relations are dynamically created to support fine-grained selection. Continuing the example of *Zeus* and *Hindu*, the algorithm would first filter out all candidates that are not direct or indirect hyponyms of {*deity*}. Then, both {*Zeus*} and {*Varuna*} would be made children of a new entity {*supreme_deity*}, allowing the algorithm to trivially select *Varuna* instead of another Hindu deity.

Analogical Pivot

In order to support the first mechanism, Veale introduces the notion of an *analogical pivot*. In essence, given W and D, the analogical pivot is the lowest node in the taxonomy that explicitly separates the domain of W and the domain specified by D. For example, the analogical pivot of *Zeus* and *Hindu* is {*deity*}, because {*deity*} is the node at which {*Greek_deity*} and {*Hindu_deity*} are separated. The analogical pivot is therefore the junction point at which the analogy shifts from the source domain into the target domain. More importantly, the analogical pivot acts as a strong filter, as all candidates that are not descendants of the pivot can be eliminated. To compute the analogical pivot, the algorithm determines the domain of W by looking at its ancestors, finds its counterpart in D by using morphological rules, then assigns the lowest common parent of the two as the analogical pivot. In the case of *Zeus* and *Hindu*, the algorithm determines that the domain of {*Zeus*} is its direct hypernym {*Greek_deity*}, builds its counterpart {*Hindu_deity*} in the *Hindu* domain by morphologically combining *Greek_deity* and *Hindu*, and finally takes their common parent, {*deity*}, to be the analogical pivot.

Dynamic Type

To support the second mechanism, Veale once again makes use of WordNet's textual glosses. Each content word in all glosses is analyzed to determine its *differentiating potential* — how suitable can the word act as a differentiator between two or more entities. For example, the word "*supreme*" is a good differentiator between ({*Varuna*}, {*Zeus*}) and ({*Ganeth*}, {*Athena*}), as "*supreme*" appears in the glosses of the first group but not the glosses of the second group. All words with high differentiating potential are turned into new nodes, called *dynamic types*. For each new node, a hyponymy link is added between the new node and each entity whose gloss contains it. Dynamic types allow fine-grained selection to be made. Given the dynamic type {*supreme*}, for example, {*Varuna*} would be differentiated from {*Ganeth*} because the former is now a child of {*supreme*} whereas the latter is not. Moreover, dynamic types often lead to new analogical pivots. For example, in WordNet, all letters from all alphabets are direct hyponyms of {*letter*}. Therefore, there is only one domain under {*letter*}, and hence the node cannot act as an analogical pivot. However, if dynamic types of {*Greek_letter*} and {*Hebrew_letter*} are established, {*letter*} would then become an analogical pivot between, for example, {*alpha*} in {*Greek_letter*} and {*aleph*} in {*Hebrew_letter*}.

Veale's [18] experimental evaluation uses WordNet 1.6. In total, 9,822 new dynamic types are created, and 28,998 new hyponymy relations are added. The quality of these new additions is demonstrated by comparing the analogy-making power of WordNet by itself against its augmented form. In the domain of alphabetical letters, for example, mapping from a Greek letter to the corresponding Hebrew letter yields a precision of 4% in WordNet alone as all letters are direct hyponyms of {*letter*}, while it improves to 96% in the augmented WordNet.

Veale: Analogy Generation Using HowNet

Veale [19] [20] continues to elaborate and refine this system of analogical pivots and dynamic types. His latest work on lexical analogy generation [21] again follows the same intuition, but foregoes WordNet in favour of a more formal bilingual lexical ontology called HowNet [6]. Similar to WordNet, HowNet is a lexical ontology with a taxonomic structure. Unlike WordNet, however, HowNet does not associate each word with a gloss, but instead a formal predicate-based definition. For example, the definition in HowNet for {*surgeon*}, ignoring the bilingual component, is "*surgeon* = {*human* : *HostOf* =

$\{Occupation\}$, $domain = \{medical\}$, $\{doctor : agent = \{\sim\}\}\}$", where the $\sim$ serves as a self-reference. The verbal interpretation of the above definition is "a surgeon is a human with an occupation in the medical domain who acts as the agent of a doctoring activity" [13]. Veale's [21] primary contribution in this case is to demonstrate that HowNet is able to support analogy-making using both a taxonomic approach similar to the dynamic types, as well as a structure-mapping approach that matches the relational structures between entities to facilitate analogy-making. The formal definitions that HowNet offers are crucial, as they allow word relations to be determined precisely.

Chapter 3

GELATI - Theory and Intuitions

In this chapter we present the theory and intuitions behind GELATI, our system for **GE**nerating **L**exical **A**nalogies from **T**ext **I**nformation. Section 3.1 grounds GELATI to the well-established Structure-Mapping Theory of analogy-making, and Sections 3.2 to 3.4 describe the necessary extensions to the theory in order to apply it to the problem of lexical analogy generation.

3.1 Theoretical Grounding

On a conceptual level, GELATI follows Gentner's [10] Structure-Mapping Theory (see Section 2.1) and views an analogy as a mapping between two objects sharing an identical or similar structure. However, SMT by itself is insufficient for the purpose of lexical analogy generation. There are two main limitations. First, as a general theory that is application-independent, SMT leaves open the necessary details for specific applications. For example, in SMT, the basic unit from which analogies are drawn are collectively referred to as *objects*. But what exactly are objects? Are they physical items or ontological concepts? Are all objects suitable for analogy-making? What corresponds to objects in the context of lexical analogy generation, and what are their *structures*? The second limitation is that SMT is explanatory in nature rather than constructive. SMT offers an explanation about *why* two objects form an analogy, but does not provide a procedure on *how* analogies can be generated in the first place.

Clearly, in order to apply SMT to the problem of lexical analogy generation, it must be augmented with extensions specialized for lexical analogies. In the following sections, we propose three key extensions. We will refer to these extensions collectively as the

Structure-Mapping Theory for Lexical Analogies, or SMT-LA.

3.2 Objects: Semantically-Related Word-Pairs

The first extension in SMT-LA deals with the notion of *objects* in SMT. Given that lexical analogies are mappings between word-pairs sharing identical or similar underlying relations, it is not difficult to conclude that objects in the context of lexical analogy generation are word-pairs. However, not all word-pairs are suitable for the purpose of lexical analogies. Consider, for example, the word-pair (*mushroom*, *engagement*): what lexical analogies can it form? Obviously, the answer is none, because the word-pair itself lacks a meaningful underlying relation. In the framework of SMT, such a word-pair would correspond to an object with no structure at all, and hence cannot be *structurally* mapped to another object. Similarly, a word-pair with a singular underlying relation not shared by any other word-pair also cannot be used to form lexical analogies. Lastly, as the focus of this research is semantic lexical analogies, a word-pair with a non-semantic underlying relation such as (*ten*, *net*) is also not suitable.

Even if a word-pair *does* have a clear and non-singular semantic underlying relation, it is still not necessarily a good candidate for lexical analogies. In general, good lexical analogies are derived from relations that are precise and specific. Word-pairs such as (*company*, *right*) or (*pencil*, *entity*) are difficult to form good lexical analogies with because their underlying relations (*has* or *entitled-to* for the former, and *is-a* for the latter) are either ambiguous or overly general.

In light of this vagueness surrounding the notion of objects in SMT, our first extension in SMT-LA is to give a precise definition of an object in the context of lexical analogy generation. Specifically, we propose that an object is a word-pair with at least one clearly identifiable underlying relation R that satisfies the following criteria:

1. R must be semantic in nature.
2. R must be shared by at least one other word-pair.
3. R must be reasonably precise, which means it must have a clear and unambiguous semantic interpretation.
4. R must be reasonably specific, which means it is not shared by too many other word-pairs.

3.3 Structure: Word-Pair Features

The second extension involves the notion of *structure* in SMT. SMT defines an object's structure as its external relations with other objects, and its internal relations between its components. Clearly, the internal relations of a word-pair are exactly its underlying relations. What are the external relations of a word-pair? In general, there are none — most word-pairs are not meaningfully related. Moreover, when two word-pairs *are* related, the relation is almost always due to a correspondence in their underlying relations. For example, (*generosity*, *kindness*) and (*greediness*, *rudeness*) appear to be related, but the relation really stems from their having opposite underlying relations. As such, we propose that the structure of a word-pair is equivalent to its set of underlying relations.

Unfortunately, defining a word-pair's structure by its underlying relations gives rise to a practical problem: it is very difficult to precisely define a word-pair's underlying relations especially when no semantic knowledge about the words of the word-pair is given. This is exactly the case in lexical analogy generation from text in which the input, a corpus of text documents, contains a large amount of syntactic information between words but no semantic information about each word. The only reason that a person can conclude the underlying relation of (*poet*, *poem*) is *produces* is because she or he actually knows the meanings of *poet* and *poem*. Even when semantic knowledge about the words is available, assigning a precise label to the underlying relation is still difficult. A precise label for the underlying relation of (*poet*, *poem*), for example, would need to take into account that the production is artistic and professional in nature, so as to distinguish it from the underlying relation of (*cow*, *milk*). Therefore, instead of defining a word-pair's structure by its underlying relations, we propose to define a word-pair's structure loosely by the syntactic, semantic, and pragmatic clues that give hints about its underlying relations. These clues can include phrases in which the word-pair most frequently appear, words with which the word-pair co-occurs, and so on. We refer to all clues collectively as the word-pair's *features*.

Based on the above intuition, our second extension in SMT-LA is as follows. The structure of a word-pair is defined either by its underlying relations if available, or by the features that characterize these underlying relations if the relations cannot be explicitly inferred.

3.4 Analogy: Feature Matching

The last extension in SMT-LA deals with the actual generation of lexical analogies. SMT states that an analogy is a mapping between two objects with a similar structure. A trivial algorithm for analogy generation, therefore, is to simply compare every pair of objects and select the ones whose structures match. However, when an object's structure is not precisely defined, as is the case with a word-pair and its features, what constitutes a structural match becomes difficult to interpret. Should two word-pairs sharing one common feature out of a hundred be considered a valid match? What about two word-pairs sharing no common features, but all of their features are in fact semantically similar?

To resolve these issues, we propose to view the validity of the mapping between two word-pairs not as a binary dichotomy but as a continuum. The term *analogousness* is used to refer to the quality of a mapping between two word-pairs — in other words, their degree of relational similarity. Word-pairs with high analogousness form good lexical analogies, while those with low analogousness form poor or even invalid lexical analogies. Clearly the actual measure for analogousness depends on the intended use of the resulting lexical analogies. In general, however, the analogousness between two word-pairs should correspond to how much overlapping there is between their features.

Summing up, the following definition is the final extension in SMT-LA: A lexical analogy is two word-pairs whose analogousness is higher than some threshold $K_{threshold}$, where analogousness is determined by an application-specific function that is proportional to the amount of overlap between the two word-pairs' set of features.

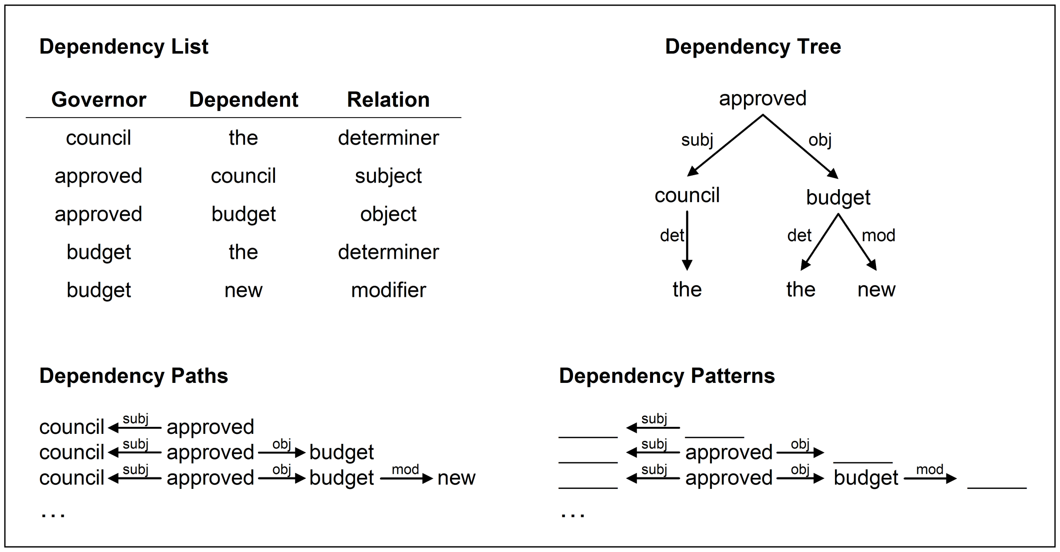
Chapter 4

GELATI - Implementation

In this chapter we present the implementation details of GELATI. We begin by describing an important linguistic formalism used throughout GELATI's implementation, *dependency grammar*, in Section 4.1. We then proceed to give an overview of GELATI in Section 4.2, followed by detailed discussions of each component in Sections 4.3 to 4.8.

4.1 Dependency Grammar

A key structure behind GELATI's implementation is that of a *dependency grammar* [22]. Dependency grammar is a linguistic formalism that describes the syntactic structure of a sentence much like the familiar phrase-structure grammar. Unlike phrase-structure grammars, which associate each word of a sentence to the syntactic phrase in which the word is contained, a dependency grammar associates each word to its syntactic superordinate as determined by a set of rules. Consider, for example, the sentence "*the council approved the new budget*". In this sentence, the first word, "*the*", is a determiner for the second word, "*council*". As such, the syntactic function of "*the*" is only meaningful in the context of "*council*". Dependency grammar hence dictates that "*the*" depends on "*council*". Similarly, "*council*" depends on "*approved*", "*new*" depends on "*budget*", and so on. Each pair of depending words is called a *dependency*. Within a dependency, the word being depended on is called the *governor*, and the word depending on the governor is called the *dependent*. Each dependency is also labelled with the syntactic relation between the governor and the dependent. For example, the dependency (*the*, *council*) from the example above would be labelled *determiner*. Dependency grammars require that each word of a sentence have exactly one governor, except for one word called the *head word* that has no governor at all. The dependency structure of a sentence can be concisely represented by a *dependency tree*, in which each word becomes a node, each dependent becomes a child of its governor, and the head word becomes the root. A *dependency path* is an undirected path through a dependency tree, and a *dependency pattern* is a dependency path with both ends replaced by slots [23]. Figure 4.1 illustrates various dependency structures of the sentence "*the council approved the new budget*".


Dependency List
Dependency Tree
approved
subj
obj
council
budget
det
det
mod
the
the
new
Dependency Paths
council ← subj → approved
council ← subj → approved —obj→ budget
council ← subj → approved —obj→ budget —mod→ new
...
Dependency Patterns
_____ ← subj → _____
_____ ← subj → approved —obj→ _____
_____ ← subj → approved —obj→ budget —mod→ _____
...

Governor	Dependent	Relation
council	the	determiner
approved	council	subject
approved	budget	object
budget	the	determiner
budget	new	modifier

Figure 4.1: Dependency Structures of "*the council approved the new budget*"

Proposition Collapsing

An important advantage of using dependency grammar over phrase-structure grammar is that dependencies allow directly related words to be linked together even if they are not adjacent in a sentence. Unfortunately, this advantage does not hold for phrases involving propositions. Consider the phrase "*rice from China*", for example. In this phrase, clearly there is a direct relation between "*rice*" and "*China*". However, the dependency structure of this phrase contains two dependencies, one linking "*rice*" to "*from*" and one linking "*from*" to "*China*", instead of a single dependency linking "*rice*" to "*China*". To resolve this issue, Lin and Pantel [23] propose a simple transformation that collapses propositions. Specifically, the transformation involves merging two dependencies of the form (*word 1*, *proposition*) and (*proposition*, *word 2*) into a single dependency (*word 1*, *word 2*), then setting "*proposition*" as the relation label of the new dependency. Figure 4.2 illustrates an example of proposition collapsing of the phrase "*rice from China*".

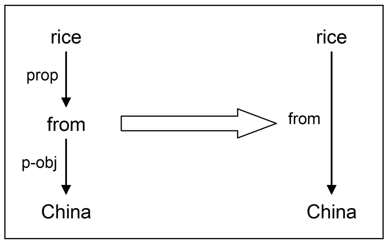


Figure 4.2: Proposition Collapsing of "*rice from China*"

4.2 GELATI Overview

GELATI is a componential system arranged in a pipeline architecture, as depicted in Figure 4.3. The core of GELATI is a series of three components corresponding to the three extensions discussed in Chapter 3. The word-pair extractor extracts semantically related word-pairs from the input corpus; the feature extractor extracts the word-pairs' syntactic features; and the analogy generator maps word-pairs with similar features to form lexical analogies. In addition to these core components, GELATI also includes a preprocessor at the beginning of the pipeline, and various filters throughout the system.

4.3 Preprocessor

The preprocessor is the first component in GELATI's pipeline. It is responsible for converting the raw input data into a list of dependency trees. Specifically, the preprocessor involves the following tasks:

Text Extraction

Most documents available in large text corpora are not stored as straight text files, but in structured formats such as Standard Generalized Markup Language (SGML). Often each document also includes some additional information on top of its content, such as its publication date and keywords. The first task that the preprocessor performs, therefore, is to extract the actual text content of each document and strip away all unnecessary additional data. Moreover, some formats such as SGML encode special characters into entity strings — for example, SGML uses &amp; to represent the & character. The preprocessor therefore must also convert these strings back to the characters they represent. Naturally, as both text extraction and character conversion are format-specific,

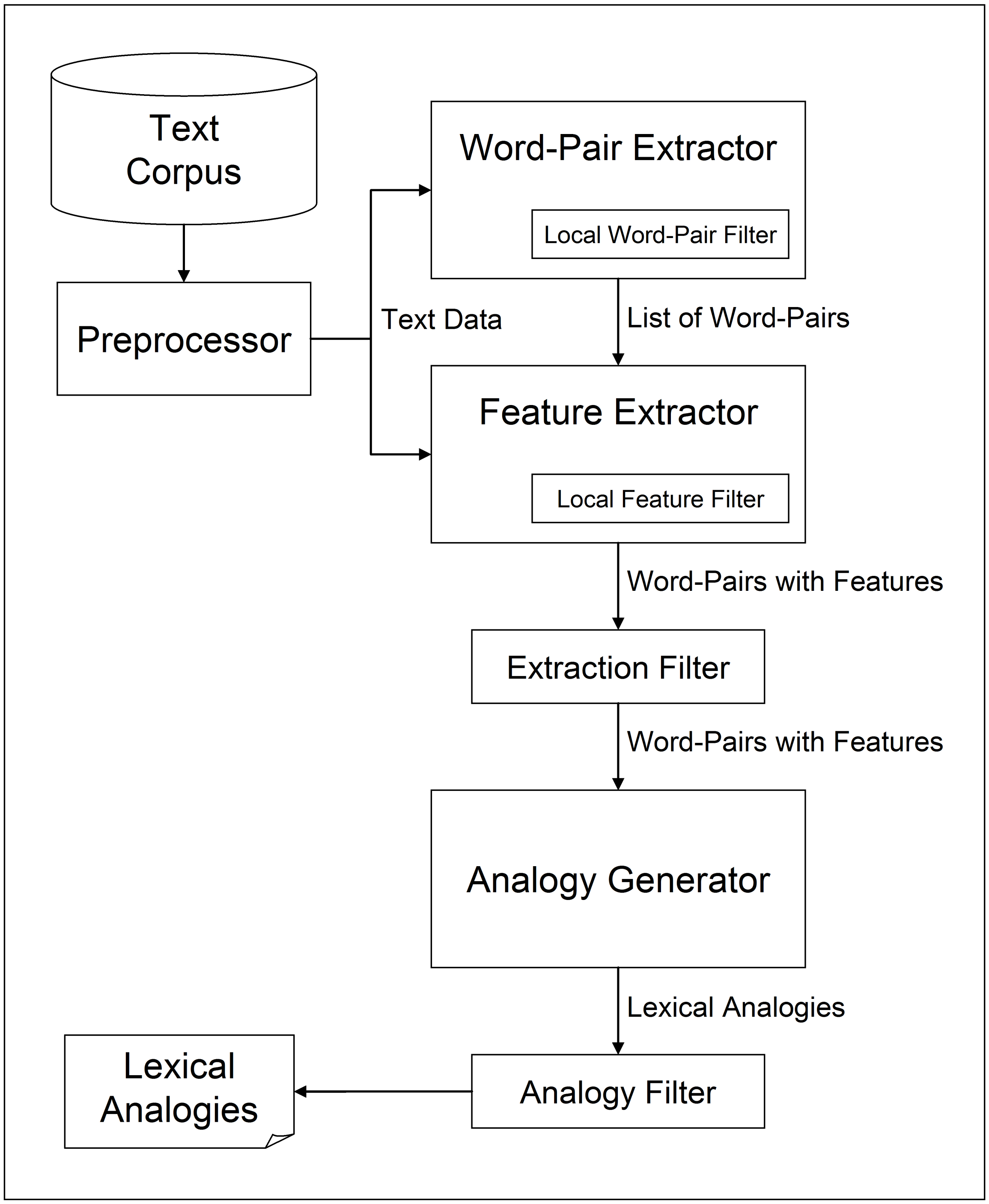


Figure 4.3: Architectural Overview of GELATI

the preprocessor needs to implement different extraction modules to handle different formats. Currently the preprocessor is able to handle SGML, Hypertext Markup Language (HTML), and plain text documents.

Sentence Segmentation

The next task that the preprocessor performs is to segment the text data into individual sentences. Although segmentation may appear trivial at first, it is actually quite complex due to the fact that sentence termination punctuations are also frequently used for other purposes. For example, in English, the period denotes both sentence termination and word abbreviation. A segmenter that relies solely on periods, therefore, would erroneously split the single sentence "*Mr. Smith arrived at the U.S. on Jan. 2005.*" into five sentences. To properly perform sentence segmentation, the preprocessor uses MxTerminator [24], an accurate and efficient sentence segmenter that detects sentence boundaries statistically.

Dependency Parsing

The last task that the preprocessor performs is to parse each sentence into a dependency tree. For this task, the preprocessor uses a dependency parser called MINIPAR [25]. MINIPAR is a broad-coverage English parser based on an efficient distributed chart-passing algorithm. MINIPAR has excellent accuracy, showing a precision of close to 90% [26] on the SUSANNE corpus [27]. More importantly, MINIPAR is very fast, averaging close to 10,000 characters per second on our test system with a 2.5 gigahertz processor. For each sentence, MINIPAR generates a list of dependency relations between words, which the preprocessor organizes into a dependency tree. MINIPAR also additionally performs part-of-speech tagging, word stemming, simple compound noun recognition, dependency relation labelling, and proposition collapsing, all of which the preprocessor also incorporates into the dependency trees.

4.4 Word-Pair Extractor

The word-pair extractor is the first of the three core components of GELATI. As the name implies, the word-pair extractor is responsible for extracting a list of word-pairs from the input corpus that will act as the basic building blocks from which lexical analogies are formed. As discussed in Section 3.2, not all word-pairs are suitable for the purpose of lexical analogy. In particular, only word-pairs with a clear, precise, and specific semantic underlying relation should be extracted.

In order to determine what words are semantically related, we make use of the following hypothesis: highly syntactically related words also tend to be semantically related. This hypothesis allows a sentence's syntactic dependency structure to approximate the semantic relations between its words. Therefore, two words are assumed to be semantically related if they are syntactically related by some dependency relations — in other words, related through a dependency path. As such, to extract a list of semantically related word-pairs, the word-pair extractor simply takes the dependency trees from the preprocessor then generates all possible dependency paths. Each dependency path results in a syntactically related, and hence semantically related, word-pair, defined by the two words at the ends of the path.

Obviously, this list of word-pairs is extremely large — every pair of words occurring in a same sentence would be extracted! Moreover, many of these word-pairs are very weakly related, if at all. In order to keep only the most relevant word-pairs, the word-pair extractor applies two filters on the list: a set of constraints on the dependency paths, and a list of stop words on the end words.

Dependency Path Constraints

The first filter that the word-pair extractor uses is a set of constraints on the dependency paths. Each dependency path is tested against these constraints, and only those that pass through all constraints are actually extracted. The constraints are:

1. The dependency path must contain exactly three words.

2. Both ends of the dependency path must be nouns.

3. The intermediate word must be a verb.

The first constraint places a restriction on the length of the dependency paths to extract. It is obvious that each dependency path must be of length at least two, or else it would not be possible to extract a *pair* of words from the path. It is also obvious why shorter dependency paths are preferred — a shorter dependency path means it goes through fewer dependency relations, and hence the relation between its end words is more direct and stronger. On the other hand, it is perhaps surprising that dependency paths of length three are actually more appropriate than dependency paths of length two. The end words of a dependency path of length two are directly related by a dependency relation, which generally means they are also semantically related. However, the semantic relation between such words is often very general. For example, consider a sentence's subject, which is usually a direct dependent of the sentences's main verb. The semantic relation between them, however, is simply that the subject *is-able-to-perform* the action specified by the verb, a very general relation. On the other hand, a dependency path of length three forces a more specific semantic relation, because the relation has to at least conform to the middle word. In essence, the middle word acts as a specifier that describes the semantic relation between the end words. For these reasons, only dependency paths of length three are extracted.

The last two constraints are inspired by the fact that the noun-verb-noun pattern is the most common and direct construct to relate two nouns in English, and by a similar set of constraints in Lin's [23] work on inference rules. These constraints therefore help the word-pair extractor to pinpoint strongly related nouns, at the expense of neglecting semantically related adjectives, adverbs, and verbs. We justify this decision by acknowledging that the majority of our lexical analogies are indeed based on nouns as in general nouns have the most inter-relations. Nevertheless, one of our immediate future directions is to relax these constraints to allow non-noun words to be extracted. These constraints also highlight an important advantage of performing pattern extraction on dependency trees rather than on the original unparsed sentences — words that are involved in the pattern do not necessarily have to be adjacent to each other. In "*the council approved the new budget*", for example, *council-approve-budget* would be a path to extract even though *approve* and *budget* are separated by two other words.

Stop Words

In addition to the constraints on the dependency paths, the word-pair extractor also filters out any word-pair containing words from a list of 37 stop words as listed in Table 4.1.

The stop words in group one are due to noise in the input data. The word *'n* is especially common, sometimes as an abbreviation for *and* (such as *rock'n'roll*) and sometimes simply as a typo when writing contractions involving *not*. Group 2 stop words are mostly caused by parser error, as MINIPAR sometimes mistakenly tags these words as nouns when they should be adverbs or pronouns. On the other hand, there are rare circumstances when these words really *are* used as nouns. In those cases, however, these words

Group	Stop Words
1	'n, 't
2	no, yes, now, there, here, many, few, more, less, higher, lower, these, those, something, nothing, someone, no one, last
3	first, second, third, hundred, thousand, million, year, week, month, day, time, date
4	man, woman, part, parts, way

Table 4.1: List of Stop Words

merely act as referring expressions and do not contain any meaning themselves. For example, in the sentence "*eagles tend to nest on the higher*", the word "*higher*" really refers to a higher platform, a higher tree, or a higher something else. The semantic relation is therefore between *eagle* and that *something*, and not between *eagle* and *higher*. Group 3 stop words are similar to Group 2, with the exception that they *are* actually used as meaningful entities in some situations. In the sentence "*a million is a large number*", for example, there is indeed a semantic relation between *million* and *number*. Such uses, however, occur rarely and hence these words are still filtered out. The last group of stop words are filtered out because their uses are extremely broad and general, especially in non–domain-specific documents such as newspaper articles. Word-pairs involving these words, therefore, rarely have clear and precise underlying relations, and are therefore not very useful for lexical analogies.

Most of the stop words, especially those in Group 1 and 4, are tailored specifically for the input corpus used in our experimental evaluation (see Chapter 5). If a different corpus is to be used, the list of stop words will also likely need to be updated.

4.5 Feature Extractor

The feature extractor is the second core component of GELATI and corresponds to the second extension discussed in Section 3.3. The feature extractor is responsible for extracting a set of features — syntactic, semantic, and pragmatic clues — about the underlying relations of each word-pair extracted by the word-pair extractor.

To determine which features to extract, we again turn to dependency grammar. Recall that each word-pair extracted by the word-pair extractor comes from a dependency path of the form: noun-verb-noun. This path, and specifically the middle verb, is precisely something that describes the semantic relation between the two nouns — in other words, a feature. Therefore, for each word-pair extracted by the word-pair extractor, the feature extractor simply extracts the dependency pattern derived from the word-pair's originating dependency path, and sets the dependency pattern as a feature of the word-pair. In addition, the feature extractor utilizes the same set of filters as the word-pair extractor. As an example, consider again the sentence "*the council approved the new budget*". The word-pair extractor would extract the word-pair (*council*, *budget*), while the feature extractor would extract the dependency pattern "$\_\_\_\_\_ \overset{subj}{\leftarrow}$ *approve* $\overset{obj}{\rightarrow} \_\_\_\_\_$" and set it as a feature of (*council*, *budget*).

Dependency patterns have previously been shown to be effective at characterizing lexico-syntactic relations [23] [28]. As Chapter 5 demonstrates, they are also surprisingly capable of characterizing word-pairs' underlying relations. As such, currently dependency patterns are the only features that GELATI extracts. However, there are many other possible features that may benefit GELATI's performance. We explore some of these possibilities for future work in Chapter 6.

The final output of the feature extractor is the same list of word-pairs received from the word-pair extractor, but with each word-pair's features appended to the word-pair's entry.

4.6 Extraction Filter

In addition to the local filters used in the word-pair and the feature extractors, GELATI also employees a set of global extraction filters after both extractors complete. The reason that global filters are required in addition to local filters is because some useful information for filtering do not become available until all extractions are finished. For example, the number of word-pairs sharing a feature is a good indicator of the generality of that feature. In the extreme case, if all word-pairs share a feature, the feature is obviously much too general to be useful and therefore should be filtered out. Such information, however, does not become available until the features of all word-pairs have been extracted. GELATI uses four global extraction filters as described below:

Occurrence Filters

The first global extraction filter counts the number of occurrences of each word-pair in the entire corpus, and eliminates those that occur less than K_{filter_1} times. This filter is based on the assumption that, as long as the input corpus is sufficiently large, strongly related word-pairs tend to occur repeatedly. This simple yet effective filter therefore weeds out weakly related word-pairs. Table 4.2 lists the first six word-pairs that occur once, twice, five times, and 25 times in a corpus of about two hundred megabytes. As the table shows, word-pairs that occur more frequently indeed tend to be more strongly related.

Occurrences	Sample Word-Pairs
1	(report, ex), (gratitude, treatment), (authority, war criminal), (prisoner, war criminal), (number, publicity), (total, camp)
2	(officer, camp), (camp, amnesty), (delegation, ceremony), (citizen, society), (ground, poll), (effect, battle)
5	(gratitude, government), (prisoner, charge), (jail, death), (decision, speech), (pledge, tax increase), (department, evidence)
25	(support, people), (ratification, treaty), (group, activity), (demand, release), (union, employee), (dollar, europe)

Table 4.2: Word-Pair Frequency Samples

The second global extraction filter is the dual of the first — it counts the number of occurrences of each *feature* and eliminates those occurring less than K_{filter_2} times.

Unique Occurrence Filters

The third global filter considers the number of distinct features for each word-pair. A word-pair that has many features means that there are many different ways to characterize its underlying relations, and hence the relations are most likely overly general. On the other hand, a word-pair that has very few features is difficult to match to another word-pair — there just is not enough material to make meaningful comparisons. The filter, therefore, eliminates all word-pairs that have more than K_{filter_3} or less than K_{filter_4} features.

The last global extraction filter is the dual of the third — it considers the number of distinct word-pairs for each *feature*, and eliminates those that are associated with more than K_{filter_5} or less than K_{filter_6} word-pairs.

4.7 Analogy Generator

The analogy generator is the last of GELATI's three core components and corresponds to the last extension discussed in Section 3.4. The purpose of the analogy generator is to generate lexical analogies, by computing the analogousness between every pair of word-pairs and output the ones scoring higher than some threshold $K_{threshold}$. As discussed in Section 3.4, the analogousness between two word-pairs is largely determined by how many features they share — the more features they have in common, the higher their analogousness. However, computing analogousness solely by shared features does not yield an optimal result. The problem stems from the fact that features that are different are not necessarily semantically distinct. Many 'different' features may end up being mere surface variants of the same semantic content. As such, two word-pairs that have few features in common may still be highly relationally similar if a large number of their non-shared features are semantically identical or similar. In order to properly generate lexical analogies, therefore, the analogy generator must compensate for features that are only superficially different.

GELATI implements two analogy generators that use two fundamentally different approaches to solve this problem. The first generator operates within the framework of a Vector Space Model (VSM) [29], and is therefore called the VSM generator. The second generator involves collecting evidence about word-pairs' shared features, and is therefore called the Evidence Counting or EC generator. The following subsections describe each generator in detail.

Vector Space Model Generator

The VSM generator considers analogy generation as an application of the Vector Space Model (VSM), and uses familiar VSM similarity measures to compute analogousness. Specifically, for each word-pair, the generator creates a vector of dimensions equal to the total number of features extracted. Each dimension of the vector corresponds to a feature, and is set to one if that feature is a feature of the word-pair, or zero otherwise. As such, the vector concisely summarizes the feature structure of the word-pair, and in effect becomes a *signature* for the word-pair. These signature vectors provide the basis from which analogousness can be computed — the analogousness between two word-pairs is simply the degree of similarity between their signature vectors, with respect to some vector similarity measure.

The VSM generator uses one of the most common vector similarity measures: the cosine measure. Mathematically, the cosine measure CS between vectors v_1 and v_2 is defined as:

$$CS(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|_2 + \|v_2\|_2} \tag{4.1}$$

Geometrically, the cosine measure corresponds to the cosine of the angle between the two vectors, as Figure 4.4 depicts.

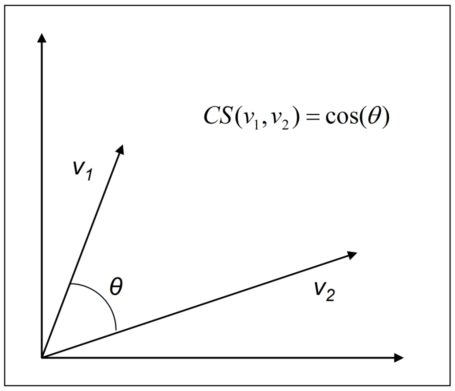


Figure 4.4: Graphical View of the Cosine Measure

Clearly, the cosine measure by itself does not compensate for superficial feature differences. To solve the problem, the VSM generator turns to a popular technique in Information Retrieval (IR): Latent Semantic Analysis (LSA) [17]. LSA attempts to solve a very similar problem in IR, namely, the relevancy between two documents cannot be entirely determined by the words they have in common because different words may be semantically equivalent. The solution that LSA proposes is to compress the feature space in such a way that the compressed space retains the most differentiating dimensions from the original feature space while merging less differentiating ones. The authors of LSA claim that this compressed space represents a semantic space in which surface differences are minimized. While the validity of this claim may be debatable, there is no doubt that LSA indeed performs remarkably well in solving the IR problem.

The success of LSA prompts the VSM generator to adopt the same approach. The mechanism that LSA uses to reduce the feature space's dimension is Singular Value Decomposition (SVD) [30], which is also what the VSM generator uses. SVD is a matrix operation that decomposes an m-by-n matrix M into a product of three matrices: an m-by-m unitary matrix U, an m-by-n diagonal matrix Σ that contains the singular values of M in its diagonal, and an n-by-n unitary matrix V. The remarkable property about SVD is that if Σ is rearranged in non-increasing order of singular values, and all three matrices are truncated to a smaller dimension t, then the resulting product M' is a rank t matrix that is the best approximation of M among *all* rank t matrices; in other words, optimal dimension reduction.

Putting this all together yields the following steps that the VSM generator takes to generate lexical analogies:

1. Compute the signature vectors of all word-pairs.
2. Concatenate all signature vectors to form a matrix representing the feature space.
3. Reduce the dimension of the feature space to K_{dim} using SVD.
4. Use the reduced vectors to compute pair-wise cosine measures.
5. Output all pairs scoring higher than a threshold $K_{threshold}$ as lexical analogies.

Evidence Counting Generator

The EC generator centres around the concept of *evidence features*. An *evidence feature* of two word-pairs is a feature that suggests the two word-pairs may be relationally similar. For example, a feature shared by both word-pairs is certainly an evidence feature. The EC generator computes the analogousness between two word-pairs by counting their evidence features and producing a score proportional to the count.

Precisely what features are considered evidence features? As mentioned, common features shared by two word-pairs certainly qualify. Using just the common features, however, causes the EC generator to be vulnerable to superficial feature differences. To avoid the problem, we consider a much larger set of features than just the common features, by taking advantage of *transitivity*. Consider, for example, two word-pairs wp_1 and wp_2 that are known to be relationally similar. Suppose wp_2 has a lexical analogue, wp_3. Then by way of transitivity, wp_3 is also likely a lexical analogue of wp_1 even if they do not share many features. Similarly, suppose f is a feature of wp_2. Then f is also likely a valid feature of wp_1 even if wp_1 never occurs with f in the input corpus. By using transitivity, f becomes an *indirect feature* of wp_1 that can be included in the computation of wp_1's analogousness.

The following definitions formalize the above intuition.

j-neighbour For a word-pair wp_1, the *0-neighbour* is the word-pair wp_1 itself, a *1-neighbour*, or simply *neighbour*, is a word-pair that shares some features with wp_1, and a *j-neighbour*, $j \geq 2$, is a neighbour of a $(j-1)$-neighbour of wp_1.

j-feature For a word-pair wp_1, a *level j feature*, or simply *j-feature*, $j \geq 0$, is a feature of a j-neighbour of wp_1.

j-evidence A *level j evidence feature* of two word-pairs wp_1 and wp_2 is a feature of wp_1 and a j-feature of wp_2. The set of all level j evidence features of wp_1 and wp_2 are collectively referred to as the j-evidence of the two word-pairs.

∞-evidence Given a maximum level K_{levels} to consider, the *∞-evidence* of two word-pairs wp_1 and wp_2 is the set of features of wp_1 that are *not* j-features of wp_2 for all $j \leq K_{levels}$.

Feature level captures the intuition that, in addition to direct features, a word-pair also has indirect features through transitivity. Evidence level captures the intuition that these indirect features can also contribute to the word-pair's relational similarity. Clearly, the 0-evidence of two word-pairs is precisely their set of common features. The rest of the definitions are best illustrated through an example. Consider the scenario depicted in Figure 4.5. In this example, there are five word-pairs labelled *Word-Pair 1* to *Word-Pair 5*, as well as four features labelled *Feature 1* to *Feature 4*. *Word-Pair 1* occurs with *Feature 1*, *Word Pair 2* with *Feature 1* and *Feature 2*, *Word-Pair 3* with *Feature 2* and *Feature 3*, *Word-Pair 4* with *Feature 3*, and *Word-Pair 5* with *Feature 4*.

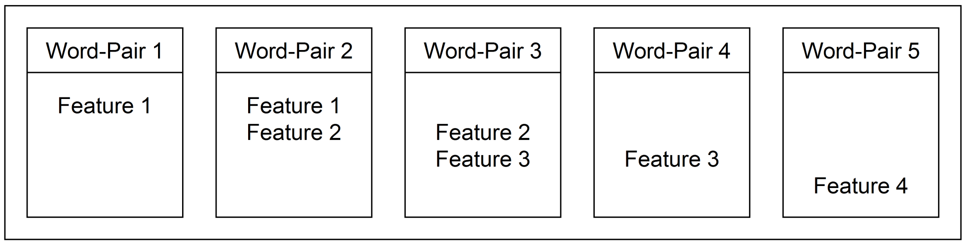


Figure 4.5: Example for Evidence Counting Generator

Some observations from the example:

1. Each word-pair is a 0-neighbour of itself.

2. *Word-Pair 1* is a 1-neighbour of *Word-Pair 2*, as they both share *Feature 1*.

3. Similarly, *Word-Pair 2* is a 1-neighbour of *Word-Pair 3* due to *Feature 2*.

4. Combining the above means that *Word-Pair 3* is a 2-neighbour of *Word-Pair 1*.

5. *Feature 1* is a 0-feature of *Word-Pair 1*.

6. *Feature 2* is a 1-feature of *Word-Pair 1*, due to *Word-Pair 2* being a 1-neighbour of *Word-Pair 1*.

7. *Feature 1* is a level 0 evidence feature of *Word-Pair 1* and *Word-Pair 2*, a level 1 evidence feature of *Word-Pair 1* and *Word-Pair 3*, and a level 2 evidence feature of *Word-Pair 1* and *Word-Pair 4*.

8. *Feature 1* is also an element of the ∞-evidence between *Word-Pair 1* and *Word-Pair 5*.

To merge the contributions of all evidence features, the EC generator linearly combines them by levels. Let $E(wp_1, wp_2, j)$ be the number of level j evidence features of wp_1 and wp_2, K_{levels} be the highest evidence level to consider, and $c_1, \ldots, c_{K_{levels}}, c_\infty$ be a set of weights. Then the analogousness A_{EC} between two word-pairs wp_1 and wp_2 with respect to the EC generator is:

$$\begin{aligned} A_{EC}(wp_1, wp_2) \;=\; & c_0 \cdot E(wp_1, wp_2, 0) + \ldots + \\ & c_{K_{levels}} \cdot E(wp_1, wp_2, K_{levels}) + c_\infty \cdot E(wp_1, wp_2, \infty) \end{aligned} \tag{4.2}$$

The weights are learned by training on a set of hand-labelled samples. In practice, the training set can be quite small due to the fact that typically only the first few levels of evidence are relevant, leaving only a few weights to learn. There is a wide variety of supervised machine learning techniques that can be used to learn the weights. The EC generator uses *linear least-squares regression* (LLR), mostly due to LLR's simplicity and generally good performance. LLR computes the weights by minimizing their square error when applied to the training set. Specifically, let $T = \{t_1, \ldots, t_k\}$ be the training set, in which each t_i is composed of two word-pairs $wp_{1_{t_i}}$ and $wp_{2_{t_i}}$. Let $B = \{b_1, \ldots, b_k\}$ be a set of values, with $b_i = 1$ if t_i's two word-pairs form a valid lexical analogy, and 0 otherwise. Let $\vec{c}$ be a vector corresponding to the weights $c_1, \ldots, c_{K_{levels}}, c_\infty$, and let $\vec{b}$ be a vector corresponding to B. Define a matrix A as follows:

$$A = \begin{bmatrix} E(wp1_{t_1}, wp2_{t_1}, 0) & \ldots & E(wp1_{t_1}, wp2_{t_1}, K_{levels}) & E(wp1_{t_1}, wp2_{t_1}, \infty) \\ E(wp1_{t_2}, wp2_{t_2}, 0) & \ldots & E(wp1_{t_2}, wp2_{t_2}, K_{levels}) & E(wp1_{t_2}, wp2_{t_2}, \infty) \\ & & \vdots & \\ & & \vdots & \\ E(wp1_{t_k}, wp2_{t_k}, 0) & \ldots & E(wp1_{t_k}, wp2_{t_k}, K_{levels}) & E(wp1_{t_k}, wp2_{t_k}, \infty) \end{bmatrix}$$

Then, the goal of LLR is to determine the value of $\vec{c}$ that minimizes the following:

$$\left\| A\vec{c} - \vec{b} \right\|_2$$

It turns out that the optimal value of $\vec{c}$ can be computed by the following [1]:

$$\vec{c} = (A^T A)^{-1} A^T \vec{b} \tag{4.3}$$

where A^T is the transpose of A. Using LLR, therefore, reduces the computation of optimal weights to a few simple matrix operations.

The analogousness measure as defined in Equation 4.2 properly compensates for superficial feature differences. However, it suffers from another limitation: the function gives an unfair advantage to word-pairs having many features. To solve this problem, the EC generator replaces the count of evidence features with the *ratio* between the number of evidence features and the number of total features. Obviously, using the ratio gives a slight advantage to word-pairs having very few features. Fortunately, the global extraction filters discussed in Section 4.6 ensure that this will not be a problem, as all word-pairs will have at least K_{filter_3} features. Reflecting this change, the analogousness measure now becomes:

$$\begin{aligned} A_{EC}(wp_1, wp_2) \quad = \quad & c_0 \cdot \tfrac{E(wp_1, wp_2, 0)}{d} + \ldots + \\ & c_{K_{levels}} \cdot \tfrac{E(wp_1, wp_2, K_{levels})}{d} + c_\infty \cdot \tfrac{E(wp_1, wp_2, \infty)}{d} \end{aligned} \tag{4.4}$$

where d is the number of features of wp_1.

As A_{EC} depends on the number of features of wp_1, it is no longer symmetric: the analogousness of (wp_1, wp_2) is different from the analogousness of (wp_2, wp_1) unless the two word-pairs have the same number of features. The EC generator's solution is to

[1]See http://en.wikipedia.org/wiki/Linear_least_squares for an example of the derivation.

maintain a canonical parameter ordering by reversing the parameters if the second word-pair has more features than the first word-pair. Taking this last change into account, the final analogousness measure with respect to the EC generator is as follows:

$$A_{EC}(wp_1, wp_2) = \left\{ \begin{array}{ll} \hat{A}_{EC}(wp_2, wp_1) & \text{if } wp_2 \text{ has more features than } wp_1 \\ \hat{A}_{EC}(wp_1, wp_2) & \text{otherwise} \end{array} \right\} \tag{4.5}$$

$$\begin{aligned} \hat{A}_{EC}(wp_1, wp_2) &= c_0 \cdot \tfrac{E(wp_1, wp_2, 0)}{d} + \ldots + \\ & \quad c_{K_{levels}} \cdot \tfrac{E(wp_1, wp_2, K_{levels})}{d} + c_\infty \cdot \tfrac{E(wp_1, wp_2, \infty)}{d} \end{aligned} \tag{4.6}$$

4.8 Analogy Filter

The last component in GELATI's pipeline is the analogy filter, which filters out inappropriate lexical analogies generated by the analogy generator. The filter specifically performs the following.

1. Eliminate all lexical analogies of the form *A:B::A:C* or *B:A::C:A*. For example, one such lexical analogy is *rain:snow::rain:frost*. These lexical analogies, while generally valid, really capture the near-synonymy relation between B and C more than the analogy between (A, B) and (A, C).

2. Eliminate all but one *permutations* of each lexical analogy. A lexical analogy can appear as four different permutations of its constituent words: *A:B::C:D*, *B:A::D:C*, *C:D::A:B*, *D:C::B:A*. Obviously, all four permutations describe the same lexical analogy, hence only one needs to be kept. The filter does not enforce a canonical permutation, so whichever is generated first is the one that will be kept.

Chapter 5

Experimental Evaluation

In this chapter we present the results of an experimental evaluation of GELATI. Section 5.1 describes the experimental setup and evaluation protocol. Section 5.2 summarizes the results. Lastly, Section 5.3 discusses some interesting observations as well as key issues of GELATI as revealed by the experiment.

5.1 Experimental Setup

The experimental evaluation of GELATI consisted of the following five steps:

1. Collect a large corpus of text documents to be used as the input data.
2. Set GELATI's parameters to appropriate values.
3. Run GELATI on the corpus using the VSM generator.
4. Run GELATI on the corpus using the EC generator.
5. Evaluate the validity and quality of the generated lexical analogies.

Corpus

The input for the experiment was composed of a subset of the text corpus used by the **T**ext **RE**trieval **C**onference (TREC)[1]. TREC is a conference and competition on various aspects of text retrieval sponsored by the United States government. The corpus that TREC uses originates from two sources: the Linguistic Data Consortium[2], and the NIST

[1]http://trec.nist.gov/
[2]http://www.ldc.upenn.edu/

Standard Reference Data[3]. The corpus contains a variety of general news articles as well as domain-specific documents, organized into collections based on the publication source of each document. All documents are encoded in SGML, and all contain various meta-data in addition to the actual text content. For the experiment, the following collections were selected: AP Newswire 1988–1990, LA Times 1989–1990, and San Jose Mercury 1991. Table 5.1 lists the size of each collection, both in its original SGML format and after extracting text content.

Track	SGML Size	Extracted Size
AP Newswire	728 MB	627 MB
LA Times	475 MB	359 MB
San Jose Mercury	286 MB	210 MB
Total	1489 MB	1196 MB

Table 5.1: Experiment Corpus Statistics

Parameter Values

The parameter values of GELATI were determined largely through trial-and-error on smaller corpora. The filter parameter values were additionally influenced by the need to restrict the number of word-pairs and features in order to allow the computationally intense analogy generators to complete within a reasonable time. The maximum evidence level of the EC generator was set to one, simply because most features turned out to be either in 0-evidence or 1-evidence. The final parameter values for the experiment are summarized in Table 5.2.

Evaluation Protocol

Unfortunately, devising an objective measure to evaluate GELATI is very difficult due to the lack of a reference dictionary of lexical analogies and that lexical analogies are, by definition, subjective. As such, we elected to use a subjective measure to evaluate GELATI's output — asking human judges to grade the lexical analogies GELATI generated. The evaluation process involved three human participants, all of whom are proficient English speakers with at least a Master's degree in Arts, and none of whom had any connection to this research prior to the experiment. Each participant was given a survey containing

[3]http://www.nist.gov/srd/index.html

Parameter	VSM Gen	EC Gen	Explanation
K_{filter_1}	50	50	filter: min word-pair occurrences
K_{filter_2}	50	50	filter: min feature occurrences
K_{filter_3}	40	40	filter: min word-pair unique occurrences
K_{filter_4}	∞	∞	filter: max word-pair unique occurrences
K_{filter_5}	10	10	filter: min feature unique occurrences
K_{filter_6}	100	100	filter: max feature unique occurrences
K_{dim}	400	N/A	VSM generator: reduced dimension
K_{levels}	N/A	1	EC generator: maximum evidence level
$K_{threshold}$	0.53	0.47	analogousness threshold for output

Table 5.2: Experiment Parameter Values

a list of lexical analogies, and was asked to grade each lexical analogy with a score from zero to 10, with zero denoting an invalid lexical analogy and 10 denoting a perfect lexical analogy. To ensure at least a rudimentary level of consistency, each participant was also given detailed instructions as well as examples of lexical analogies. The exact instructions given to each participant are listed in Appendix A.

Each survey was unique to the participant to whom the survey was given, and was generated using the following procedure:

1. All lexical analogies generated by the VSM generator were ranked by their analogousness score and divided into 10 partitions.

2. Similarly, all lexical analogies generated by the EC generator were ranked and divided into 10 partitions.

3. Four lexical analogies were randomly drawn from each of the 20 partitions, resulting in 80 lexical analogies which were included in the survey in random order.

In addition, 10 lexical analogies drawn from real or practice SAT verbal analogy questions were randomly inserted into each survey. These SAT lexical analogies, which are listed in Table 5.3, acted as a control set of the best manually generated lexical analogies. Each survey therefore contained 90 lexical analogies in total.

After all the surveys were completed, the results were analyzed using the following two metrics:

1. *Precision*: the percentage of *valid* lexical analogies among all that were graded, where valid means a score greater than zero.

(mason, stone) and (carpenter, wood)
(amplifier, ear) and (telescope, eye)
(devotion, obsession) and (confidence,conceit)
(swarm, bee) and (school, fish)
(body guard, person) and (soldier, country)
(archeology, science) and (chair, furniture)
(dalmatian, dog) and (oriole, bird)
(cub, bear) and (puppy, dog)
(saw, carpenter) and (scissors, tailor)
(census, population) and (inventory, merchandise)

Table 5.3: Lexical Analogies in SAT Control Set

2. *Quality*: the average score of all graded lexical analogies.

An important metric that was *not* included in our analysis is *recall*: the number of valid lexical analogies generated divided by the number of all valid lexical analogies that could potentially be generated from the corpus. Recall was omitted because there is no simple method to even estimate the number of lexical analogies that could potentially be generated — the corpus is much too large to be analyzed manually.

5.2 Experimental Results

The experiment resulted in 6,916 word-pairs and 9,010 features after filtering. The VSM generator generated 2,097 lexical analogies, while the EC generator generated 2,207 lexical analogies. The EC generator was trained using 20 hand-labelled samples, 10 positive and 10 negative. The trained weights are shown in Table 5.4. These values, however, are slightly misleading. 1-evidence is much more influencing than the weight suggests, due to the fact that there are far more level 1 evidence features than other levels. Similarly, ∞-evidence is not nearly as influencing as it appears to be. In the end, 0-evidence has the highest influence, followed by 1-evidence then ∞-evidence.

Performance

Table 5.5 lists the running times of the major processes of GELATI, based on a Java implementation running on a 2.5 gigahertz processor. For this experiment, the SVD

Weight	Value
c_0	2.88708
c_1	0.09236
c_∞	-3.75070

Table 5.4: EC Generator Weights After Training

computation was conducted externally in MATLAB[4] on an Intel Itanium2 processor. Unfortunately, as MATLAB does not natively support Itanium2, it was forced to run in emulation mode which was very slow. Furthermore, a large portion of the computation time was actually spent in the importation and exportation of the data matrix. The reported running time therefore does not truly reflect the performance of the VSM generator. Nevertheless, even without SVD the VSM generator is slightly slower than the EC generator. Clearly, by far the most expensive operation in GELATI is parsing in the preprocessor. A faster dependency parser would significantly improve GELATI's performance.

Process	VSM Gen	EC Gen
Preprocessor: Text Extraction	37 min	
Preprocessor: Segmentation	20 min	
Preprocessor: Parsing	2232 min	
Extraction	126 min	
SVD Computation	208 min	N/A
Analogy Generation	88 min	83 min

Table 5.5: GELATI Running Times

Precision Analysis

Table 5.6 lists the average precision scores of the VSM generator, the EC generator, and the SAT control set. Figure 5.1 compares these scores graphically.

GELATI performs remarkably well with respect to precision, approaching the level of the SAT control set which unsurprisingly has a precision of 1.00. The high precision scores mean that most of the lexical analogies that GELATI generates are valid. The VSM generator performs slightly better than the EC generator, but the difference is not

[4]http://www.mathworks.co.uk/products/matlab/

Source	Precision Score
VSM Generator	0.93
EC Generator	0.92
SAT Control	1.00

Table 5.6: Precision Scores

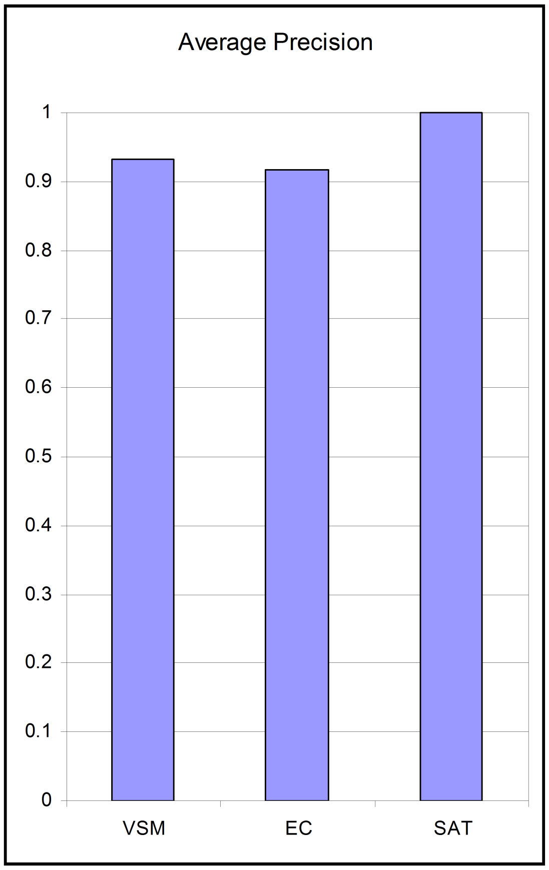


Figure 5.1: Precision Scores Chart

large enough to be statistically significant. Table 5.7 breaks down the precision scores to individual partitions, and Figure 5.2 compares them graphically.

Partition	VSM Gen	EC Gen
1	1.00	1.00
2	1.00	0.92
3	1.00	0.83
4	0.92	0.83
5	0.83	1.00
6	0.83	0.92
7	0.92	0.83
8	1.00	0.92
9	0.92	0.92
10	0.92	1.00

Table 5.7: Partition-wise Precision Scores

As Table 5.7 shows, the precision score does not appear to decline toward higher partitions. There are a few possible explanations. First, the analogousness measures may be too coarse to support ranking. In other words, while a high analogousness score is a good indication of validity, a slightly higher analogousness score is not a good indication of better relational similarity. Second, the selected threshold values may be too high. As a result, the precision has not yet started to decline before the threshold value is reached. Finally, there may be sampling errors due to the small number of samples used.

Quality Analysis

The quality scores of the VSM generator, the EC generator, and the SAT control set are listed in Table 5.8 and illustrated in Figure 5.3.

Source	Quality Score
VSM Generator	6.18
EC Generator	6.23
SAT Control	9.33

Table 5.8: Quality Scores

Unlike the precision scores, the quality scores of GELATI's lexical analogies are respectable but significantly lower than that of the SAT control set. The EC generator

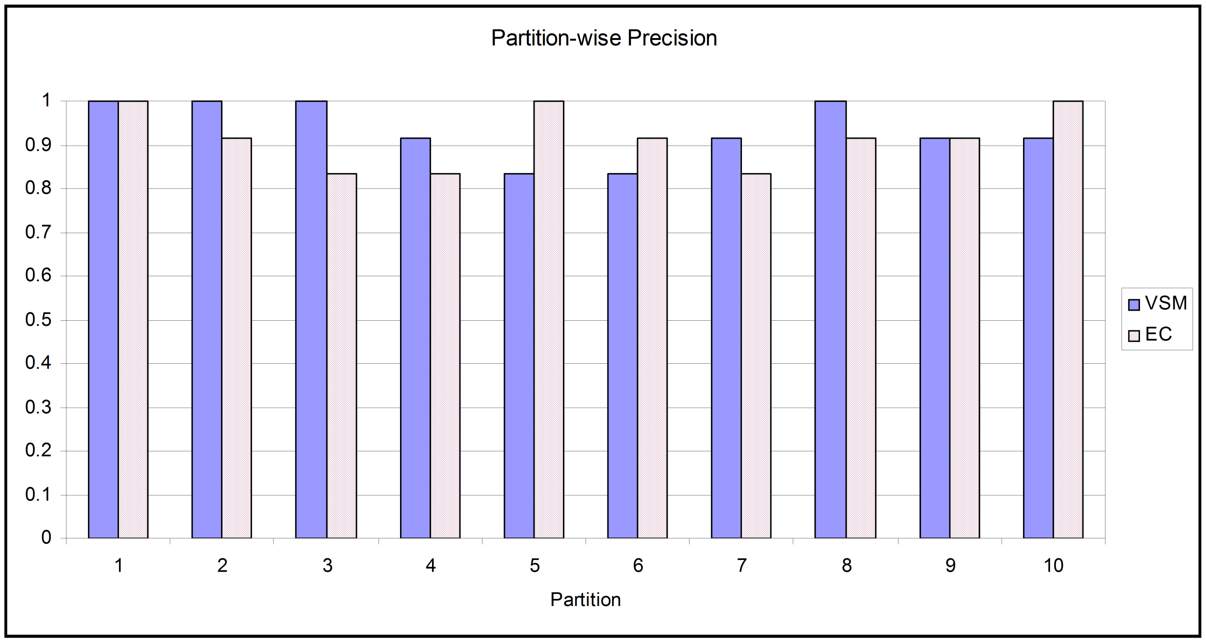


Figure 5.2: Partition-wise Precision Scores Chart

appears to perform slightly better than the VSM generator, however the difference is again too small to be statistically significant. Table 5.9 breaks down the quality scores to individual partitions, and Figure 5.4 compares them graphically. As with the precision scores, quality scores do not appear to decline toward higher partitions.

Partition	VSM Gen	EC Gen
1	7.25	7.00
2	6.58	5.83
3	5.75	5.75
4	6.50	5.75
5	6.42	6.50
6	4.75	6.58
7	5.75	4.33
8	6.67	7.00
9	6.17	5.75
10	5.92	7.75

Table 5.9: Partition-wise Quality Scores

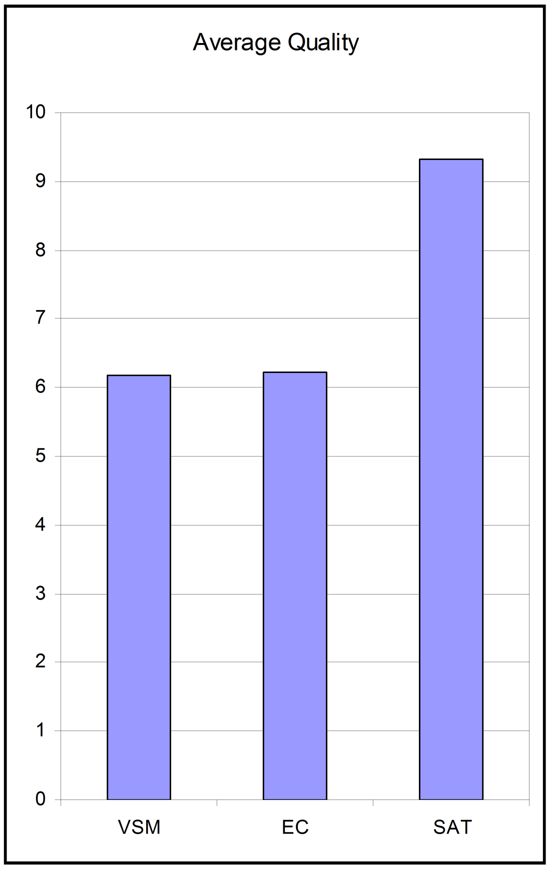


Figure 5.3: Quality Scores Chart

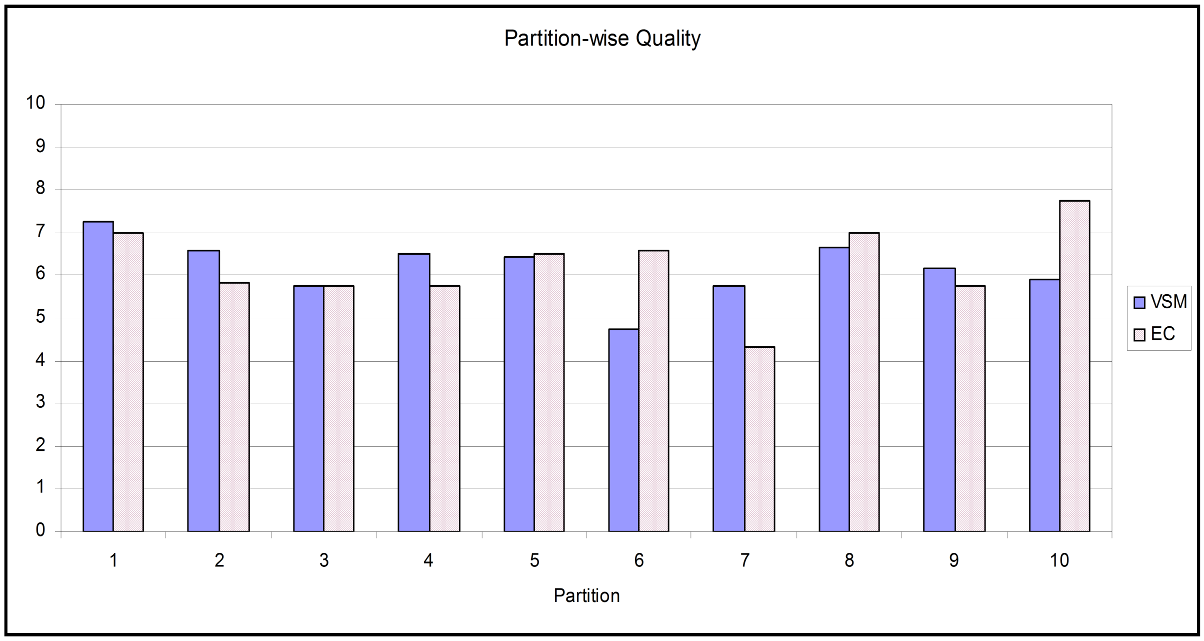


Figure 5.4: Partition-wise Quality Scores Chart

Sample Output

Table 5.10 lists the top 30 lexical analogies generated by the VSM generator, while Table 5.11 lists the top 30 by the EC generator. The tables are exactly what GELATI produced without any editing. The format of each lexical analogy is as follows: rank of the analogy, identification number of the first word-pair, the first word-pair, identification number of the second word-pair, the second word-pair, the analogousness score. In addition, Table 5.12 shows a manually selected list of some of the best and most creative lexical analogies generated by GELATI. The shared relations shown in the table are manually labelled.

5.3 Discussions

Noise

Despite the rather high precision score, the moderate quality score means that GELATI still produced a fair amount of completely invalid or very poor lexical analogies, which are collectively referred to as *noise*. Noise is contributed by almost every part of GELATI, starting from the input corpus. The input data is surprisingly noisy, containing many fragmented sentences, isolated words, and other language errors. The corpus is also noisy

1 : 2393 (stock market,trading) AND 4092 (price,session) — 0.9109935656865431
2 : 2451 (stocks,trading) AND 4092 (price,session) — 0.9039325577308122
3 : 4093 (session,price) AND 5645 (trading,stock) — 0.8877049519788612
4 : 1999 (dollar,trading) AND 4092 (price,session) — 0.8817705036576743
5 : 338 (soldier,palestinian) AND 5496 (troops,people) — 0.8487917204466584
6 : 984 (apartment,police) AND 3464 (home,officer) — 0.8457190323552611
7 : 2182 (force,withdrawal) AND 3045 (hostage,release) — 0.8454509782234105
8 : 1867 (bill,legislature) AND 3016 (legislation,house) — 0.8452682765360523
9 : 1494 (u.s.,judge) AND 3325 (attorney,general) — 0.8425441590865859
10 : 310 (increase,rise) AND 4596 (decline,drop) — 0.8376257126089509
11 : 1777 (inflation,rate) AND 6578 (sales,level) — 0.8360240039988636
12 : 3016 (legislation,house) AND 3132 (bill,assembly) — 0.8274124593118469
13 : 2708 (house,police) AND 3464 (home,officer) — 0.8263104003140124
14 : 576 (newspaper,article) AND 1498 (magazine,story) — 0.8254199555013249
15 : 2069 (senate,measure) AND 3015 (house,legislation) — 0.8186809947338843
16 : 1715 (legislation,congress) AND 1867 (bill,legislature) — 0.8169632812325749
17 : 338 (soldier,palestinian) AND 3064 (police,worker) — 0.8162788761713085
18 : 2038 (senate,legislation) AND 3028 (house,measure) — 0.8156125414678665
19 : 310 (increase,rise) AND 4597 (drop,decline) — 0.8153169661545098
20 : 4142 (moscow,gorbachev) AND 4225 (washington,bush) — 0.8116149384985234
21 : 1867 (bill,legislature) AND 2674 (measure,congress) — 0.8084539172519039
22 : 184 (house,resolution) AND 2038 (senate,legislation) — 0.8076013194220543
23 : 984 (apartment,police) AND 3141 (house,officer) — 0.8073661078829227
24 : 984 (apartment,police) AND 6150 (home,agent) — 0.8068929753013839
25 : 1570 (interest rate,market) AND 5644 (stock,trading) — 0.8038515556945531
26 : 338 (soldier,palestinian) AND 704 (police,student) — 0.8037038701966855
27 : 338 (soldier,palestinian) AND 703 (police,protester) — 0.8015985866695314
28 : 3028 (house,measure) AND 3133 (assembly,bill) — 0.7995749039521474
29 : 1777 (inflation,rate) AND 2348 (price,level) — 0.7994595492145963
30 : 1284 (trading,price) AND 1571 (market,interest rate) — 0.7991148497556326

Table 5.10: Top-30 Lexical Analogies using the VSM Generator

1 : 2393 (stock market,trading) AND 4092 (price,session) — 1.4588876478294364

2 : 1999 (dollar,trading) AND 4092 (price,session) — 1.2467015623798323

3 : 2069 (senate,measure) AND 3015 (house,legislation) — 1.2330627119313111

4 : 2037 (legislation,senate) AND 3027 (measure,house) — 1.2200546855067245

5 : 576 (newspaper,article) AND 1498 (magazine,story) — 1.1683151677963508

6 : 2262 (president,ceremony) AND 6350 (bush,session) — 1.1403805236561322

7 : 302 (shareholder,stock) AND 6641 (investor,share) — 1.1219941786136878

8 : 1867 (bill,legislature) AND 3016 (legislation,house) — 1.11899232636186

9 : 3097 (court of appeals,ruling) AND 3241 (appeals court,decision) — 1.1086222913100916

10 : 3097 (court of appeals,ruling) AND 4583 (u.s. supreme court,decision) — 1.0984596569593588

11 : 4141 (gorbachev,moscow) AND 4224 (bush,washington) — 1.0984596569593588

12 : 310 (increase,rise) AND 4596 (decline,drop) — 1.0644369245677747

13 : 91 (attorney,lawsuit) AND 5000 (company,suit) — 1.056056941220094

14 : 2866 (people,fighting) AND 4422 (palestinian,uprising) — 1.0239336717206506

15 : 1867 (bill,legislature) AND 2037 (legislation,senate) — 1.0217306177575771

16 : 2069 (senate,measure) AND 3133 (assembly,bill) — 1.0049219407924088

17 : 3015 (house,legislation) AND 3133 (assembly,bill) — 1.0049219407924088

18 : 983 (police,apartment) AND 3140 (officer,house) — 0.996534412441714

19 : 983 (police,apartment) AND 3463 (officer,home) — 0.996534412441714

20 : 665 (official,hearing) AND 3101 (president,news conference) — 0.9938828712211717

21 : 2220 (satellite,orbit) AND 2908 (fund,account) — 0.9866706791012968

22 : 3111 (supreme court,ruling) AND 4583 (u.s. supreme court,decision) — 0.9707008251215736

23 : 5000 (company,suit) AND 6777 (city,lawsuit) — 0.9657102457529101

24 : 183 (resolution,house) AND 3132 (bill,assembly) — 0.9463024370969966

25 : 310 (increase,rise) AND 4597 (drop,decline) — 0.9429271660264029

26 : 667 (law,legislature) AND 5471 (ordinance,city council) — 0.9392450521312099

27 : 1272 (effort,economy) AND 6627 (plan,system) — 0.9360492551655706

28 : 3027 (measure,house) AND 3132 (bill,assembly) — 0.9258731648276138

29 : 1570 (interest rate,market) AND 4092 (price,session) — 0.9143366346049038

30 : 1867 (bill,legislature) AND 3607 (law,parliament) — 0.910326986905547

Table 5.11: Top-30 Lexical Analogies from EC Generator

Lexical Analogy	Shared Relation
(force, withdrawal) and (hostage, release)	safely-returns
(increase, rise) and (decline, drop)	synonymy
(moscow, gorbachev) and (washington, bush)	led-by
(satellite, orbit) and (fund, account)	resides-in
(law, legislature) and (ordinance, council)	approved-by
(approval, agreement) and (passage, bill)	legitimizes
(investigation, arrest) and (debate, vote)	leads-to
(newspaper, article) and (magazine, story)	publishes

Table 5.12: Examples of Good Lexical Analogies Generated by GELATI

in the sense that it contains tables and figures represented in text form, which ordinary parsers such as MINIPAR are unable to handle. MINIPAR itself is also far from perfect. Our experience shows that MINIPAR has a precision of between 80–90% on general news articles. However, MINIPAR's performance significantly degrades for sentences that are either very long or contain complex subclauses. The extraction filters also contribute to noise due to the fact that they use rather coarse measures to estimate word-pair and feature relevancy. As a result, some word-pairs with rather weak relations such as (*officer*, *house*) still manage to pass through the filters. Last but definitely not least, ultimately dependency patterns are syntactic constructs and really only approximate semantic relations. Two word-pairs sharing many dependency patterns are guaranteed to be syntactically similar, but they are only *likely* to be semantically similar. This approximation inevitably leads to some errors, which eventually become noise.

Polysemic Words

Many words in GELATI's output involve multiple meanings, some of which are very specific to the context in which they occur. As a result, many lexical analogies produced by GELATI are difficult to interpret unless they are considered in very specific contexts. Consider the lexical analogy *legislation:senate::measure:house*, for example. This lexical analogy appears to be completely invalid if each word is interpreted as its most common sense. However, these words actually come from political articles on law-making. In that context, *measure* actually refers to a *measure of law*, and *house* actually refers to the *House of Commons*, hence the lexical analogy is in fact valid.

There are at least three areas that need to be extended in order to properly resolve this issue. First, obviously, word-sense disambiguation must be performed on the input. Secondly, referring expressions must be resolved, so that GELATI is able to recognize, for example, that "*the House*" actually refers to "*the House of Commons*". Lastly, all processing throughout GELATI's pipeline must be performed on sense-pairs instead of word-pairs, and the final output must be represented in a format that shows word senses instead of words.

Granularity

GELATI's dependency on syntactic features results in another limitation. The problem is that GELATI cannot differentiate between concepts at different *granularities*. Consider the lexical analogy *country:relation::united states:ties* for example. This lexical analogy is valid because a country has relations with other countries just as the United States has ties with other countries. On the other hand, it is not very satisfying because the two word-pairs are at two different levels of granularity. The first word-pair involves countries in general, while the second word-pair involves a specific country, namely the United States. GELATI fails to distinguish this difference because in most cases both word-pairs participate in similar syntactic constructs.

Analogy Clusters

A manual scan of the lexical analogies generated by GELATI revealed another observation: the output often contains what we refer to as *analogy clusters*. Essentially, an analogy cluster involves multiple lexical analogies that all share similar words. Usually they are caused by different combinations of two sets of synonyms. For example, consider the sets {*senate*, *assembly*, *legislature*} and {*bill*, *measure*, *legislation*}. The first set describes law-making bodies, and the second set describes laws. Taking a word from the first set and another from the second set, there is a total of nine possible word-pairs that can be formed. Using those nine word-pairs, there are $\binom{9}{2} = 36$ lexical analogies that can be formed, all sharing the same underlying relation and similar words, yet all distinct.

Chapter 6

Conclusion

6.1 Summary

In this thesis we have presented GELATI, a system that generates lexical analogies from a corpus of text documents. GELATI divides the task of lexical analogy generation into three main processes: identify semantically related word-pairs, extract features to characterize their underlying relations, and match word-pairs with similar features to generate lexical analogies. GELATI uses dependency structures to identify semantically related word-pairs and to characterize their underlying relations, and machine learning techniques to compute their relational similarity. Experimental evaluation shows that over 90% of the lexical analogies GELATI generates are valid, although their quality is not as satisfying as the best lexical analogies generated by humans.

Going forward, there are a number of possible future directions for improvement and extension, as discussed in the following sections.

6.2 Future Directions

Alternative Corpus

The corpus we used for the experimental evaluation consisted entirely of news articles, which may not be ideal for the purpose of lexical analogy generation. A future direction, therefore, is to experiment with different genres of text. In particular, poems, novels, and other creative writings may be good candidates as they tend to use metaphors and analogies more freely and more creatively than other genres.

Additional Features

In addition to dependency patterns, there are many other syntactic and semantic features that could assist GELATI in characterizing underlying relations. The following is a list of some possibilities. For a word-pair (w_1, w_2):

1. Words that commonly occur in sentences involving both w_1 and w_2, subject to a list of stop words such as determiners.

2. Common words in the definitions of w_1 and w_2 in an electronic lexical resource, again subject to a list of stop words. Two particular resources that may be useful are WordNet [4] and Wikipedia[1].

3. Patterns based on frames, semantic role labels, or other semantic structures in sentences involving both w_1 and w_2. These features would be very similar to dependency patterns, except they operate at the semantic level rather than the syntactic level. An unfortunate limitation of these features is that they are difficult to obtain — current semantic parsers are nowhere near as accurate and efficient as current syntactic parsers.

Semantic features are particularly useful, as they allow underlying relations to be differentiated at a finer level. Note that because the analogy generator does not depend on the type of features used, all of these features, as well as dependency patterns, could be used together in a heterogeneous feature set.

Clustering Algorithms for Analogy Generation

The VSM and the EC generators illustrate two possible approaches to generating lexical analogies from word-pairs and features. Certainly, there are other possibilities. A particularly interesting class of algorithms is clustering algorithms. In essence, relational similarity is no different from attributional similarity, except different features are used. As such, algorithms that have been shown to be successful for attributional similarity, such as clustering algorithms, are logical candidates for relational similarity. Clustering algorithms also make sense on a theoretical level: after all, lexical analogies are really nothing more than clusters of word-pairs sharing similar features.

[1] www.wikipedia.com

```
1.  repeat
2.    pick a random subset $S$ of $E$ with $|S| \geq 2$
3.    for each element in $S$, obtain its set of analogues from $A$
4.    take the intersection $T$ of all the above sets
5.    add all elements in $T$ to $E$
6.  until all possible subsets have been tried
```

Table 6.1: Algorithm for Learning Ontological Relations

6.3 Learning Ontological Relations

An exciting application of GELATI, and in fact what originally inspired this research, is to use the lexical analogies generated by GELATI to learn ontological relations. As briefly discussed in Section 1.4, manually constructing ontologies is an expensive process, so there is a growing need for methods to automate the process. Ontological relations present a particularly difficult challenge due to the sheer number and variety of relations that can, and need, to be captured. Most ontologies contain only what Morris and Hirst [5] call *classical relations*: WordNet relations such as synonymy and hyponomy. However, the majority of relations in real-world data are non-classical — for example, *positive-qualities* (*humbleness* and *kindness*), *cause-of* (*alcohol* and *drunk*), and *founder-of* (*Gate* and *Microsoft*) [5]. Methods are needed to automatically enrich ontologies with both classical and non-classical relations.

We propose that GELATI could be a key component in facilitating the process of learning ontological relations. The main observation that justifies this proposal is that, to a large extent, the underlying relations that GELATI learns are precisely the ontological relations that ontologies need to acquire. Although GELATI does not explicitly learn what these underlying relations *are*, it does offer a method to identify when they are *similar*. This identification of similarity can be a valuable resource for relation learning. As an example, Table 6.1 illustrates an algorithm that uses GELATI's output to learn instances of a given ontological relation.

The algorithm takes two inputs: a dictionary A of many lexical analogies generated by GELATI, and a small sample set E of instances of the ontological relation of interest. For example, to use the algorithm to learn instances of the *produces* relation, one would give as input (*poet*, *poem*), (*composer*, *music*), (*painter*. *painting*), and so on. The algorithm works on the very simple principle that if (w_1, w_2) and (w_3, w_4) form a lexical analogy, and w_1 and w_2 are related by some relation r, then likely w_3 and w_4 are related by the same relation r. Using this principle, the algorithm bootstraps from a small sample set, and increasingly adds to the sample set by looking for lexical analogues in A that are common to at least two samples. In essence, the algorithm uses lexical analogies as bridges through which the relation from the samples are spread to other word-pairs.

Clearly, this algorithm is only a prototype and lacks some necessary details. For example, the algorithm does not provide a method to map words to and from ontological concepts, which is non-trivial, as a word can map to multiple concepts (polysemy) and a concept can map to multiple words (synonymy). The algorithm also does not specify how the initial sample set can be generated. Nevertheless, the algorithm clearly shows that lexical analogies, and hence GELATI, *can* indeed act as an important component in learning ontological relations.

Appendix A

Instructions for Human Judges

Background, Definitions, and Notations

We define the *semantic relations* between two words to be the relations that link their meanings. For example, a semantic relation between *doctor* and *hospital* is *works-in* (i.e. a doctor *works in* a hospital), and a semantic relation between *poet* and *poem* is *writes* (i.e. a poet *writes* poems). Often there are multiple semantic relations between two words. Using *poet* and *poem* again, the two words are also related by *produces*, *enjoys*, *studies*, *understands*, *earns-money-with*, and so on. We call a pair of word a *word-pair*, and we call the semantic relations between the two words the word-pair's *underlying relations*. For example, the underlying relations of (*poet*, *poem*) are *writes*, *produces*, *enjoys*, etc.

A *lexical analogy* is formed by two word-pairs that share one or more underlying relations. In other words, if (A, B) and (C, D) form a lexical analogy, there is a common semantic relation between (A, B), and between (C, D). For example, (*poet*, *poem*) and (*painter*, k) form a lexical analogy because a poet produces poems just as a painter produces paintings. Similarly, (*abbreviation*, *word*) and (*abstract*, *report*) form a lexical analogy because in both word-pairs, the former is a shortened version of the later. Other examples of lexical analogies include (*cub*, *bear*) and (*puppy*, *dog*), (*newspaper*, *article*) and (*magazine*, *story*), and (*increase*, *rise)* and (*decrease*, *drop*).

Some lexical analogies are "better" than others. Broadly speaking, lexical analogies that involve clear and specific underlying relations are more satisfying than those that involve obscure or overly general underlying relations. For example, (*abbreviation*, *word*) and (*abstract*, *report*) form a good lexical analogy because the common underlying relation (*shortened-version-of*) is clear and specific. On the other hand, (*company*, *right*) and (*city*, *property*) form a poor lexical analogy because the common underlying relation (*has* or *entitled-to*) is very general. Of course, many word-pairs do not form lexical analogies at all. For example, (*interest rate*, *market*) and (*people*, *police*) do not form a lexical analogy because they do not share any meaningful underlying relations. We sometimes use the term *invalid* to refer to two word-pairs that do not form a lexical analogy at all.

What You Need To Do

In the following pages, you will be presented with a survey containing ninety entries. Each entry consists of two word-pairs. Your job is to determine if the two word-pairs form a lexical analogy, and if so, how good of a lexical analogy it is. You are to score each entry with a score between 0 and 10, where 0 denotes that the two word-pairs do not form a lexical analogy at all, and 10 denotes that they form a perfect lexical analogy. Obviously, the scoring is subjective, which is ok as long as you remain consistent throughout the survey. Note that lexical analogies are not always immediately obvious — sometimes it takes some creative thinking to arrive at the common underlying relation. Please take the time to consider each entry carefully and thoroughly before scoring it. As the process of scoring may become tedious and monotonic, we strongly recommend that you do not fill out the entire survey at once. Instead, score a few entries at a time and take necessary breaks. In total, we expect the survey to last no more than two hours, including the time to read these instructions.

Notes about Scoring

1. When a word-pair involves a polysemic word (i.e. a word with multiple meanings), choose the meaning that makes the best possible lexical analogy. For example, the word-pairs (*bill*, *legislature*) and (*legislation*, *house*) do not form a lexical analogy at all if *bill* is taken to mean a dollar bill and *house* is taken to mean a physical building. However, they form a good lexical analogy if *bill* is taken to mean a bill of law and *house* is taken to mean the House of Commons. Please make sure you consider all possibilities.

2. Best lexical analogies happen between word-pairs at the same granularity. For example, (*country*, *relation*) forms a valid lexical analogy with (*united states*, *tie*), because a country has relations with other counties just like the United States has

ties with other countries. However, this lexical analogy is not very good because of the fact that the first word-pair is at a higher level of granularity (country in general) than the second word-pair (a particular country).

3. The order of the two words in a word-pair matters! Most underlying relations only go in one way. For example, while *writes* is an underlying relation of (*poet*, *poem*), it certainly is NOT an underlying relation of (*poem*, *poet*). As such, (*poet*, *poem*) and (*painter*, *painting*) form a lexical analogy, but not (*poem*, *poet*) and (*painter*, *painting*). Having said this, some relations, such as synonymy (*same-as*), do work both ways, in which case the order does not matter.

4. Please feel free to consult a dictionary for words you are not familiar with. However, please do not consult another person.

5. Again, please do *not* rush through the survey. Please consider each entry carefully and thoroughly, and take necessary breaks instead of filling up the entire survey at once. It is best if you take several days to complete the survey.

Logistics

1. Only the aggregated result from all participants of this experiment will be used in publications. Your personal information will be kept to the strictest confidence and will never be revealed in any way whatsoever.

2. Please feel free to email Andy at any time should you have any questions or concerns.

3. **Please email the completed survey to Andy no later than Tue, Oct 3rd, 2006.**

Thank you very much for your participation!

Bibliography

[1] Hugh Petrie and Rebecca Oshlag. Metaphor and learning. In Andrew Ortony, editor, *Metaphor and Thought*, pages 579–609. Cambridge University Press, second edition, 1993.

[2] George Lakoff and Mark Johnson. *Metaphors We Live By.* University of Chicago Press, first edition, 1980.

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

[4] Christine Fellbaum, editor. *WordNet – An Electronic Lexical Database.* MIT Press, 1998.

[5] Jane Morris and Graeme Hirst. Non-classical lexical semantic relations. In *Proceedings of HTL-NAACL Workshop on Computational Lexical Semantics*, Boston, U.S.A., 2004.

[6] Dong Zhen Dong. What, how and who? In *Proceedings of the International Symposium on Electronic Dictionaries*, Tokyo, Japan, 1988.

[7] Rogers Hall. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120, 1989.

[8] Robert French. The computational modeling of analogy-making. *Trends in Cognitive Scinces*, 6(5):200–205, 2002.

[9] T. G. Evans. A program for the solution of a class of geometric analogy intelligence test questions. In M Minsky, editor, *Semantic Information Processing*, pages 271–353. MIT Press, 1968.

[10] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170, 1983.

[11] B. Falkenhainer, K.D. Forbus, and D. Gentner. The structure-mapping engine. *Artificial Intelligence*, 41:1–63, 1989.

[12] Samuel Butcher. *Aristotle's Poetics.* Hill and Wang, New York, 1961.

[13] Tony Veale. WordNet sits the S.A.T.: A knowledge-based approach to lexical analogy. In *Proceedings of the 16th European Conference on Artificial Intelligence*, Valencia, Spain, 2004.

[14] Douglas S. Blank. *Learning to See Analogies: A Connectionist Exploration.* PhD thesis, Indiana University, 1997.

[15] Peter Turney, Michael 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*, pages 482–489, Borovets, Bulgaria, 2003.

[16] Peter Turney. Measuring semantic similarity by latent relational analysis. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1136–1141, Edinburgh, Scotland, 2005.

[17] Thomas Landauer and Susan Dumais. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.

[18] Tony Veale. The analogical thesaurus: An emerging application at the juncture of lexical metaphor and information retrieval. In *Proceedings of the 2003 International Conference on Innovative Applications of Artificial Intelligence*, Acapulco, Mexico, 2003.

[19] Tony Veale. Dynamic type creation in metaphor interpretation and analogical reasoning: A case-study with WordNet. In *Proceedings of the 2003 International Conference on Conceptual Structures*, Dresden, Germany, 2003.

[20] Tony Veale, Jer Hayes, and Nuno Seco. The Bible is the Christian Koran: Discovering simple analogical compounds. In *Proceedings of the Workshop on Computational Creativity*, Madrid, Spain, 2004.

[21] Tony Veale. Analogy generation with HowNet. In *Proceedings of the 2005 International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

[22] Lucien Tesnière. *Éléments de syntaxe structurale.* Paris: Librairie C. Klincksieck, 1959.

[23] Dekang Lin and Patrick Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[24] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 16–19, Washington, U.S.A., 1997.

[25] Dekang Lin. Principle-based parsing without overgeneration. In *Proceedings of the 31st Annual Meeting on ACL*, pages 112–120, Columbus, U.S.A., 1993.

[26] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, Granada, Spain, 1998.

[27] Geoffrey Sampson. *English for the Computer: The SUSANNE Corpus and Analytic Scheme.* Oxford University Press, 1995.

[28] Rion Snow, Daniel Jurafsky, and Andrew Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the 2004 Neural Information Processing Systems Conference*, Vancouver, Canada, 2004.

[29] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 13(11):613–620, 1975.

[30] Gene Golub and Charles van Loan. *Matrix Computations.* Johns Hopkins University Press, third edition, 1996.