# Learning Common Sense Knowledge from User Interaction and Principal Component Analysis

by

Robert Speer

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007



Author .....
Department of Electrical Engineering and Computer Science
August 26, 2007

Certified by .....
Henry Lieberman
Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

# Learning Common Sense Knowledge from User Interaction and Principal Component Analysis

by

Robert Speer

Submitted to the Department of Electrical Engineering and Computer Science
on August 26, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

## Abstract

In this thesis, I present a system for reasoning with common sense knowledge in multiple natural languages, as part of the Open Mind Common Sense project. The knowledge that Open Mind collects from volunteer contributors is represented as a semantic network called ConceptNet.

Using principal component analysis on the graph structure of ConceptNet yields AnalogySpace, a vector space representation of common sense knowledge. This representation reveals large-scale patterns in the data, while smoothing over noise, and predicts new knowledge that the database should contain. The inferred knowledge, which a user survey shows is often correct, is used as part of a feedback loop that shows contributors what the system is learning and guides them to contribute useful new knowledge.

Thesis Supervisor: Henry Lieberman
Title: Research Scientist

# Acknowledgments

This work is dedicated to the memory of Push Singh. Push was my advisor when I joined the Open Mind Common Sense project two years ago, and his big ideas about what the project could become inspired the direction my research would take. He valued research for the long term – to him, what a project would do for the future of AI was more important than what it would do for the next demo week – and I wish he could be here to see the results of the task he gave me.

I owe a lot to Henry Lieberman for taking me as a student and ensuring that this research could continue. I thank him for his advice and support, and for organizing opportunities for me to present my research to a larger audience.

Catherine Havasi was essential to the success of this research. The ideas that made AnalogySpace work arose largely from the research and brainstorming we did together, and I thank her for sharing her thoughts, her critiques, and the expertise she brought from the many years she has spent working on common sense. I am also grateful for her love, friendship, and emotional support, which gave me the motivation to keep going.

I thank Jason Alonso for his design ideas and technical help, and in general I thank my colleagues in the Commonsense Computing Group for being a bunch of excellent people who do interesting research. I would also like to thank my friends . . . and their friends, and their friends' friends . . . for turning out in droves to take my user evaluation. Finally, I owe some thanks to the over 10,000 contributors to Open Mind Common Sense, without whom none of this would have been possible.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In many applications of artificial intelligence, interaction between the human and the computer is hindered by a fundamental gap in understanding between the user and the computer. The user's goal in using the program is to accomplish things in the real world; meanwhile, the computer doesn't even know what the real world is. What the computer is lacking is *common sense*, the body of basic knowledge that people know and computers don't.

The Open Mind Common Sense (OMCS) project in the MIT Media Lab has collected a large corpus of this common sense knowledge by harnessing the knowledge of large numbers of ordinary volunteers on the Web. The project's main Web site has collected over 700,000 statements from tens of thousands of volunteers.

The knowledge that OMCS has collected is represented in a semantic network called ConceptNet, which gives many AI applications access to common sense knowledge [Liu and Singh, 2004]. But the task of making ConceptNet more broadly useful poses the following questions:

- How can we improve the depth of knowledge that ConceptNet has on each topic?

- How do we represent that knowledge in a way that makes use of its implicit large-scale structure?

- How can we encourage contributors to add high-quality knowledge to ConceptNet?

- How can ConceptNet automatically form hypotheses and conclusions that improve its ability to learn?

In my view, the answers to all of these questions are related. We need to extract from the Open Mind corpus a knowledge representation that is informative to both ordinary users and to the computer; create an inference engine that makes efficient use of the knowledge representation, letting the computer determine what it knows and what it has yet to learn; and set up a dialogue between the user and the computer, where the computer uses its inference engine to ask relevant questions and inform the user on how best to teach it.

The knowledge representation that makes this all possible involves transferring knowledge between three levels of representation:

1. The Open Mind corpus, which is made of ordinary sentences in natural language

2. ConceptNet 3, a semantic network built from predicates that are extracted from OpenMind

3. AnalogySpace, which uses principal component analysis to map ConceptNet into a vector space where fast linear algebra operations can make use of its large-scale structure.

ConceptNet 3 represents much of the information in Open Mind in a machine-understandable form. It expresses the statements of common sense collected by Open Mind as a semantic network, built of edges (*predicates*) representing machine-understandable versions of the natural-language statements and nodes (*concepts*) representing the phrases related by those statements.

Open Mind Commons is a knowledge collection interface that runs on top of ConceptNet 3, bridging the gap between the computer-understandable knowledge and contributors. When contributors enter new statements through this interface,

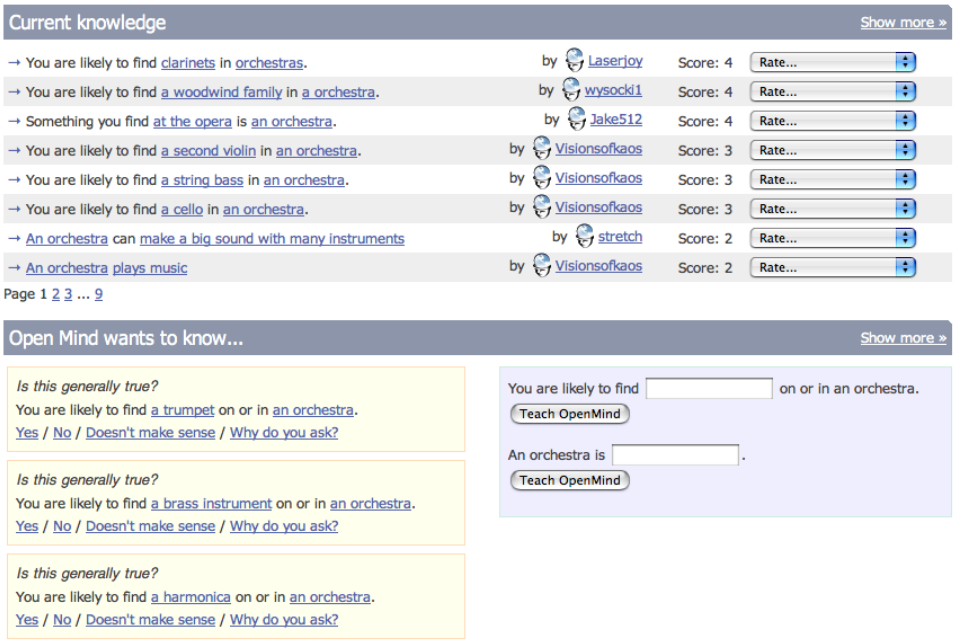


Figure 1-1: The Open Mind Commons interface, displaying some existing knowledge and guiding the user to provide more.

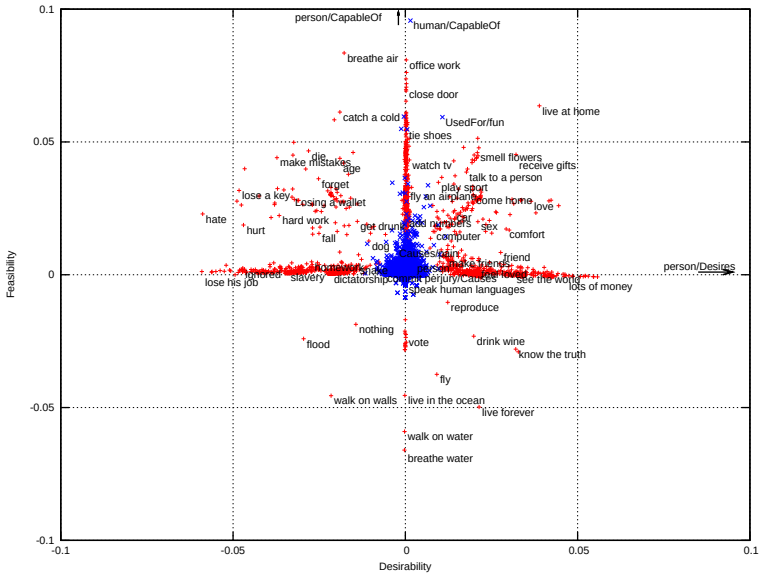


Figure 1-2: A visualization of AnalogySpace, projected into two dimensions by the first two principal components. These particular components distinguish desirable things from undesirable things, and possible actions from impossible ones.

they immediately become part of ConceptNet, and inference procedures that run on top of ConceptNet help to guide the user, by asking relevant questions whose answers will fill gaps in ConceptNet.

These questions are provided by AnalogySpace, which represents ConceptNet in a vector space, similarly to the way that latent semantic indexing (LSI) represents documents and search terms, or Google's PageRank represents Web pages. With this representation, a method of common-sense inference called "cumulative analogy" can be implemented using optimized linear algebra techniques. The inferences produced by AnalogySpace are considered "generally or occasionally true" by surveyed users 68.9% of the time, compared to 80.8% for existsing knowledge in OMCS. The vector-space representation can also be used by applications to find parts of ConceptNet that relate to a natural language input, which can be applied to AI tasks such as predictive text entry and topic sensing.

The focus of this thesis is AnalogySpace, and the new methods of reasoning that this new representation enables for the Open Mind Common Sense project. Along the way, the tools in the Open Mind project that are equipped to work with AnalogySpace – Open Mind Commons and ConceptNet 3 – will be introduced and explained.

# Chapter 2

# Background

Common sense is the body of basic real-world knowledge that people know and computers don't. It consists of the kind of things that people leave unstated because they are obvious. Things fall down, not up. Parents are older than their children. People want to be happy. These kinds of facts underlie every human conversation without being explicitly said.

Because computers have no knowledge of common sense, people cannot interact with computers as effectively as they would interact with other people. People want computers to help them accomplish things in the real world, but computers don't even know what the real world *is*. Giving computers access to common sense knowledge is thus a fundamental and important problem in AI.

## 2.1 Why do we need common sense?

When I am sending a text message from my cellular phone, I can use a system called T9 to type text on its numeric keypad with only one keypress per letter. The system deals with its incomplete input by choosing, out of the exponentially large space of texts I might have entered, a message that is made of English words. However, I often have to correct the system, in cases where multiple words are *possible* but only

one is what I mean. If I press the sequence of keys corresponding to this message:

```
I am on the bus.  I'll be home soon.
```

What appears instead is a meaningless message, spelled with the same digits:

```
I am on the cup.  I'll be good room.
```

The phone needs more information in order to get the message right. What it needs is not just a stronger model of the English language – after all, "I am on the cup" is a perfectly grammatical sentence, and the second sentence is nearly so. What the phone needs is a model of which message *makes sense*.

If my phone had access to the knowledge that I, as a person, am more likely to be on a bus than on a cup, that I am probably not a room, and that I'd more likely need to send a text message about "being home" than "being good", then it would have the information it needs to choose the correct message. In fact, the domain of predictive text entry is one where ConceptNet, the focus of this research, has already been applied [Stocky *et al.*, 2004]. Markov N-gram models are more frequently used for this purpose, but the effectiveness of N-grams falls off quickly as $N$ increases, meaning that these models cannot use more than a few words of context [Lieberman *et al.*, 2005], while a ConceptNet-based method can construct a semantic context out of many recent words.

Though T9 text can be easily corrected – the phone has the 0 key set aside for that purpose – the same problem crops up in areas where it is harder to fix the computer's mistakes. In automatic speech recognition, the computer faces a similar problem of choosing one sensible text out of a large space of possibilities; the mistakes that speech recognizers make can be hilarious, such as substituting "peach wreck in kitchen" for "speech recognition",[1] but they hinder productivity, and the very reason the errors are amusing is that the computer should be able to know better. Speech recognition is

[1]This example comes from the website of the TypeWell text transcription software: `http://www.typewell.com/speechrecog.html`.

another application that can be improved using ConceptNet [Lieberman *et al.*, 2005].

These are just a few examples where human-computer interaction would be improved by common sense knowledge. Some other problems in AI that can benefit from common sense are computer vision [Schierwagen, 2001, p. 7], and user interfaces such as calendar software [Mueller, 2000; Smith, 2007].

## 2.2 The Open Mind project

Open Mind Common Sense was started in 1999 at the MIT Media Lab, with the goal of collecting a large knowledge base of common sense from ordinary people. The focus of OMCS was its Web site, which prompted its visitors to answer questions and to contribute knowledge in several different forms. It stored its knowledge in the form of English sentences; some of these sentences came from fill-in-the-blank templates like "You would go to the store because _____", and some were entered as free text. This Web site has collected over 700,000 statements from tens of thousands of volunteers.

Even though the site provided no reward for entering correct, sensible information, most of the information it collected was judged by reviewers to be correct and sensible [Singh *et al.*, 2002]. However, because the information was collected as English statements, it would take some effort to get that information into a form that would be useful to a computer.

## 2.3 Building on Open Mind

### 2.3.1 ConceptNet

ConceptNet [Liu and Singh, 2004] is a semantic network that is built from the information that Open Mind has collected, expressing that information in a form that applications can use.

Many kinds of statements that users have entered into Open Mind can be expressed as relationships between two *concepts*, which are essentially short phrases of natural

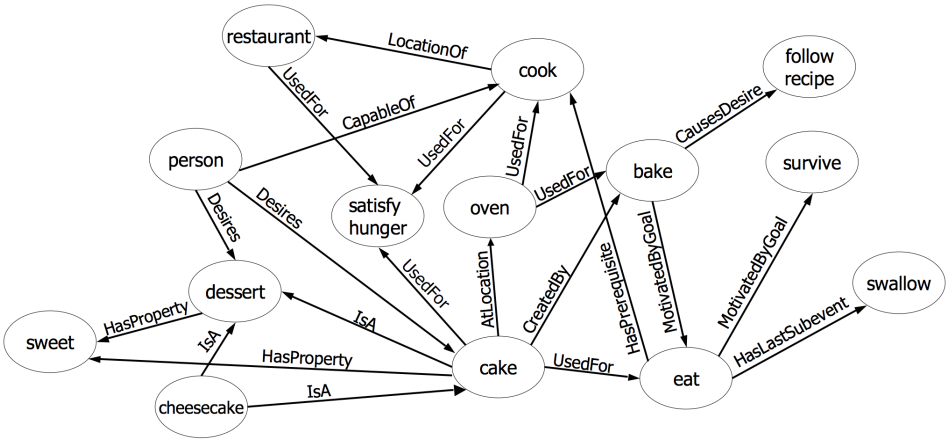


Figure 2-1: An illustration of a small section of ConceptNet.

language. In a sentence such as "a trunk is part of a car", for instance, the two concepts are "trunk" and "car", and the relationship is "PartOf". When such a relationship is expressed in a computer-understandable form, it is called a *binary predicate*.

ConceptNet scans the statements in Open Mind and expresses as many of them as possible as binary predicates. The result is that the knowledge can be viewed as a labeled, directed graph where the nodes are concepts and edges represent predicates. One edge in this graph is labeled "PartOf", and points from the concept "trunk" to the concept "car".

The previous release of ConceptNet, ConceptNet 2, uses both pattern-matching and natural language chunking to extract predicates from natural language, making use of a natural language library called MontyLingua [Liu, 2004] to extract the relation and the two concepts from each sentence. ConceptNet 3, which will be described in the next chapter, is constructed by a similar process, but it combines parsing and pattern-matching into a single operation.

### 2.3.2 Learner

Observing that "OMCS does not leverage knowledge it has gathered to guide its future acquisition in any way" [Chklovski, 2003, p. 10], Tim Chklovski used knowledge

| Objects | | Properties | | | | |
|---|---|---|---|---|---|---|
| | | contains knowledge | has pages | is cold | is for reading | |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | |
| book | ... | x | x | | x | ... |
| ice | ... | | - | x | | ... |
| **newspaper** | ... | **x?** | **x** | | **x** | ... |
| magazine | ... | x | x | | x | ... |
| ⋮ | | ⋮ | ⋮ | ⋮ | ⋮ | |

Figure 2-2: An example of reasoning by cumulative analogy, adapted from [Chklovski, 2003].

extracted from Open Mind to create the Learner system [Chklovski, 2003]. With this system, Chklovski introduced the idea of reasoning about common sense by "cumulative analogy".

In cumulative analogy, statements of common sense knowledge are divided up into *objects* and *properties*, where each object can be described as having many properties. Chklovski's implementation parses sentences that consist of a noun phrase followed by a verb phrase. The noun phrase – the subject of the sentence – is considered as the *object*, and the verb phrase, describing something that is true about that object, is the *property*.

Reasoning by cumulative analogy starts from a particular object. The first step is to find the object's nearest neighbors: other objects that have the most similar sets of properties. The properties of those nearest neighbors are then examined. Any property that occurs many times among an object's nearest neighbors, but is not already known about the object itself, can be inferred to be true about the at object.

Consider a cumulative analogy starting from the object "squirrel", where the knowledge base already knows that squirrels are living things, are small, and are mammals, among possibly other properties, but it does not know that they have fur. The nearest neighbors to "squirrel" are other objects that have similar properties, such as other small mammals. Many of these nearest neighbors are described as having fur, so by cumulative analogy to those nearest neighbors, the knowledge base infers that squirrels may have fur.

When cumulative analogy is used in ConceptNet, the *object* is renamed the *con-*

*cept*, for consistency with existing ConceptNet terminology. This kind of analogy is an important part of the newest Open Mind interface, Open Mind Commons. *AnalogySpace*, described in Chapter 4, adds more power to cumulative analogies by creating a vector space of potential analogies and expressing it in terms of its principal components.

### 2.3.3 GlobalMind

Common sense knowledge is often not universal; much of it is culturally dependent [Anacleto *et al.*, 2006]. When you enter a restaurant in the United States, for instance, you usually wait for an employee of the restaurant to seat you. In Korea, you sit down at a table and wait for service [Chung, 2006, p. 25]. Both ideas of what to do in a restaurant can be considered common sense knowledge for someone, but which a particular person considers to be true depends on the culture that person is familiar with.

The knowledge collected by OMCS inherently represents only the common sense of English speakers, and generally only that of Americans. The GlobalMind project [Chung, 2006] branched off from Open Mind in order to collect knowledge from several cultures in several languages, and allow bilingual users to supply translations between the languages. In this way, GlobalMind collects knowledge that can be used in applications in languages other than English. It can infer translations between languages to compare the common sense of different cultures, and to enable applications such as an automatic, context-sensitive phrasebook [Musa *et al.*, 2003].

GlobalMind added sentence patterns and knowledge-collection interfaces in East Asian languages to Open Mind. Its web site (`globalmind.media.mit.edu`) has collected over 15,000 binary predicates in Korean and 10,000 in Japanese since 2005.

### 2.3.4 OMCS no Brasil

Separately from the GlobalMind project, a sister project in Brazil (*Open Mind Common Sense no Brasil*) began collecting common sense in Portuguese.

Seeing the difficulties that free-text statements create in the English OMCS corpus, the creators of *OMCS no Brasil* decided to only allow statements to be entered through fill-in-the-blank templates. This makes it much easier to create a ConceptNet from the Portuguese OMCS corpus, as no parsing is necessary.

*OMCS no Brasil* has collected over 80,000 statements of from its contributors, many of them checked for accuracy by the project's maintainers. Some of these are statements of general common sense, while others have been sorted into topic areas such as health and sex education. These topics are useful to an application which uses *OMCS no Brasil* to guide learning activities [de Carvalho *et al.*, 2007].

## 2.4 Related work

### 2.4.1 Common sense knowledge bases

Along with Open Mind, other projects have worked with databases of common sense knowledge, taking somewhat different approaches.

The Cyc project [Lenat, 1995], started by Doug Lenat in 1984, acquires its knowledge from knowledge engineers who handcraft assertions and express them in Cyc's logical framework using a logical representation called CycL. To use Cyc for natural language tasks, one must first translate the natural language text into CycL. This is a complex process, whose difficulty is equivalent to semantic parsing. While Cyc does its reasoning in a domain of unambiguous logical assertions, Open Mind focuses on working in a domain where ambiguity is pervasive and logical clarity is often lacking, features that are inherent to natural language interaction with users.

Erik Mueller's ThoughtTreasure [Mueller, 1998] is a system for advanced reasoning and natural language interaction using a common sense knowledge base. It fills a different role from Open Mind – for example, it does not have a way of inferring new relational knowledge, and only acquires new knowledge if the knowledge is hand-coded or appears in a relational table [Mueller, 1998, ch. 9]. Much of the knowledge represented in ThoughtTreasure, though, is compatible with the ConceptNet repre-

sentation, and in fact a large quantity of "part-of" knowledge from ThoughtTreasure has been imported with permission into Open Mind. ThoughtTreasure and Open Mind can be seen as potential symbiotes, not as competitors.

### 2.4.2 Semantic networks

Semantic networks, which computationally define the meanings of words and phrases, are usually considered to be different from common sense knowledge bases. Much of the information that semantic networks use to define words, however, overlaps with the domain of common sense knowledge. This is one reason that Open Mind Common Sense aims to be used as a semantic network (in the form of ConceptNet) as well as a knowledge base.

WordNet [Fellbaum, 1998] is a particularly well-known semantic network that is used in many applications. Much of the information in WordNet expresses common-sense knowledge. The question of "What kind of thing is it?" is answered by WordNet's *hypernym* relation type, just as it is by ConceptNet's *IsA*. The questions "What is it part of?" and "What parts does it have?" are similarly answered by WordNet's *meronym* and *holonym* relations, as they are by ConceptNet's *PartOf*.

A newer and less well-known semantic network, the Brandeis Semantic Ontology (BSO) [Pustejovsky *et al.*, 2006], overlaps even more with the domain of common sense knowledge. Based on James Pustejovsky's Generative Lexicon theory of semantics [Pustejovsky, 1998], the BSO defines four types of relations called *qualia*:

- *Formal*, which defines a hierarchy of types; equivalent to ConceptNet's *IsA*
- *Constitutive*, the relationship between an object and its parts; equivalent to ConceptNet's *PartOf*
- *Telic*, expressing the purpose or function of an object; equivalent to ConceptNet's *UsedFor*
- *Agentive*, describing how the object comes into being. Not previously present in ConceptNet, this relation has now been introduced as *CreatedBy*.

There is a considerable amount of overlap in the knowledge contained in semantic networks such as the BSO and the knowledge contained in ConceptNet 3. This point will be elaborated in Section 3.4.1, which evaluates how many ConceptNet predicates overlap with information contained in the BSO and WordNet.

### 2.4.3 Principal component analysis

The process of using principal component analysis, or singular value decomposition,[2] to find similarities in ConceptNet derives from the AI technique of *latent semantic analysis* (LSA), which applies the same technique over a domain of words and documents [Deerwester *et al.*, 1990]. LSA is often used in information retrieval, where it is known as LSI for *latent semantic indexing*. In the representation used by LSA, a document is seen as an unordered collection of words, and the matrix of words verses documents is analyzed with SVD documents are sorted into implicit categories according to the words that are contained in them.

Various research projects have extended the idea of LSA to apply it to the goal of finding semantic similarities, not just topical similarities, between words. The approach used by [Patwardhan and Pedersen, 2006] is to apply LSA with documents replaced by short phrases or sentences, so that words acquire connections from appearing near each other instead of appearing anywhere in the same document. [Banerjee and Pedersen, 2003] use the occurrence of words in a WordNet gloss as a measure of semantic similarity. [Turney, 2005] uses co-occurring pairs of words themselves as vectors, with the other domain being the surrounding context in which they appear. That project produces analogies in a more literal sense than AnalogySpace: the resulting vector space of similarities between pairs of words can be used to answer SAT-style multiple-choice analogy questions, as in "quart:volume :: mile:distance".

The idea of applying principal component analysis to graph-structured representations of knowledge is relatively unexplored. Perhaps the closest thing to it is an-

[2]SVD and PCA refer to the same technique with different terminology [Wall *et al.*, 2003]. One way of making a distinction between them is that PCA is a general process, while SVD is a mathematical implementation of it.

other project descended from Open Mind Common Sense. This project [Gupta and Kochenderfer, 2004] used Honda's domain-specific Open Mind Indoor Common Sense corpus, constructing an SVD for each relation (such as PartOf or AtLocation) that would infer new instances of that relation.

AnalogySpace goes beyond the Honda project: instead of dividing up the different types of knowledge into distinct SVD matrices, it uses the relation type as part of the information in the SVD, by introducing the "concept/property" representation described in Section 4.1. One SVD, then, describes similarities and inferences of all kinds, which allows much more information to contribute to each inference, and helps this kind of reasoning to generalize to the entire domain of ConceptNet.

# Chapter 3

# Acquiring knowledge

The results in this thesis are built on ConceptNet 3, which I designed along with other members of the Commonsense Computing Group [Havasi *et al.*, 2007] as an architecture for extracting knowledge from the Open Mind corpus and collecting more knowledge to add to it. The original purpose of ConceptNet 3 was to be a backend for an updated Web interface to Open Mind, which would improve the quality of the data it collected using the Learner model of asking questions by analogy. This research later led us to try singular value decomposition as an algorithm for inference.

## 3.1 The Design of ConceptNet 3

In developing ConceptNet 3, we drew on our experience with working with ConceptNet as users, and observed what improvements would make it easier to work with. The new architecture of ConceptNet is more suitable to being incrementally updated, being populated from different data sources, supporting parallel semantic networks such as the English and Portuguese ConceptNets, and using various modes of inference to find potential new predicates.

### 3.1.1 Concepts

The basic nodes of ConceptNet are *concepts*, which are aspects of the world that people would talk about in natural language. Concepts correspond to various constituents of the common-sense statements that users have entered; they can represent noun phrases, verb phrases, adjective phrases, or prepositional phrases (when describing locations).

Although they are derived from constituents, concepts are not literal strings of text; a concept can represent many related phrases, through the normalization process described later.

### 3.1.2 Predicates

In a semantic network where concepts are the nodes, the edges are *binary predicates*, which express relationships between two concepts. (Remember that predicates are complete statements of common sense knowledge that express a relationship – in other areas, they may be better known as *assertions*. Predicates have three parts: a *relation*, which can be thought of as a function of two arguments, and the *left concept* and *right concept* that form the arguments of that function. Some examples of relations are IsA, PartOf, HasLocation, and UsedFor. The 19 basic relation types, shown in Table 3.1, are not a closed class, and we plan to add more in the future. One of them, ObstructedBy, has not yet been collected in English, only in Portuguese.

A predicate can be notated as a parenthesized expression, such as (UsedFor ⟨bed⟩ ⟨sleep⟩). Keep in mind that the concepts in angle brackets are not literal strings of text, but the best way of representing a concept is with a representative phrase of text that produces it. In this notation, ⟨bed⟩, ⟨a bed⟩, and ⟨beds⟩ all refer to the same concept.

Predicates are extracted from the natural language statements that contributors enter, and they maintain a connection to natural language by keeping a reference to the original sentence that generated them, as well as the substrings of the sentence that produced each of their concepts, and the string representing the sentence with

| Label | Example |
|---|---|
| IsA | Hockey is a sport. |
| PartOf | A finger is part of a hand. |
| AtLocation | You are likely to find a book in a library. |
| MadeOf | Windows are made of glass. |
| UsedFor | Pens are used for writing. |
| CapableOf | Boats can float on water. |
| HasProperty | Sunsets are beautiful. |
| Desires | A person wants love. |
| CausesDesire | Being cold would make you want to light a fire. |
| Causes | The effect of having a haircut is to have shorter hair. |
| MotivatedByGoal | You would do housework because you want to have a clean house. |
| HasSubevent | One of the things you do when you read a book is turn pages. |
| HasFirstSubevent | The first thing you do when you go for a drive is get in the car. |
| HasLastSubevent | The last thing you do when you take a shower is dry off. |
| HasPrerequisite | If you want to get fit, you should lift weights. |
| DefinedAs | Death is the end of life. |
| ReceivesAction | An apple can be eaten. |
| ObstructedBy | (*Quando se tenta dormir, um problema encontrado pode ser insônia.*) |
| CreatedBy | Music is created by composing. |

Table 3.1: Relation types in ConceptNet 3.

the concepts replaced by placeholders, such as "`{1} is used for {2}`". This last string is called a *frame*.

Because ConceptNet, whenever it stores a predicate, keeps track of the frame and the original text for each of its concepts, it maintains the information it needs to turn that predicate or similar predicates that are derived from it back into natural language.

### 3.1.3 Implementation

ConceptNet 3 is implemented as a PostgreSQL database, with interconnected tables that represent predicates, concepts, relations, users, and scores assigned by users. Each predicate also stores a reference to its natural language representation, in the form of two unstemmed strings of text and a frame, and other tables store parsing patterns and lists of stopwords for the purpose of creating ConceptNet.

The layout of the database is defined by CSAMOA [Alonso, 2007], an API for constructing a ConceptNet from a data source. CSAMOA defines a modular struc-

ture for the procedures that build ConceptNet, including natural language processing tools, allowing them to be easily replaced to deal with another data set or another language.

## 3.2 Creating ConceptNet

### 3.2.1 Pattern matching

A prominent problem that must be confronted by ConceptNet is how to translate the Open Mind corpus, which consists of unparsed English sentences, into meaningful predicates. Previous versions of ConceptNet were created with a combination of two techniques: regular-expression pattern matching, and chunking to distinguish patterns that were equivalent except for the types of phrases they should match.

Most predicates were created by simply matching a regular expression; the HasSubevent relation could come from the pattern "`One of the things you do when you (.+) is (.+)`". Given the statement "One of the things you do when you *drive* is *steer*", for example, this would produce the predicate (HasSubevent ⟨drive⟩ ⟨steer⟩).

An example of a pattern that needs to have its phrase types disambiguated is "`(.+) is (.+)`", which could match IsA statements such as "grass is a plant", HasProperty statements such as "grass is green", and ReceivesAction statements such as "grass is eaten by cows". ConceptNet 2 would resolve this by feeding the match result into the MontyLingua chunker [Liu, 2004], to determine whether it was a noun phrase, verb phrase, or adjective phrase.

Regular expression pattern matching has its limitations. Suppose that someone enters this statement: "One of the things you do when you *change a lightbulb* is *make sure the switch is off*". This is supposed to produce the predicate (HasSubevent ⟨change a lightbulb⟩ ⟨make sure the switch is off⟩), but because regular expressions match greedily by default, the regular expression instead matches (HasSubevent ⟨change a lightbulb is make sure the switch⟩ ⟨off⟩).

Using regular expressions to check patterns also makes certain patterns impossible:

the common HasLocation pattern "You are likely to find *noun-phrase prepositional-phrase*", for example, has to be broken up into cases according to the preposition, because there is otherwise no word in between to indicate where the first match ends and the second begins. Even more problematic is the catch-all CapableOf pattern, *noun-phrase verb-phrase*, intended to match statements such as "Birds fly". ConceptNet 2 handles this as a special case.

It would be infeasible to check all matches with the chunker, however, because chunking of natural language is inherently inaccurate. It would take a full parser to identify many kinds of verb phrases, for example, and even then some verb phrases may be missed because one word in them receives the wrong part-of-speech tag.

### 3.2.2 Shallow parsing

The approach taken by ConceptNet 3 is to use a simple parser as a kind of pattern matcher. The parser is not asked to produce a single correct interpretation of a sentence; it is only used to check whether an interpretation provided by a pattern is *possible*.

In this system, the HasSubevent pattern above becomes "One of the things you do when you *VP* is *VP*". When a sentence is found to potentially match this pattern, the shallow parser is consulted to determine whether the portions of the sentence that match an occurrence of *VP* are reasonable verb phrases. The parser never has to perform the extremely unreliable operation of guessing which of many parse trees is correct.

The parser is a simple, rule-based, bottom-up chart parser [Wirén, 1987] that applies this "check, don't guess" approach at all levels. In the rule $VP \rightarrow V\ NP$, the parser recursively checks whether there are reasonable $V$ and $NP$ constituents that could span the segment of text being checked as a *VP*. Even the tagger that originally assigns part-of-speech tags to the words cannot make any unilateral decisions, so the tagger is an N-best Brill tagger [Brill, 1992] that assigns all reasonable tags to each word. The complete list of parsing rules that build on the results of the N-best Brill tagger can be found in Appendix A.

As a result of this design, the parser does just what it needs to do for pattern-matching: it acts as a filter that prevents patterns from matching against the wrong constituents, or segments of text that are not constituents at all.

### 3.2.3 Link parsing

Though the shallow parser will extract the meaning that the contributor intended in most cases, it will miss useful information in certain forms of sentences. In particular, contributors who enter IsA statements in free text tend to use complex sentence structures that provide extraneous information beyond the level that can be represented in ConceptNet, but a simple, useful statement can often be found embedded within that complex structure. You can see some examples of simplified statements in Table 3.2.

| What the contributor says | What OpenMind hears |
|---|---|
| A goldfish is a type of carp that makes a nice pet | A goldfish is a carp |
| A nightgown is a long, loose garment worn to bed | A nightgown is a garment |
| A uniform is a special outfit worn by members of a group | A uniform is a outfit |
| A foot is a unit of measurement equal to twelve inches | A foot is a unit of measurement |
| A hut is a small, simple shelter | A hut is a shelter |

Table 3.2: Simplifying IsA statements.

This simplification can be done using a link parser (also known as a dependency parser), which parses complete sentences and represents them as a collection of links between words, instead of as a grammatical tree. ConceptNet uses the Link Grammar Parser [Sleator and Temperley, 1993] for this purpose.

In a complex IsA statement, we want to find the two arguments linked to the verb "is". The targets of these links are single words, but usually we would like to pull in some additional words, such as a determiner, so we can represent the simplified predicate by a reasonable English sentence, or the preposition "of" and its argument, because this often conveys information that is inherent to the concept and should not

be discarded.

ConceptNet 3 runs free-text statements through the Link Grammar Parser to find IsA statements, identify the words linked by "is", pull in additional words as necessary, and put the selected words together into a sentence. The contents of Table 3.2 are actual results from the link parser.

### 3.2.4 Normalization

When a sentence is matched against a pattern, the result is a "raw predicate" that relates two strings of text. The *normalization* process determines which two concepts these strings correspond to, turning the raw predicate into a true edge of ConceptNet.

The following steps are used to normalize a string:

1. Remove punctuation.
2. Remove stop words.
3. Run each remaining word through a stemmer. We currently use Porter's Snowball stemmer, in both its English[1] and Portuguese versions [Porter, 2001]. As a special case in English, assign "people" the stem `person`, not its Porter stem `peopl`.
4. Alphabetize the remaining stems, so that the order of content words in the phrase doesn't matter.

A concept, then, encompasses all phrases that normalize to the same text. As normalization often results in unreadable phrases such as "endang plant speci" (from "an endangered species of plant"), the normalized text is only used to group phrases into concepts, never as an external representation. This grouping intentionally lumps together many phrases, even ones that are only related by accidents of orthography, because we have found this to be an appropriate level of granularity for reasoning about undisambiguated natural language text collected from people.

[1]For compatibility with previous work, we use the original version of the English Snowball stemmer (the one commonly called "the Porter stemmer"), not the revised version.

Phrases that consist entirely of stopwords, such as "you", "something", or "that one", cannot be made into a concept. Statements that involve such phrases are discarded. This is often desirable, because it weeds out excessively vague statements.

### 3.2.5 Reliability of Assertions

In ConceptNet 3, each predicate has a score that represents its reliability. This score comes from two sources. A user on Open Mind Commons can evaluate an existing statement and increase or decrease its score by one point. The score can also be implicitly increased when multiple users independently enter sentences that map to the same predicate, and this is where the majority of scores come from so far.

The default score for a statement is 1. A statement with score 1 is supported by one person: the person who entered it. Statements that achieve zero or negative scores, because a user has decreased their score or entered the opposite statement, are considered unreliable, and are not used for making analogies. Statements with positive scores contribute to analogies proportionally to their score.

### 3.2.6 Polarity

In ConceptNet 3, we have introduced the ability to represent negative assertions. When an interface to ConceptNet, such as Open Mind Commons, asks a question that it has formed by analogy, and the user answers "no", ConceptNet needs to be able to record the relevant fact that the inferred statement is false (or that its negation is true, which we assume to be equivalent in the knowledge representation of ConceptNet). Also, it is useful to allow ConceptNet to represent contributed statements that express the negation of one of ConceptNet's standard relations, such as "A whale is not a fish" or "Cats cannot read".

To this end, we added a *polarity* parameter to our predicate models that can take the values 1 and $-1$. A negative polarity is assigned either when a user of Commons answers "no" to a question, or when the process that builds ConceptNet finds a sentence with a negative word in it.

In English, the word "not" is treated as an adverb by the Brill tagger, so it can be picked up by the shallow parser anywhere that an `ADVP` is allowed. Other words that indicate negation include the determiner "no", the morpheme "n't", and the adverb "never". If any of these words occurs in a parsed sentence, either in the frame or in one of the concepts, the resulting predicate gets its polarity negated.[2]

The Portuguese Open Mind uses different templates for users to enter positive statements and negative statements, so negation can be detected entirely within the pattern matcher. Concepts in Portuguese are not scanned for negative words.[3]

About 2.9% of the English predicates and 4.2% of the Portuguese predicates currently in ConceptNet 3 have a negative polarity.

Importantly, score and polarity are independent quantities. A predicate with a negative polarity can have a high, positive score, indicating that multiple users have attested the negative statement (an example is "People don't want to be hurt"). Predicates with a zero or negative score, meanwhile, are usually unhelpful or nonsensical statements such as "Joe is a cat" or "A garage is for asdfghjkl", not statements that are "false" in any meaningful way.

## 3.3 Open Mind Commons

Open Mind Commons [Speer, 2007] is a new Web-based interface for users to contribute knowledge to Open Mind, accessible at `http://commons.media.mit.edu`. It was based in part on the recommendations of a paper that evaluated the original OMCS site [Singh *et al.*, 2002], which proposed features that should be present in "OMCS 2", which was never actually implemented.

Among other things, the authors recommended focusing on knowledge entry through templates instead of in free-form text, allowing users to help validate and organize the

[2]In the case where there are multiple negated words, as in "A person doesn't want to not have food", the polarity ends up being positive because it is negated twice, and the resulting predicate is equivalent to "A person wants to have food". In English, this is usually the correct behavior.

[3]This has the advantage of avoiding confusion with the way double negatives are used in Portuguese, as well as allowing me to maintain the pattern matcher without actually knowing Portuguese.

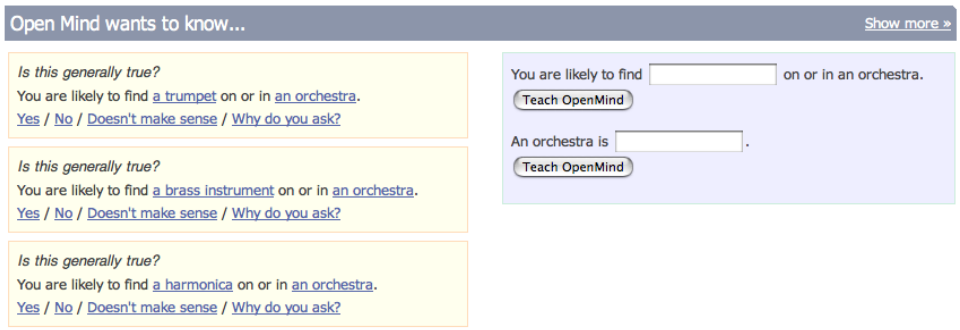


Figure 3-1: Open Mind Commons asks questions to fill gaps in its knowledge.

knowledge, and giving users feedback about what the system was learning by making inferences. We created Open Mind Commons to fulfill these goals. Its interface is designed around letting users explore and refine the existing knowledge, and using inference to ask the right questions to guide users' contributions.

Open Mind Commons uses ConceptNet 3 as its database backend. The knowledge that users see on the site is the knowledge currently present in ConceptNet 3, and when users add more knowledge, it immediately becomes part of ConceptNet.

In order to incorporate inference into Open Mind Commons, we began by using the "cumulative analogy" method that was originally implemented in Chklovski's LEARNER system [Chklovski, 2003], which is described in section 2.3.2. When Commons makes an inference by analogy, it puts it into a priority queue of questions to ask users when they browse to either of the two concepts involved in the inference. The result is that when a user looks up a concept, such as "ocean", the user sees a list of what Commons considers the most relevant questions to ask to fill gaps in its knowledge about that concept. An example of these questions is shown in Figure 3-1.

In addition to inferring entire statements by analogy, Open Mind Commons also makes *partial inferences*, in which one of the concepts is left blank. This allows the interface to present the user not just with yes/no questions, but with open-ended, fill-in-the-blank questions such as "You would find _____ in the ocean".

Later, we modified this method to find analogies by principal component analysis

instead of by nearest neighbors. This is the "AnalogySpace" method described in Chapter 4. The modified method, which is now used for inference in Open Mind Commons, remains true to the motivation behind cumulative analogy, but implements it in a very different way.

Asking questions based on analogies serves to make the database's knowledge more strongly connected, as it eliminates gaps where simply no one had thought to say a certain fact; it also helps to confirm to contributors that the system is understanding and learning from the data it acquires.

### 3.3.1 Reconstructing natural language

In order to ask users whether inferred statements are true, Open Mind Commons needs to turn abstract predicates back into natural language. It does this using the strings of text that ConceptNet stored when it originally turned natural language statements into predicates, as described on page 30. This is done by the following procedure, referred to as *InferNaturalLanguage*, which turns a predicate (specified by two concepts, a relation, and a polarity) into a sentence.

#### The InferNaturalLanguage procedure

1. Find a canonical sentence frame for the relation.

   In a language where all input to Open Mind is done through templates, such as Portuguese, the canonical frame is simply the most common template that the Web site used to acquire knowledge about that relation. In English, a canonical frame was chosen by hand for each relation, with the goal of being worded generally enough that it will make sense for any instance of that relation. For the AtLocation relation, for example, the canonical English frame is "`You are likely to find {1} on or in {2}`", and the canonical Portuguese frame is "`Uma coisa que você pode encontrar em um(a) {2} é um(a) 1`".[4]

[4]These canonical frames represent a tradeoff between natural language and generality. Often, they need to include slightly unnatural constructions, such as the English "on or in" or the Portuguese "um(a)".

2. For each concept, find the text that is most often used to express it in this context.

   (a) First, look up all predicates that use the chosen frame, and find all natural language texts in those predicates that have the same stem as the concept in question, in the same place in the frame. Reject texts containing negative words such as "not". Choose the remaining text that appears the most times in this context.

   (b) If step 2a produced no candidates, repeat it using all predicates expressing the same relation, but not necessarily the same frame.

   (c) If step 2b produced no candidates, find the 5 most common texts that normalize to the same stem, regardless of what kinds of predicates they appear in. From those 5 texts, choose the shortest one.

3. Fill in the frame with the two texts to yield a sentence.

The effect of this process is to construct new sentences out of predicates by following the example of other sentences that were parsed into predicates. Some examples of the sentences it produces can be seen when it is used in the user evaluation in Chapter 5. It does not work perfectly, but it often produces reasonable and understandable sentences.

## 3.4 Contents of ConceptNet

ConceptNet, when constructed from the Open Mind Common Sense data set using the process described in this chapter, contains over 250,000 unique predicates in English, derived from over 300,000 of the sentences in OMCS (some of them expressing the same predicate). In Portuguese, it contains more than 80,000 unique predicates derived from more than 125,000 sentences. The English sentences that were not made into ConceptNet predicates have been preserved in their raw form in the ConceptNet database, where a future parser may be able to make predicates out of them.

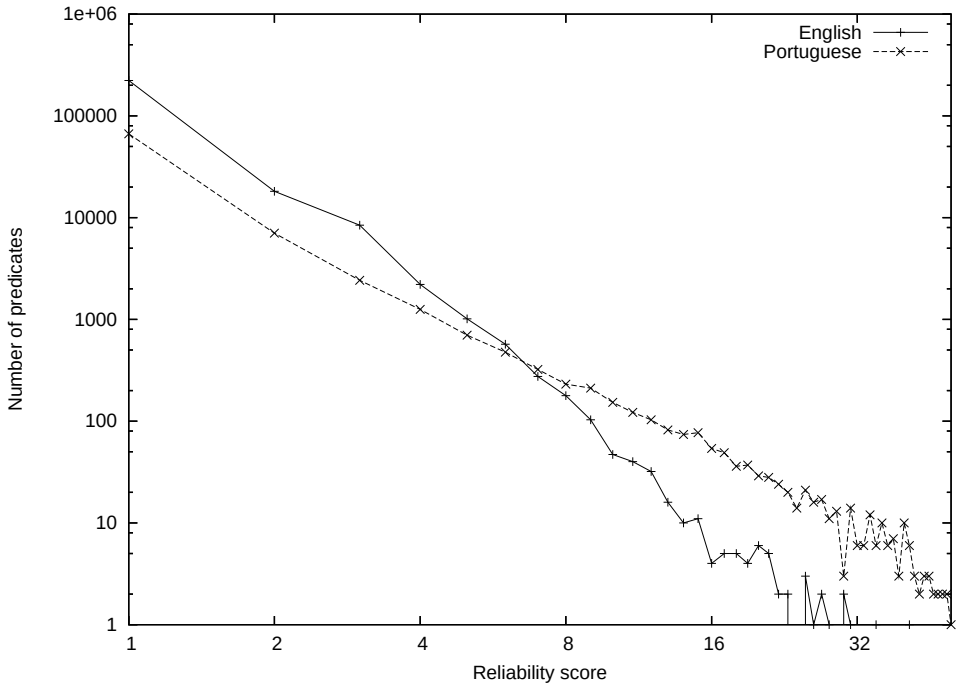


Figure 3-2: The distribution of scores among predicates in the English and Portuguese ConceptNets.

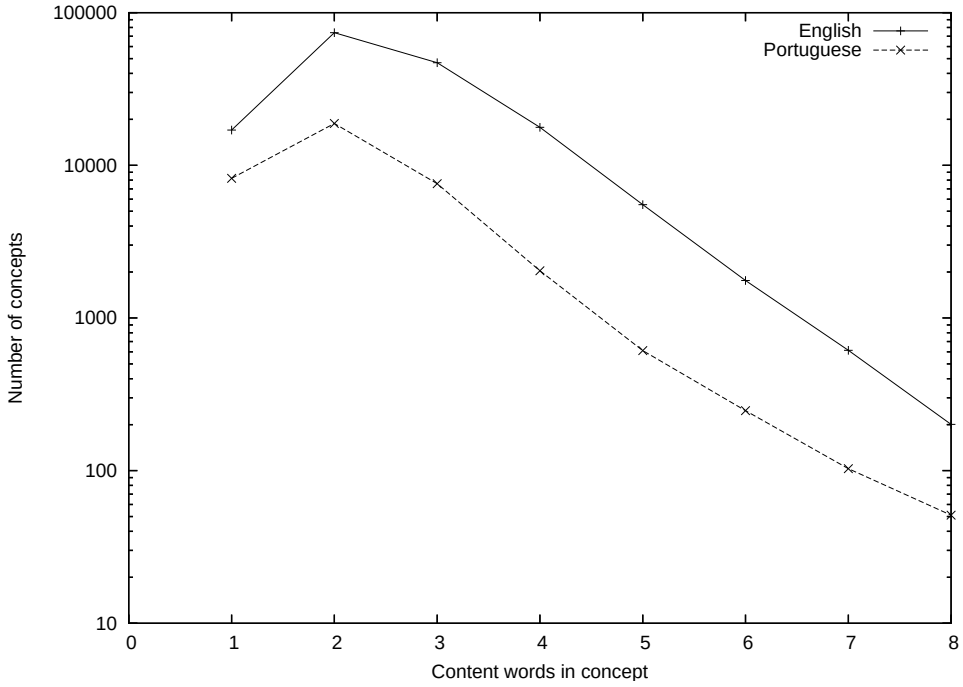


Figure 3-3: The word lengths of concepts in the English and Portuguese ConceptNets.

Surprisingly, although the Portuguese corpus has been around for a shorter time and has fewer predicates, its predicates tend to have higher scores, from being asserted by different users more times on average. The fact that all Portuguese statements were entered through structured templates, not through free text, may have caused them to coincide more often.

The highest-scored predicate in the English ConceptNet is "Hockey is a sport", asserted independently by 55 different users.[5] The highest-scored predicate in the Portuguese ConceptNet is "Pessoas dormem quando elas estão com sono" ("People sleep when they are tired"), asserted independently by 318 users. The distribution of scores is shown on a log-log graph in Figure 3-2, showing how Portuguese has fewer predicates overall than English, but has more predicates with high scores.

The concepts that are the most inherently useful are the ones containing one or two content words (words that remain after stopwords are removed), as these can be matched against natural language more often than concepts that are many words long. Although there are more possible concepts as the number of words increases, the number of concepts of each word length falls off exponentially after reaching a maximum at a length of 2 words, as shown in Figure 3-3. This shows that people do not often contribute statements involving unnecessarily long phrases. The falloff in concept length in English could be due in part to the lower probability of long phrases being accepted by the chart parser, but the same rate of falloff occurs in Portuguese, where no parser is used.

### 3.4.1 Comparison to other semantic networks

Section 2.4.2 introduced the idea that ConceptNet contains the same kind of information as other semantic networks, such as WordNet [Fellbaum, 1998] and the BSO [Pustejovsky *et al.*, 2006]. This can be shown by sampling predicates from Concept-

[5] In previous papers, the highest-scored predicate was claimed to be "Baseball is a sport", at a score of 72. It turns out that statements were being counted extra times when they were re-used by OMCS 1 as prompts for higher-level statements, expressing facts of the form " 'Baseball is a sport' is true because...", and this created a bias in which older predicates got higher scores. With this bias removed, the score of "Baseball is a sport" is now 41.

Net and determining how often they match predicates in other resources, an analysis that was first presented in [Havasi *et al.*, 2007].

As described in Section 2.4.2, WordNet and the BSO both describe relationships between their entries that are a subset of the valid ConceptNet predicate types. We begin by equating certain ConceptNet relations to WordNet's pointers and the BSO's qualia, as follows:

| **ConceptNet** | **WordNet** | **BSO** |
|---|---|---|
| IsA | Hypernym | Formal |
| PartOf | Meronym | Constitutive |
| UsedFor | *none* | Telic |

In addition to finding equivalent types of relationships, we also want to be able to say when nodes in the different semantic networks are equivalent. This is not an exact process, particularly because the different networks express information at different granularities.

In WordNet, for example, the nodes of the network are *synsets*, or sets of disambiguated word senses whose meanings are assumed to be equivalent. Every word sense is associated with a single part of speech. A given word, such as "bill", can correspond to a large number of distinct word senses. This breaks up information differently than ConceptNet, which does not disambiguate ambiguous words, and groups together words and phrases that have common word stems instead of common meanings.

The BSO also distinguishes word senses with different parts of speech, and sometimes also distinguishes multiple senses with the same part of speech, but generally uses many fewer senses for each word than WordNet does. In the BSO, also, "lexical" nodes that represent words and phrases of natural language get their properties from higher-level "ontological" nodes caled *types*, which serve a purpose similar to WordNet's synsets.

These different levels of granuality can be reconciled with a slight loss of precision. For all practical purposes, the relations in ConceptNet constrain the parts of speech of

their arguments, so in particular predicates we can distinguish noun phrases from verb phrases. IsA and PartOf take two noun phrases as their arguments, while UsedFor takes a noun phrase and a verb phrase.

We can then identify nodes in the other resources with ConceptNet's concepts, in the same way those concepts were created in the first place: by normalizing text with the Porter stemmer. Given a ConceptNet predicate, we have two concepts, represented by their stemmed text, and an inferred part of speech for each one. If the BSO contains a lexical item of the correct part of speech that stems to the same text as one of the ConceptNet concepts, we consider the BSO node's *type* and the concept to be related. (Recall that lexical items in the BSO do not have properties, only types do.) Similarly, for any WordNet synset containing a word sense that stems to the same text, and has the appropriate part of speech, we consider that synset and the concept to be related.

We can then determine whether a predicate in ConceptNet corresponds to an equivalent fact in the BSO or WordNet, by determining whether there exists a BSO qualia or WordNet pointer that connects two nodes that are related, respectively, to the two concepts in the predicate.

As an example, consider a predicate in ConceptNet representing the statement "A fork is for eating things". This would be represented in this document's usual notation as (UsedFor ⟨fork⟩ ⟨eating things⟩) or (UsedFor ⟨fork⟩ ⟨eat⟩), which are equivalent.

The phrase "a fork" provides a very straightforward association between ConceptNet and the BSO. The phrase's stem is `fork`. There is a lexical item in the BSO, "fork", that also has the stem `fork`, and whose type, incidentally, is *Fork*. So the concept ⟨fork⟩ is related to the BSO type *Fork*.

"Eating things" has the stem `eat`, which it shares with the BSO lexical item "eat", whose type is *Eat Activity*. This relates the concept ⟨eat⟩ to the type *Eat Activity*.

If the BSO contains a telic qualia (the counterpart of UsedFor) that relates *Fork* to *Eat Activity*, then this ConceptNet predicate is represented in the BSO. In this case, it isn't: the telic of *Fork* is *Assistance Activity*, not *Eat Activity*.

Now, for any predicate in ConceptNet, we can look for a corresponding statement

in WordNet or the BSO, with one of three possible outcomes:

- **No comparison**: For one or both of the concepts, we can find no nodes in the other resource that are related according to our definition. This does not necessarily indicate that the two resources disagree, but that we don't know how to map this information from ConceptNet into the other resource.

- **Hit**: We can find related nodes for both concepts, and the nodes are connected by the appropriate relationship.

- **Miss**: We can find related nodes for both concepts, but they are not connected by the appropriate relationship.

The criterion for determining whether "a relationship exists" does not require the relationship to be expressed by a single pointer or qualia. For example, the only direct hypernym of the first sense of "dog" in WordNet is "canine", but we want to be able to match more general statements such as "a dog is an animal". So instead, we check whether the target database contains the appropriate relation from the first concept to the second concept *or* to any ancestor of the second concept under the IsA relation (that is, the hypernym relation or the formal qualia). Under this criterion, ConceptNet's (IsA ⟨dog⟩ ⟨animal⟩) matches against WordNet, as WordNet contains a noun sense of "dog" that has a hypernym pointer to "canine", and a series of hypernym pointers can be followed from "canine" to reach a sense of "animal". This process allows many reasonable matches that could not otherwise be made, and uses the hierarchical structures of WordNet and the BSO as they are intended.

We ran this evaluation independently for IsA, UsedFor, and PartOf predicates, against each of WordNet and the BSO (except that it is not possible to evaluate UsedFor against WordNet). As a control to show that not too many hits arose from random noise, we also tested "randomized IsA predicates". These predicates were created by making random IsA predicates out of the shuffled arguments of the IsA predicates we tested, so that these predicates would express nonsense statements such as "soy is a kind of peninsula". Indeed, few of these predicates were hits compared to

| Resource | Type | Hit | Miss | No comparison |
|---|---|---|---|---|
| WordNet | IsA | 2530 | 3065 | 1267 |
| WordNet | PartOf | 653 | 1344 | 319 |
| WordNet | Random | 245 | 5272 | 1268 |
| BSO | IsA | 1813 | 2545 | 2044 |
| BSO | PartOf | 26 | 49 | 2241 |
| BSO | UsedFor | 382 | 1584 | 3177 |
| BSO | Random | 188 | 4456 | 2142 |

Table 3.3: The results of the comparison. A "hit" is when the appropriate concepts exist in the target database and the correct relationship holds between them, a "miss" is when the concepts exist but the relationship does not hold, and "no comparison" is when one or both concepts do not exist in the target database.

real ConceptNet predicates, even though IsA predicates are the most likely to match by chance. Table 3.3 presents the results, and Figure 3-4 charts the success rates for each trial (the ratios of hits to hits plus misses).

A Pearson's chi-square test of independence showed that the difference in the hit vs. miss distribution between the real predicates and the randomly-generated ones is very statistically significant, with $p < 0.001$ ($df = 1$) for each relation type. WordNet has $\chi^2 = 2465.3$ for IsA predicates and $\chi^2 = 1112.7$ for PartOf predicates compared to random predicates; the BSO has $\chi^2 = 1834.0$ for IsA, $\chi^2 = 159.8$ for PartOf, and $\chi^2 = 414.7$ for UsedFor compared to random predicates.

### 3.4.2 Discussion

As a resource, ConceptNet differs from most available corpora in the nature and structure of its content. Unlike free text corpora, each sentence of OMCS was entered by a goal-directed user hoping to contribute common sense, resulting in a wealth of statements that focus on simple, real-world concepts that often go unstated.

This comparison has shown that our information frequently overlaps with two expert-created resources, WordNet and the Brandeis Semantic Ontology, on the types of predicates where they are comparable. The goal of ConceptNet is not just to emulate these other resources, though; it also contains useful information beyond what is found in WordNet or the BSO. For example, many "misses" in our evaluation are useful statements in ConceptNet that simply do not appear in the other resources

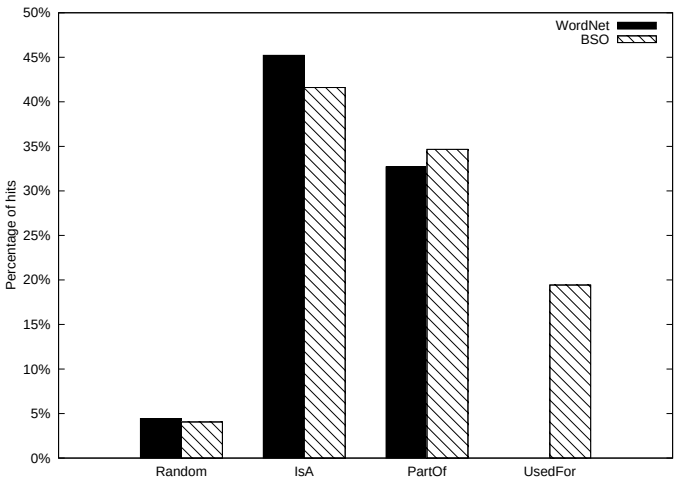


Figure 3-4: When ConceptNet predicates can be mapped onto relations between WordNet and BSO entries, they match a significant percentage of the time.

we evaluated it against, such as "sauce is a part of pizza", "a son is part of a family", and "weekends are used for recovery".

Those kinds of details, based around the way that people see the world, make ConceptNet 3 a good basis for making new inferences about common sense. The next chapter will explain AnalogySpace, one method for making those inferences, which together with human interaction completes a feedback loop that adds more knowledge to ConceptNet.

# Chapter 4

# AnalogySpace

As discussed in previous chapters, the ability to find similarities and analogies in ConceptNet is important to many applications, particularly the Open Mind Commons interface. When the cumulative analogy model (see Section 2.3.2) of similarity and analogy is implemented as an SQL query over ConceptNet, however, it does not scale well. When there are hundreds of thousands of common sense predicates to work with, it requires either performing a search operation that does not complete quickly enough to run interactively on a Web site, or reducing the number of nearest neighbors used to the point that it seriously degrades the quality of the analogies.

This chapter presents a representation of ConceptNet that allows similarity and related operations to be expressed in terms of linear algebra. This allows the operations to be optimized using known linear algebra techniques, particularly truncated singular value decomposition (SVD) [Wall *et al.*, 2003], a form of principal component analysis.

Analyzing ConceptNet with SVD has the interesting side-effect of revealing the principal components of the knowledge in ConceptNet, and making a map of all its concepts in terms of these components. This representation, called AnalogySpace, helps people to visualize the structure of ConceptNet and allows the computer to work with a more intuitive, generalized idea of similarity.

## 4.1 The concept/property representation

The *similarity* between two concepts $a$ and $b$ measures how many predicates about $a$ are also true when $a$ is replaced by $b$. If a predicate changes from true to false when $a$ is replaced by $b$, it counts against similarity. A concise way to calculate this similarity function, $\text{sim}(a, b)$, is to find the number of properties that $a$ and $b$ have in common, minus the number where they explicitly differ.

It will help to define this idea of a "property" more formally. A property is something that, when combined with a concept, forms a predicate.

Because predicates can be expressed as the ordered triple

$$(concept_L, relation, concept_R),$$

a property can be expressed as an ordered pair, either as $(concept_L, relation)$ or $(relation, concept_R)$. These are distinct kinds of properties, called *left properties* and *right properties* respectively. A left property can be made into a predicate by filling in its $concept_R$, and a right property can be made into a predicate by filling in its $concept_L$.

We can define these functions for talking about concepts and properties mathematically:

$\text{pred}(c, p)$ is defined as the predicate in the knowledge base where concept $c$ fills the unfilled slot of property $p$. For most combinations of $c$ and $p$, there will be no such predicate, so consider $\text{pred}(c, p)$ to signify a "null predicate" in that case.

$\text{value}(pred)$ is the truth value of a predicate, which is equal to the predicate's polarity (1 or -1) if the predicate exists. If $pred$ is the null predicate, its value is 0.

Using these definitions, we can express the similarity function as a sum that compares all possible properties:

$$sim(a, b) = \sum_p \text{value}(\text{pred}(a, p)) \cdot \text{value}(\text{pred}(b, p))$$

## 4.2 Similarity is a linear operation

We can look at ConceptNet from the point of view of linear algebra, by expressing all of ConceptNet as an $m \times n$ matrix $A$, where the $m$ rows are the concepts (named $c_1 \ldots c_m$) and the $n$ columns are properties ($p_1 \ldots p_n$). These concepts and properties can also be seen unit vectors that form a basis: the space of concepts is made of the unit vectors $\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_m$, and the space of properties is made of the basis vectors $\hat{\mathbf{p}}_i, \ldots \hat{\mathbf{p}}_j$. As an example, the first concept in order, $c_1$ (suppose it represents "aardvark") can be expressed as the vector $\hat{\mathbf{c}}_1 = [1\ 0\ 0\ 0\ \ldots]^T$.

The entry $A_{ij}$ in the ConceptNet matrix is equal to value(pred($c_i, p_j$)). Row $i$ of the matrix is equal to $A^T\hat{\mathbf{c}}_i$, a vector which expresses concept $c_i$ as a combination of unit properties, and column $j$ similarly equals $A\hat{\mathbf{p}}_i$, expressing property $p_j$ as a combination of unit concepts. (We could also multiply A on both sides by a unit concept and a unit property, as in $\hat{\mathbf{c}}_i^T A\hat{\mathbf{p}}_j$, and this is the same as value(pred($c_i, p_j$)) because it is equivalent to looking up an entry in the matrix.)

The above formula for similarity can now be expressed as a vector equation. The similarity between two concepts is their dot product in the space of properties:

$$\text{sim}(c_i, c_j) = (A^T\hat{\mathbf{c}}_i) \cdot (A^T\hat{\mathbf{c}}_j) = \hat{\mathbf{c}}_i^T AA^T\hat{\mathbf{c}}_j = (AA^T)_{ij}$$

This means that we can express similarity as a product of matrices. $AA^T$ is a matrix of similarities between all pairs of concepts, based on the properties they have in common. Dually, $A^TA$ is a matrix of similarities between properties based on the concepts they have in common. Similarity is therefore a linear operation that we can model using linear algebra.

This representation does not yet make similarity any easier to compute. Calculating $AA^T$ is at least as costly as calculating sim($c_i, c_j$) for every pair of concepts in ConceptNet. However, linear algebra provides a tool that we can use to approximate this operation, and to discover other useful results along the way.

## 4.3 Singular value decomposition

Any matrix $A$ can be factored into an orthonormal matrix $U$, a diagonal matrix $\Sigma$, and an orthonormal matrix $V^T$. The operation that does so is known as *singular value decomposition*. In the context of natural language processing, SVD is most often used to classify documents using latent semantic analysis [Deerwester *et al.*, 1990], but here we are using SVD for a different purpose.

This factorization, $A = U\Sigma V^T$, allows the matrix $A$ to be built from a sum of $r$ outer products of column vectors from $U$ and row vectors from $V$, weighted by the singular values $\sigma_i$ that appear on the diagonal of the matrix. Here, $r$ is the rank of the matrix, or equivalently the number of non-zero values in $\Sigma$.

$$A = \sum_{i=1}^{r} u_i \sigma_i v_i^T$$

Typically, the matrices are arranged so that the singular values in $\Sigma$ are ordered from largest to smallest. This allows the SVD to be approximated by a *truncated SVD*, a product of smaller matrices. Let $U_k$ contain the first $k$ columns of $U$, $V_k^T$ contain the first $k$ rows of $V^T$, and $\Sigma_k$ contain the first $k$ singular values in $\Sigma$. Then

$$A_k = U_k \Sigma_k V_k^T$$

where $A_k$ is the rank-$k$ matrix that best approximates $A$ by a least-squares measure [Wall *et al.*, 2003].

Let us run a truncated SVD on ConceptNet, where $k = 50$ (a value that I have found to be reasonable through experimentation). The resulting $A_k$ is a rank-50 approximation to $A$ that we can store as a product of three smaller matrices. Since we never need the un-truncated results of the SVD (and it would not even be feasible to compute them), let us remove the subscripts on those variables from here on: $A_k = U\Sigma V^T$.

Besides just being the pieces of an approximation, these matrices carry a meaning of their own. $U$ and $V$ define a set of 50 orthogonal axes, forming a 50-dimensional

vector space of data that we can call AnalogySpace. $U$ expresses each axis as a linear combination of concepts, while $V$ expresses each axis as a linear combination of properties.

So while concepts and properties started out as incompatible types of data, the SVD turns both concepts and properties into vectors in AnalogySpace. One vector in this space represents the concept "dog", for example, and that same vector represents the linear combination of properties that dogs have.

Let us now represent this idea in our notation. Just like $A\hat{\mathbf{p}}_i$ represents a property as a vector of concepts, and $A^T\hat{\mathbf{c}}_i$ represents a concept as a vector of properties, we can now represent both concepts and properties as vectors over the 50 orthogonal axes of AnalogySpace. Define $\mathbf{s}(c_i)$ and $\mathbf{s}(p_i)$ to represent both concepts and properties as vectors in the new basis of AnalogySpace. These vectors can be found by transforming the input vectors with the matrices $U$ and $V$:

$$\begin{aligned} \mathbf{s}(c_i) &= U^T\hat{\mathbf{c}}_i \\ \mathbf{s}(p_i) &= V^T\hat{\mathbf{p}}_i \end{aligned}$$

## 4.4 Weighting and normalizing

So far, we have been treating all predicates with equal weight. A natural way to improve the results is to weight them by their ConceptNet score, which measures their reliability. Again, high scores indicate that predicates have been verified or independently stated by multiple users.

To change the entries in $A$ to reflect the predicates' scores, we redefine the function value($pred$) that defines its entries, by multiplying the predicate's polarity by its score, as long as the score is non-negative. The result of value($pred$) can now be any integer, not just -1, 0, or 1.

It's important to remember the distinction between *score* and *truth value* here. A statement with a zero or negative score is completely unreliable, so its entry in the matrix should be 0 whether its truth value is positive or negative. A statement with a

negative *truth value*, however, can have a positive score: "Pigs can't fly" is a reliable negative statement, and it has a score of 2 in the current database, so the entry in the $A$ matrix indexed by ⟨pig⟩ and (CapableOf ⟨fly⟩) is -2.

It is also useful, at this point, to adjust for a bias in $A$ that will skew the results of the SVD. ConceptNet inherently includes more information about some concepts than others: there are hundreds of predicates involving the concept ⟨person⟩, and only a few involving the concept ⟨turtle⟩. This makes the similarity function tend to return values of higher magnitude for ⟨person⟩ than ⟨turtle⟩, either in the positive or negative direction. The result is that ⟨tortoise⟩ appears more similar to ⟨person⟩ than ⟨turtle⟩, simply because ⟨person⟩ has more properties to line up with.

This can be fixed by normalizing the rows of the $A$ matrix – that is, by dividing each row by its Euclidean norm, turning it into a vector of length 1. This, however, creates an opposite problem: concepts with a single property get magnified out of proportion. One property can appear as significant as ConceptNet's entire body of knowledge about people.

To prevent low-frequency concepts from being magnified unnecessarily, the concepts can be given a minimum magnitude before normalizing them. One way to do this is to add a unique component of a fixed magnitude $m$ to each concept vector. These components will not correlate with anything else, so they will not affect the directions of vectors in the SVD result, but they will ensure that no concept can have a magnitude smaller than that of the added component. As a practical implementation, these additional components can be simulated by simply dividing each row $A_i$ of the matrix by $\sqrt{A_i \cdot A_i + m^2}$, instead of by $\sqrt{A_i \cdot A_i}$. This method is used inside ConceptNet with $m = 5$.

After normalizing by concepts, call the resulting concept/property matrix $\hat{A}$. This will be used in place of $A$ from here on, in similarity and other results. Notice that $\hat{A}\hat{A}^T$ is very similar to the covariance matrix of concepts (if it were fully normalized instead of semi-normalized, it would be the same). The similarity between concepts according to this matrix is also close to their cosine similarity (defined as the cosine of the angle between two concept vectors), differing only in the effect of the added

components:

$$\hat{A}\hat{A}^T \approx \frac{\text{sim}(c_i, c_j)}{||A^T\hat{\mathbf{c}}_i||\,||A^T\hat{\mathbf{c}}_j||} = \frac{(A^T\hat{\mathbf{c}}_i)\cdot(A^T\hat{\mathbf{c}}_j)}{||A^T\hat{\mathbf{c}}_i||\,||A^T\hat{\mathbf{c}}_j||}$$

### 4.4.1 Canonical directions

In the results of an SVD, there is no reason to prefer a particular principal component over its negation. These components are eigenvectors, after all, and eigenvectors are unaffected by scalar factors. Normalizing the eigenvectors into unit vectors still leaves two equally good possibilities, $\mathbf{v}$ and $-\mathbf{v}$, for each eigenvector. The *las2* algorithm, as implemented in SVDLIBC, tends to choose between the two directions arbitrarily, and it can even make different choices when run on the same data.

To impose consistency between multiple invocations of the SVD, we add an additional constraint: the concept ⟨person⟩ must be in the positive direction on every axis. If *las2* returns a principal component with a negative ⟨person⟩ term, then that component is replaced by its negation.

## 4.5 Implementation

The contents of the ConceptNet matrix $\hat{A}$ are, of course, very sparse. Most of its entries are 0, and should remain 0 because it makes no sense for certain concepts and properties to be related. The SVD, then, needs to be run over a sparse representation of the matrix.

ConceptNet's SVD is performed by a wrapper around Doug Rohde's SVDLIBC library [Rohde, 2001], which contains an implementation of the *las2* single-vector Lanczos method, a fast SVD algorithm for sparse data. Sparse SVD methods can introduce error through floating-point imprecision, but *las2* has the advantage that most of the imprecision occurs in the low-order singular values, which are discarded anyway.

With this implementation, it actually takes longer to initially construct the sparse matrix $\hat{A}$, including reading all the necessary data from ConceptNet's database, than it does to compute its SVD. When the SVD results are being used in a context where

ConceptNet is being updated interactively, such as the Open Mind Commons web site, it is efficient enough to simply keep the matrix in memory and update it incrementally as predicates in ConceptNet are added or updated. A background process can then periodically recalculate the SVD.

The columns of the sparse matrix represent all concepts that occur in at least 5 predicates, and the rows represent all properties that occur at least once in a predicate with a positive score and include a concept that occurs at least 5 times. If this would lead to a row or column of all zeros (for example, if a concept never appears together with a property that is significant enough to be represented in the matrix), that row or column is left out of the matrix, as it might otherwise cause SVDLIBC to return erroneous results. Currently, this results in an SVD that is run over a matrix of 69236 properties by 12671 concepts, containing 219474 predicates.

### 4.5.1 Incorporating IsA inference

An advantage of AnalogySpace is that it is easy to augment it with additional information to be used in making analogies. , we would like to be able to make inferences that move information across IsA links – there are many cases in which a concept should pass on its properties to its children under the IsA relation, and if several objects with the same IsA parent have the same property, then the parent might have that property as well.

This does not inherently occur in AnalogySpace. It is possible for the concept ⟨dog⟩ to be nowhere near the property (IsA, ⟨dog⟩), and therefore nowhere near concepts that are types of dogs. It can be beneficial to remedy this in making analogies. The way to do this is to add "identity" predicates that connect each concept to its own IsA properties.

When interpreted in natural language, identity predicates express statements such as "a dog is a dog". These are not meant to be useful statements, or even to act as assertions of common sense at all – in fact, predicates that relate a concept to itself are usually disallowed from ConceptNet as being uninformative. In this case, though, we have a reason to add these virtual predicates into the SVD, which is that they

produce useful similarities with other predicates. Think of an identity predicate as not expressing the assertion "A dog is a dog", but the meta-assertion "Subtypes of dogs should inherit properties from dogs". To encourage properties to be inherited, we can give the identity predicates a moderately high score, such as 5.

Incidentally, this removes the need for the meaningless components that were being added earlier to ensure that concept vectors had a minimum magnitude before being normalized. If every concept has an IsA property pointing to itself with score 5 (actually two of them, since one is a left property and the other is a right property), then every concept has a minimum magnitude, and these properties are in fact different properties for every concept.

There is one situation where adding identity predicates is detrimental, however, and that is in inferring IsA statements themselves. If the cumulative analogy process tries to propagate identity predicates to similar concepts, it breaks the abstraction that makes identity predicates work. Identity predicates applied to other concepts are no longer identity predicates – they are just IsA predicates which will be false half of the time.

The following example shows how such an inference can go wrong:

- A piano is a musical instrument.
- A musical instrument is a musical instrument (identity).
- Therefore, a piano is similar to a musical instrument.
- A piano is a piano (identity).
- Therefore, infer that a musical instrument is a piano (not generally true!)

I currently deal with this by keeping two SVDs in memory. One SVD is used only for inferring IsA predicates. This SVD does not include identity predicates, and adds virtual components that ensure a minimum magnitude before normalizing, as described before. The other SVD includes identity predicates, and is used for inferring all types of predicates *except* IsA predicates. The results that will be presented from here on correspond to the SVD that uses identity predicates.

Another way to deal with this, which I have not used, would be to make identity predicates use a new relation which acts like IsA but is not equal to it. Suppose there is a relation called "InheritsFrom" which appears everywhere that IsA appears, but also appears as identity predicates connecting every concept to itself. Then InheritsFrom has the desired effect of propagating other properties through similarity, and the problem above can be prevented by disallowing attempts to create new InheritsFrom predicates through inference (they can only be created when a new IsA predicate is created)

Adding identity predicates leads to the final version of the procedure for creating AnalogySpace. Begin by defining a helper procedure, Add-Entry:

### The Add-Entry procedure

Take as input the 4-tuple $(c, p, value, M)$, where $c$ is a concept, $p$ is a property, $value$ is a number, $M$ is a sparse matrix, and the rows and columns of $M$ are augmented with lists that assign a label to each index.

1. Find a row of $\hat{A}$ labeled with $c$, or add a new one if none exists. Call the index of this row $i$.

2. Find a column of $\hat{A}$ labeled with $p$, or add a new one if none exists. Call the index of this column $j$.

3. Set $M_{ij} := value$.

Now the Create-AnalogySpace procedure can be defined.

### The Create-AnalogySpace procedure

Take as input a minimum concept degree $d$, a number of dimensions $k$, and a minimum concept magnitude $m$.

1. Create an empty sparse matrix $A$, augmented with lists of labels for its rows and columns.

2. For each predicate *pred*, with score $> 0$, that connects concepts of degree $\geq d$:

   (a) Let $value := \text{score}(pred) \cdot \text{polarity}(pred)$.

   (b) Break up *pred* into a left concept $c_L$ and a right property $p_R$.

   (c) Run Add-Entry($c_L$, $p_R$, *value*, $A$).

   (d) Break up *pred* into a right concept $c_R$ and a left property $p_L$.

   (e) Run Add-Entry($c_R$, $p_L$, *value*, $A$).

3. Copy $A$ to $A'$, where identity predicates will be added.

4. For each concept $c$ of degree $\geq d$:

   (a) Create the properties $p_L = (c, \text{IsA})$ and $p_R = (\text{IsA}, c)$.

   (b) Run Add-Entry($c$, $p_L$, $m$, $A'$).

   (c) Run Add-Entry($c$, $p_R$, $m$, $A'$).

5. Create $\hat{A}$ by dividing each row $A_i$ in $A$ by $\sqrt{A_i \cdot A_i + m^2}$.

6. Create $\hat{A}'$ by dividing each row $A'_i$ in $A'$ by $||A'_i||$.

7. Find $U$, $\Sigma$, and $V$ by decomposing $\hat{A}$ using *las2*, stopping after $k$ dimensions have been found. Repeat to find $U'$, $\Sigma'$, and $V'$ from $\hat{A}'$.

8. For each column $U_c$ in $U$, if the entry of $U_c$ labeled "person" is negative, negate all entries in $U_c$ and $V_c$. Repeat with $U'$ and $V'$.

If a predicate is added or changed later, add it to the matrices using step 2. To enter it in $\hat{A}$ and $\hat{A}'$, scale the rows that have changed according to their new norms. When a sufficient number of new predicates have been added, run steps 7 and 8 again.

The resulting singular value decompositions, $(U, \Sigma, V)$ and $(U', \Sigma', V')$, can then be used to find similarities and analogies by calculating dot products between their rows. For the reasons described on page 4.5.1, the $(U', \Sigma', V')$ version should not be used for inferring IsA predicates.

## 4.6 Results

### 4.6.1 Generalized similarity

As AnalogySpace is an orthogonal transformation of the original concept and property spaces, dot products in AnalogySpace should approximate dot products in the original spaces. This fact can be used to compute similarity between concepts or between properties in AnalogySpace. I call the result *generalized similarity*.

Not only is generalized similarity easier to compute than the original similarity function, it can actually be more informative. It takes into account not only whether two concepts have properties exactly in common, but whether they have properties that are *similar* (which, dually, is determined by whether those properties are satisfied by similar concepts). This means that generalized similarity is evaluated over multiple levels of reasoning at the same time.

Concepts that ought to be similar, but share no exact properties, get an ordinary similarity value of 0, but their generalized similarity can be a positive number. So generalized similarity allows a useful similarity value to be calculated for any two concepts, not only concepts with exact properties in common.

Just as $AA^T$ is a matrix of similarity for all pairs of concepts, $A_k A_k^T$ is a matrix of generalized similarity. This matrix shows us how to calculate generalized similarity using $U$ and $V$:

$$A_k A_k^T = U\Sigma V^T V \Sigma U^T = U\Sigma I \Sigma U^T = U\Sigma^2 U^T$$

For two concepts $c_i$ and $c_j$, their entry in this matrix – their generalized similarity – is $\Sigma U^T \hat{\mathbf{c}}_i \cdot \Sigma U^T \hat{\mathbf{c}}_j$, or, equivalently, $\Sigma \mathbf{s}(c_i) \cdot \Sigma \mathbf{s}(c_j)$.

### 4.6.2 Automatic analogies

What gives AnalogySpace its name is the way it naturally produces new hypotheses by making analogies to similar objects, much like cumulative analogy. In AnalogySpace, analogy becomes as easy as similarity: while similarities come from dot products

between two concepts or two properties, analogies come from dot products between a concept and a property.

Remember that the factors of the SVD multiply to give a reduced-rank approximation to $\hat{A}$: $\hat{A}_k = U\Sigma V^T$. $\hat{A}_k$ contains a "smoothed" version of ConceptNet, described by the 50 principal components. If you look up a concept as a row of $\hat{A}_k$, you get a weighted list of properties that the concept "should" have according to those principal components. It gets assigned those properties based on the properties that similar concepts have, which is similar to how analogies work in the Learner model, except without the limitation that only a certain number of nearest neighbors participate in the analogy.

The value of an analogy between concept $c_i$ and property $p_j$ can be found, then, by indexing into $\hat{A}_k$:

$$\hat{\mathbf{c}}_i^T \hat{A}_k \hat{\mathbf{p}}_j = \hat{\mathbf{p}}_j^T U\Sigma V^T \hat{\mathbf{c}}_i = \mathbf{s}(p_j)^T \Sigma \mathbf{s}(c_i)$$

This is essentially a dot product between the concept $\mathbf{s}(c_i)$ and the property $\mathbf{s}(p_j)$, except that the components are weighted by $\Sigma$. Selecting the $n$ highest-valued analogies that do not correspond to existing predicates in ConceptNet yields a list of $n$ new inferences.

One can visualize how this works spatially. In the concept/property representation, concepts are sums of unit properties, and properties are sums of unit concepts. These sums still approximately hold when the concepts and properties are transformed into the same space by a truncated SVD, so concepts will tend to point in the same direction as the properties they have, and vice versa. Then, if a concept and property point in approximately the same direction but have no predicate linking them, the system can hypothesize that that predicate should exist.

In Section 5, the performance of this analogy procedure is evaluated by having users check its results.

### Partial inferences

The user interface of Open Mind Commons requires not just the kind of inferences that come from the "automatic analogy" process described above; it also requires *partial inferences*, in which one of the concepts is left unspecified, which are presented to users as fill-in-the-blank questions instead of yes/no questions. The purpose of partial inferences is to prompt the user for under-represented types of information.

AnalogySpace can produce partial inferences by aggregating the values of the analogies it produces. Let a *property type* be defined as a property with the concept removed, so the only information it specifies is the relation and whether it is a left or right property. We can find out how strongly a property type is inferred for a given concept, instead of a single property, by adding up the analogy values of all properties of that type:

$$\text{partialValue}(c_i, type) = \sum_{p_j \in type} (\hat{A}_k)_{ij}$$

However, this is not quite the right information. Since $\hat{A}_k$ is an approximation to the information in ConceptNet, the properties that are already most strongly represented already in ConceptNet will tend to get the highest weights. The next step, then, is to discount these values based on how many predicates in ConceptNet already relate that concept to that property type. Call this number $n$, and let the score of a partial inference decrease exponentially as $n$ increases:

$$\text{partialScore}(c_i, type) = \frac{\text{partialValue}(c_i, type)}{(0.8)^n}$$

The base of 0.8 is a parameter that can be adjusted, determining whether this procedure favors collecting multiple predicates of the same type, or introducing new property types that AnalogySpace does not suggest as strongly.

### 4.6.3 Eigenconcepts

The operations discussed so far, similarity and analogy, have not depended on concepts' absolute positions in AnalogySpace, but only their positions relative to each other. Each concept is represented by a vector of 50 coordinates in AnalogySpace; these coordinates can be seen as describing the concept in terms of 50 *eigenconcepts* that form the axes of AnalogySpace.

As an inherent result of how the SVD is calculated, matrix $U$ contains the eigenvectors of $AA^T$, the matrix of similarities between concepts; $V$ contains the eigenvectors of $A^TA$, the matrix of similarities between properties; and $\Sigma$ contains the eigenvalues that are shared by both sets of eigenvectors. Because these eigenvectors are vectors in the space of concepts and properties respectively, they can be considered *eigenconcepts* and *eigenproperties*. These eigenconcepts and eigenproperties become the basis vectors $\mathbf{e}_0, \ldots, \mathbf{e}_{49}$ of AnalogySpace. Since the eigenproperties are identical to the eigenconcepts in AnalogySpace, it is reasonable to simply call them all "eigenconcepts".

The coordinates of a concept or property in AnalogySpace can be thought of as expressing that concept or property as a combination of eigenconcepts. Equivalently, the coordinates can be seen as expressing how similar a concept or property is to each eigenconcept.

Although the eigenconcepts can be arbitrary linear combinations of concepts, they turn out to often represent useful real-world categories such as "living things", "actions", "household objects", "desirable things", and so on. Some of the more interesting eigenconcepts are shown below, and the graphs in Appendix C plot the concepts and properties in ConceptNet relative to the first 10 eigenconcepts. Remember that all of these eigenconcepts have been oriented so that ⟨person⟩ is in the positive direction.

| Weight | Concept | Weight | Property |
|---|---|---|---|
| 0.056 | ⟨fulfilment⟩ | 0.984 | (⟨person⟩, Desires) |
| 0.055 | ⟨a good job⟩ | 0.012 | (AtLocation, ⟨home⟩) |
| 0.055 | ⟨respect⟩ | 0.011 | (UsedFor, ⟨fun⟩) |
| 0.053 | ⟨compliment⟩ | 0.010 | (HasProperty, ⟨good⟩) |
| 0.052 | ⟨achieve a goal⟩ | 0.010 | (HasProperty, ⟨important⟩) |
| | ... | | ... |
| -0.059 | ⟨alone⟩ | -0.008 | (⟨drive a car⟩, Causes) |
| -0.059 | ⟨lose friends⟩ | -0.008 | (⟨lie⟩, Causes) |
| -0.056 | ⟨ridicule⟩ | -0.009 | (HasProperty, ⟨bad⟩) |
| -0.056 | ⟨hate⟩ | -0.011 | (⟨person⟩, HasProperty) |
| -0.055 | ⟨lose job⟩ | -0.012 | (IsA, ⟨disease⟩) |

Table 4.1: The largest terms in eigenconcept $\mathbf{e}_0$ ("desirability").

| Weight | Concept | Weight | Property |
|---|---|---|---|
| 0.083 | ⟨breathe air⟩ | 0.934 | (⟨person⟩, CapableOf) |
| 0.081 | ⟨work in an office⟩ | 0.096 | (⟨human⟩, CapableOf) |
| 0.076 | ⟨see with eyes⟩ | 0.060 | (⟨children⟩, CapableOf) |
| 0.074 | ⟨voice opinions⟩ | 0.059 | (UsedFor, ⟨fun⟩) |
| 0.072 | ⟨like to win⟩ | 0.055 | (⟨child⟩, CapableOf) |
| | ... | | ... |
| -0.045 | ⟨live in the ocean⟩ | -0.008 | (IsA, ⟨walk on water⟩) |
| -0.046 | ⟨walk on walls⟩ | -0.009 | (⟨walk on water⟩, IsA) |
| -0.050 | ⟨live forever⟩ | -0.009 | (IsA, ⟨breathe water⟩) |
| -0.059 | ⟨walk on water⟩ | -0.009 | (⟨breathe water⟩, IsA) |
| -0.066 | ⟨breathe water⟩ | -0.011 | (⟨person⟩, Desires) |

Table 4.2: The largest terms in eigenconcept $\mathbf{e}_1$ ("feasibility").

### $\mathbf{e}_0$: Desirability

The most significant eigenconcept, $\mathbf{e}_0$, represents *things people want*. That is, the larger a concept's component in the $\mathbf{e}_0$ direction is, the more desirable it is likely to be. Concepts with negative $\mathbf{e}_0$ components, then, are particularly undesirable.

Table 4.1 shows the concepts and properties that contribute the most weight to $\mathbf{e}_0$ – that is, the most desirable and undesirable concepts, and the properties that most indicate desirability and undesirability, according to ConceptSpace.

### $\mathbf{e}_1$: Feasibility

The next eigenconcept represents *actions that people can do*. To a lesser extent, it represents actions that can be done by certain kinds of people, such as children, or

| Weight | Concept | Weight | Property |
|---|---|---|---|
| 0.093 | ⟨Utah⟩ | 0.246 | (AtLocation, ⟨desk⟩) |
| 0.092 | ⟨Maine⟩ | 0.200 | (⟨cat⟩, AtLocation) |
| 0.091 | ⟨chair⟩ | 0.162 | (AtLocation, ⟨home⟩) |
| 0.083 | ⟨Delaware⟩ | 0.160 | (AtLocation, ⟨city⟩) |
| 0.079 | ⟨zoo⟩ | 0.147 | (AtLocation, ⟨house⟩) |
| | ... | | ... |
| -0.010 | ⟨voice opinions⟩ | -0.005 | (⟨man⟩, CapableOf) |
| -0.011 | ⟨like to win⟩ | -0.008 | (CapableOf, ⟨talk⟩) |
| -0.011 | ⟨see with eyes⟩ | -0.010 | (⟨human⟩, CapableOf) |
| -0.012 | ⟨breathe air⟩ | -0.017 | (⟨person⟩, Desires) |
| -0.012 | ⟨work in an office⟩ | -0.112 | (⟨person⟩, CapableOf) |

Table 4.3: The largest terms in eigenconcept $\mathbf{e}_2$ ("things vs. events").

by things that are similar to people, such as cats or computers. Its largest terms are listed in Table 4.2. Like $\mathbf{e}_0$, the representation of $\mathbf{e}_1$ in property space is dominated by a single term – (⟨person⟩, CapableOf), with a weight of 0.93.

In the appendices, Figure C-1 plots the coordinates of concepts and properties in AnalogySpace, relative to $\mathbf{e}_0$ and $\mathbf{e}_1$.

### $\mathbf{e}_2$: Things versus events

Concepts that fall on the positive side of $\mathbf{e}_2$ tend to be "things", and those on the negative side tend to be events. It is tempting to call this a distinction between nouns and verbs, but those terms would be misleading, as this is a semantic distinction and not a grammatical one. The positive direction of $\mathbf{e}_2$ contains clusters representing places, people, household objects, and animals, while the negative direction consist of actions people can take and events that they want.

Some actions that are strongly associated with a single object, such as "play a guitar", are located in the positive direction, near their associated objects. Negation also puts some things in counterintuitive places: some atypical actions go in the positive direction because people cannot or do not want to do them,[1] and properties that are false for most objects, such as (CapableOf, ⟨talk⟩), go in the negative direction.

Figure C-2 plots $\mathbf{e}_2$ against the "feasibility" axis $\mathbf{e}_1$.

[1]It worries me that ⟨vote⟩ apparently falls in this category.

### More eigenconcepts

After this, major clusters of concepts begin to fall in places other than the axes. This has to happen, because the clusters are not precisely orthogonal to each other. The eigenconcepts may point in directions that are not easily described because they are combinations of slightly related categories of things.

Axis $\mathbf{e}_3$, in the positive direction, contains moderately-sized physical objects, particularly a large cluster of musical instruments such as ⟨flute⟩ and ⟨guitar⟩. Concepts in the opposite direction tend to be places. Axis $\mathbf{e}_4$ contains household objects, which includes many of the things in the $\mathbf{e}_3$ direction, but not musical instruments. Figure C-3 plots $\mathbf{e}_3$ against $\mathbf{e}_4$.

Axis $\mathbf{e}_5$, in the negative direction, contains events that are entertaining, such as ⟨see a movie⟩ and ⟨go to a concert⟩. Figure C-4 compares $\mathbf{e}_1$ and $\mathbf{e}_5$, which between them distinguish four classes of concepts: possible actions, fun actions, impossible actions, and objects that are not actions at all.

$\mathbf{e}_6$ contains buildings, particularly human dwellings in the positive direction. In the opposite direction are things that are not buildings, either because they are larger places (states and countries) or because they are small objects. It is interesting to plot $\mathbf{e}_6$ against $\mathbf{e}_3$ (Figure C-5) to see the different distinctions they make between classes of objects.

Eigenconcepts have less obvious interpretations as they appear later in the list, but continue to distinguish relevant groups of concepts and properties. The two next eigenconcepts are $\mathbf{e}_7$ and $\mathbf{e}_8$, plotted in Figure C-6. Both eigenconcepts mix together too many different kinds of things to have a straightforward interpretation, but between them they create a sort of circular spectrum of nouns, identifying definite clusters of living things, food, furniture, places, and office supplies.

The aforementioned eigenconcepts show how the contents of ConceptNet fall into natural categories of things with similar properties. Looking at extreme values on the axes described by the eigenconcepts can help to highlight and compare these natural categories.

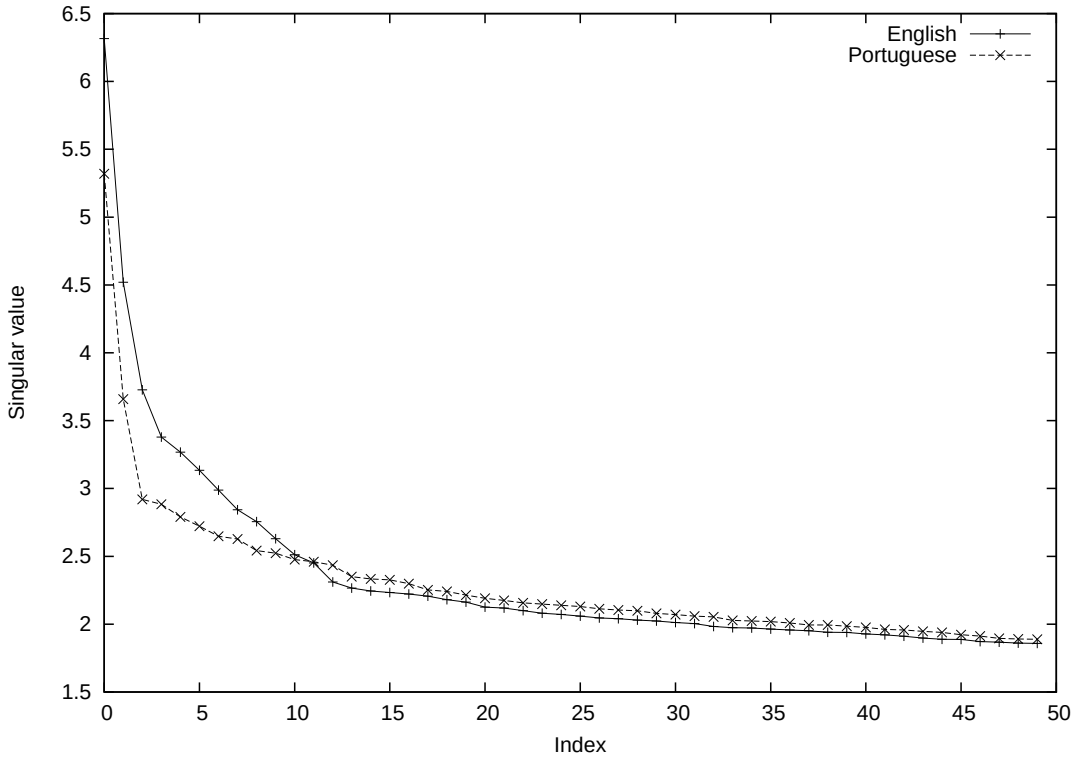


Figure 4-1: The first 50 singular values in English and Portuguese.

### 4.6.4 Significance of principal components

In implementing the SVD, I chose the number of principal components $k = 50$ rather arbitrarily. If there are many more than 50 meaningful factors that explain similarities between concepts, this is okay; part of the reason for truncating the SVD at 50 components is for speed. It is important, however, to confirm that there are *at least* 50 factors that convey meaningful information and not just random noise.

A plot of the singular values (Figure 4-1) shows a slope change after 12 singular values in English and 13 in Portuguese, a sign that there may be some special significance to the first 12 or so eigenconcepts, but this does not prevent the later eigenconcepts from providing useful information. I have found by experimentation that AnalogySpace produces more useful analogies with $k = 50$ than it does with $k = 20$. The slope change here distinguishes particularly interesting eigenconcepts from ordinary ones, not ordinary eigenconcepts from random noise.

Sources such as [Harris, 1975, p. 177] discuss a straightforward way to estimate

the number of statistically significant factors in the results of a principal component analysis. If $s_i^2$ is the variance of row $i$ of the covariance matrix $AA^T$, and $m$ is the total number of rows, then the number of significant factors can be estimated as the number of principal components whose eigenvalue $\sigma^2$ is greater than the average variance, $\sum_i s_i^2/m$.

The *las2* algorithm in SVDLIBC computes, at most, the first 429 singular values for ConceptNet, with the 429th one having $\sigma^2 = 1.93$. Singular values beyond that are presumably lost in floating-point imprecision. All the values returned by *las2* are far greater than the average variance of 0.08, showing that ConceptNet has more than enough significant components.

### 4.6.5 Ad-hoc categories

The vectors that are compared in AnalogySpace do not have to correspond to existing concepts or properties in ConceptNet. In some cases, it is useful to construct a new vector in AnalogySpace, such as a synthetic concept made of a sum of concepts. Such a vector can represent an *ad-hoc category* that is useful in an application. The idea of ad-hoc categories is similar to the idea of a "mini-document" in LSI, a collection of a few words that represents, for example, a search query.

As an example, we can create a category of furniture from the sum of concepts $\langle\text{chair}\rangle + \langle\text{table}\rangle + \langle\text{desk}\rangle + \langle\text{bed}\rangle + \langle\text{couch}\rangle$. If we add up the ConceptSpace vectors for these concepts, we get a combined vector in ConceptSpace, $\mathbf{s}_{cat}$, that represents the category as a whole. (We could also create a sum of unit concepts, $\mathbf{c}_{cat}$, and transform it into AnalogySpace by multiplying it by $U$, to get the same result.)

One simple way to use an ad-hoc category is to look for its strongest similarities to existing concepts, and thus discover more objects that seem to belong in the category. More applications are explored in Chapter 6.

To look for similarities, we use $\mathbf{s}_{cat}$ in the same way we would use a concept's AnalogySpace vector, $\mathbf{s}(c)$. We can find the similarity of a particular concept $c_i$ to the category by calculating $\Sigma\mathbf{s}(c_i) \cdot \Sigma\mathbf{s}_{cat}$. A vector of similarities over *all* concepts –

call it $\mathbf{c}'$ – can be found as:

$$\mathbf{c}' = U\Sigma^2\mathbf{s}_{cat} = U\Sigma^2U^T\mathbf{c}_{cat}^T$$

The largest components of $\mathbf{c}'$ reflect the objects that are most similar to the category. $\mathbf{c}'$ should look like the original category $\mathbf{c}_{cat}$, except extended to assign a value to all concepts.

For the category of furniture, the most similar concepts include other kinds of furniture and parts of a house, as shown in Table 4.4.

| Weight | Concept |
|---|---|
| 1.218 | chair |
| 1.013 | couch |
| 0.966 | sofa |
| 0.906 | bed |
| 0.761 | floor |
| 0.744 | carpet |
| 0.712 | table |
| 0.682 | living room |
| 0.667 | beanbag chair |
| 0.643 | armchair |

Table 4.4: Concepts that are similar to an ad-hoc category of furniture.

# Chapter 5

# Evaluation

In order to show that SVD-based reasoning produces reasonable, useful inferences, I invited people to participate in an evaluation of its performance. The results show that users consider inferences from AnalogySpace to be true a large percentage of the time, significantly more often than randomly-generated predicates.

## 5.1 Experimental setup

In this evaluation, subjects were presented with a list of statements from various sources. The statements were all expressed as English sentences by the *InferNaturalLanguage* procedure in Open Mind Commons (described in Section 3.3.1), and were selected randomly from four different sources:

- Existing predicates in ConceptNet 3 with a score greater than 1
- Predicates inferred by vanilla AnalogySpace
- Predicates inferred by AnalogySpace, with identity predicates added (as described on p. 56)
- Randomly-generated predicates

Participants were presented with a list of 60 such sentences. Each sentence was accompanied by a drop-down box labeled "Rate...", which the participants would

use to give their opinion of how true the sentence was. The possible options were "Generally true", "Occasionally true", "Don't know / Opinion", "Generally false", and "Doesn't make sense". The participant's choices were submitted via JavaScript as soon as they were made, so that partial results would be available even from users who did not finish the evaluation. If a participant chose one answer and then changed it, the new answer would replace the old one.

The following text appeared above the list:

> The following is a list of 60 statements. Some of them are true statements about common sense knowledge, while others are false or nonsensical. We'd like you to label which statements are which.
>
> For each statement, please rate how true you consider the statement (or whether it doesn't make sense) using the dropdown box to its right. Disregard errors in grammar if you can still make sense out of the sentence.

After answering the questions and clicking "Done", participants then saw a page explaining what the experiment was about, followed by the list of statements, this time sorted according to their source. This page had no effect on the experiment's results, but it allowed participants to conclude for themselves whether the inference procedures had performed well.

## 5.2 Producing statements to evaluate

All statements that participate were asked to evaluate were produced by the *InferNaturalLanguage* procedure, even the statements from ConceptNet which already had natural language representations. The reason for this was to make statements from different sources appear the same. *InferNaturalLanguage* produces some sentences with incorrect grammar, but running it on all predicates makes the quality of the grammar consistent across all the displayed sentences.

The statements from ConceptNet were selected uniformly randomly from all predicates with a score greater than or equal to 2 – that is, all predicates that have been

confirmed by a user other than the user who originally entered them.

The inferred statements were chosen by a weighted random process, designed to emphasize the inferences that have higher dot products in AnalogySpace, as these are the inferences that Commons is most likely to ask its users about, while not presenting the same set of top-ranked inferences to every user. Given an inference with a dot-product score of $s$, it is assigned a value of $s * r^4$, where $r$ is a random value chosen uniformly from $[0, 1)$, and the inferences with the 15 highest values are selected.

The random statements are chosen in such a way that they have a reasonable chance of making grammatical sense. Properties are chosen at random until a left property and right property are found that share the same relation, so that they can be overlapped to form a predicate. The resulting predicate is turned into a sentence with *InferNaturalLanguage*.

## 5.3 Results

238 people participated in the evaluation, with no compensation offered. After discarding results from participants who answered less than half of the questions, there were 195 participants remaining.

As expected, the statements that participants most often considered true were the statements that were already present in ConceptNet. Random statements were rarely considered true, and were most often rated as "Doesn't make sense". The inferred statements were much more highly rated than random predicates, and in fact performed comparably to the existing statements in OMCS. Users gave inferred predicates a positive rating – that is, one of "Generally true" or "Occasionally true" – 68.9% of the time, compared to 80.8% positive ratings received by the existing predicates that the inferences were based on. Figure 5-1 shows a breakdown of the users' responses to the evaluation by the predicate's source.

To compare the average performance of each source of statements, the possible responses were assigned numerical values on the 4-point scale shown in Table 5.1:

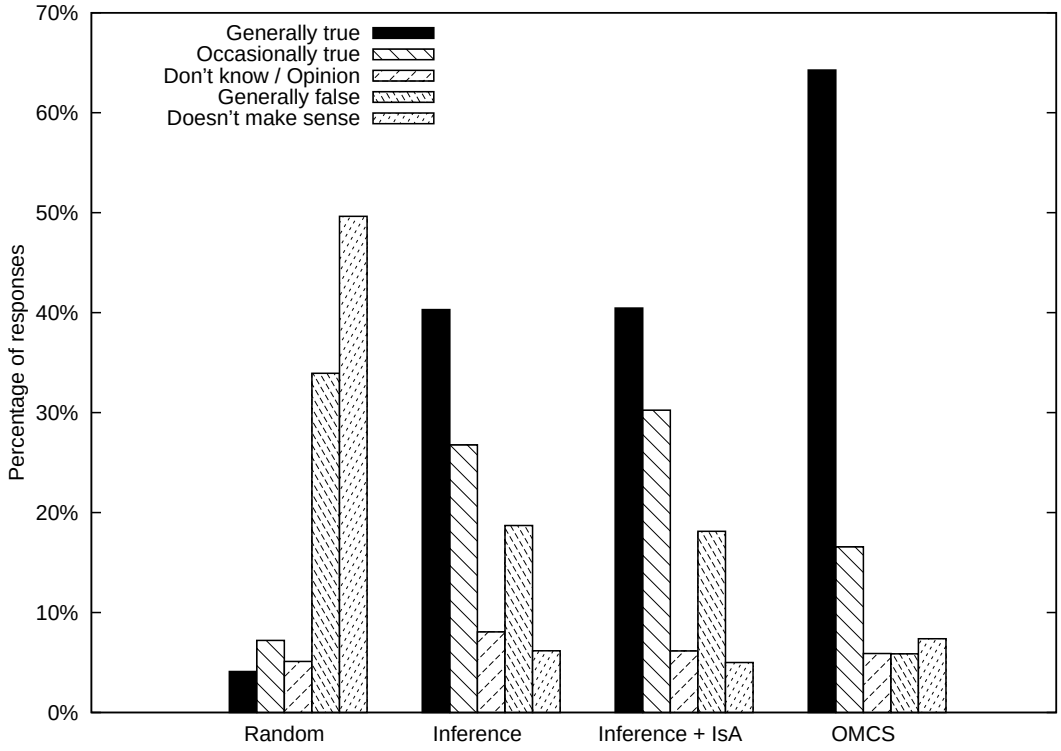


Figure 5-1: The distribution of users' responses to the evaluation.

| Score | Response |
|---|---|
| 2 | Generally true |
| 1 | Occasionally true |
| 0 | Don't know / Opinion |
| -1 | Generally false *or* Doesn't make sense |

Table 5.1: The scale used to assign aggregate scores to responses to the user evaluation.

| Source | Score |
|---|---|
| Random | -0.688 |
| AnalogySpace | 0.820 |
| AnalogySpace + identity | 0.874 |
| ConceptNet | 1.333 |

Table 5.2: Mean scores for each source of predicates in the user evaluation.

With this scale, Table 5.2 shows the mean scores received by each source of statements, averaged across all contributors.

To evaluate the significance of these results, I ran a correlated-samples ANOVA over the mean scores that each participant assigned to each source. Table B.1, in Appendix B, lists these mean scores for each user. The ANOVA showed that it is very statistically significant that there is a difference between the groups ($p < .0001$, $df = 3$).

In order to determine *which* differences were statistically significant, I then ran a Tukey HSD test on the results of the ANOVA. The test calculated that, for a difference to be significant at the $p < .01$ level, the means must differ by at least 0.10.

The difference between AnalogySpace with and without identity predicates, with a difference in means of 0.054, was not enough to be significant. All other differences were significant, which shows in particular that both inference algorithms performed significantly better than random predicates.

# Chapter 6

# Applying AnalogySpace

The properties of AnalogySpace make it useful in ways besides just generating potential inferences. This chapter describes some other ways that AnalogySpace can be applied.

## 6.1 Verifying existing knowledge

The examples of using AnalogySpace so far have focused on using it to produce new inferences, by discarding entries that correspond to existing knowledge. If *only* the entries corresponding to existing knowledge are kept instead, then AnalogySpace can be used to verify existing knowledge.

Because a predicate can be broken down into a left concept and a right property, or a right concept and a left property, each predicate corresponds to up to two entries in the reconstructed ConceptNet matrix $\hat{A}'_k$. (Entries may be missing if they involve a concept or property used too infrequently to appear in the matrix.) We can find these entries in the same way we made analogies before, by taking the weighted dot product of their coordinates in AnalogySpace, $\mathbf{s}(p_j)^T \Sigma \mathbf{s}(c_i)$. The sum of these entries – call it $v$ – is a "verification score" for the predicate, which indicates how well the predicate is supported by other knowledge.

As an example, these five predicates were the most well-supported predicates about the concept ⟨book⟩, receiving the highest verification scores:

- Something you find at school is books ($v = 0.2131$)
- Something you find on your desk is books ($v = 0.1979$)
- You are likely to find a book in a library ($v = 0.1755$)
- Something you find on a table is books ($v = 0.1561$)
- Something you find in a cabinet is books ($v = 0.1162$)

For the concept ⟨book⟩, LocationOf predicates tended to receive high verification scores. The highest-scoring non-LocationOf predicate was "A book is used for learning" ($v = 0.1022$).

The five most dubious predicates about ⟨book⟩, having the lowest non-zero verification scores, are:

- Books taste good ($v = -0.0040$)
- You are likely to find a cover in a book ($v = 0.0013$)
- Books can win prizes ($v = 0.0016$)
- A religious book ($v = 0.0019$)[1]
- Books contain pictures ($v = 0.0019$)

This procedure could be useful to incorporate into a future version of Open Mind Commons, which could use it to draw attention to dubious statements, so that users could either confirm or reject them.

## 6.2 Dealing with systematic vandalism

Nothing prevents Open Mind's contributors from entering inaccurate, nonsensical, or malicious statements, and sometimes they do. Vandalism is a problem faced by many

[1] This was mis-parsed as a CapableOf predicate, with "a religious" as the noun and "book" as the verb.

online projects that are open to the public, and Open Mind is no exception, but the AnalogySpace representation can help to curb it in some cases.

Vandalism in Open Mind can take many forms, many of which turn out to be harmless or self-defeating. Other forms can be easily removed by searching for certain words. Occasionally, more advanced techniques are necessary to sort the good statements from the garbage.

One common form of vandalism is where a user types a statement made of nonsense characters, such as "hfghfghdf", or fills in nonsense characters in a template, such as "An activity something can do is sdf". Statements made completely of nonsense have no effect because they will never parse as anything. Nonsense characters in a template will either be rejected by the shallow parser, or turn into a ConceptNet node of degree 1, which has essentially no effect on inference because it will contribute nothing to similarity.

Other vandalism takes the form of sentences that don't convey knowledge, and may not convey much of anything at all, such as "They can be serving cake there now". These, too, can become irrelevant if the parser fails to parse them, or if one of the concepts turns out to be entirely made of stopwords (such as "they"), at which point it will be rejected from ConceptNet.

Many kinds of vandalism use abusive language, making them easy to spot. ConceptNet 3 automatically rejects concepts containing the word "fuck" or any of various slurs against groups of people. (Statements containing the word "shit", on the other hand, are retained, because they tend to be correct, if crudely stated.)

The vandalism that remains consists of incorrect or unhelpful statements that connect two established concepts, such as "Joe is a cat" or "You are likely to find a weasel in your pants". These statements are typically rare enough that they blend in with other well-meaning but confused statements in ConceptNet. They never achieve high reliability scores, and tend not to correlate with anything, so most of these statements appear as noise from a principal components point of view. In this case, the fact that a truncated SVD maintains correlations while removing noise is very useful, as it discards the information conveyed by random incorrect statements.

Only one form of vandalism actually has a significant effect on SVD-based inference. A few users have spent a great deal of time entering *systematic* nonsense into Open Mind, instead of random nonsense that doesn't connect to anything. One user of the original Open Mind, for example, answered every fill-in-the-blank question with "thermonuclear war", a pattern that was easy to detect and remove. Another user, identified here as User #3595, has a subtler contribution history that illustrates the problems posed by systematic vandalism, along with a solution that AnalogySpace provides.

User #3595 was a prodigious contributor, adding approximately 6500 new statements to Open Mind. Several hundred of these, near the beginning, were valid, useful common sense statements. It seems that the user grew tired of making useful contributions, and possibly became frustrated at the lack of feedback. With increasing frequency, he began entering meaningless, systematic statements involving people's names: "Sheila is late. Ed is late. Brenda is late. Sheila is poor. Ed is poor. Brenda is poor. Sheila is in jail..." Even so, he would then return to adding useful statements at some point. It is possible that the user was confused and not malicious, but either way, he left behind thousands of statements of systematic nonsense to be identified.

Instead of being neutralized by the SVD, these statements were magnified, because they form a very consistent pattern across many concepts. Fortunately, this means that the SVD could also be used to identify these statements. The names and properties that #3595 used comprised an obvious spike in the SVD results, with all of them having AnalogySpace vectors pointing in essentially the same direction (except for some ambiguous names such as "bill").

Simply removing all of his contributions would sacrifice too many valid statements. The key was to identify only the statements that were part of this pattern of nonsense, and remove them. To do so, I only had to search for the predicates entered by user #3595 whose properties were indicated by the SVD to be similar to a signature property of his, such as (LocationOf, ⟨jail⟩). Applying this process a few times allowed me to remove about 2700 nonsense predicates, which I did before running the user test or reporting the other results in this thesis.

## 6.3 Topic detection

AnalogySpace has shown some promising preliminary results when applied to the task of topic detection, using the method of ad-hoc categories described in Section 4.6.5. By creating categories for each possible topic, and categories representing the texts we wish to classify, and comparing the dot products between these categories, we can find which topic corresponds most strongly to each text.

Creating topics does not necessarily require a training set of texts. All we need to do is pick a number of representative concepts that are related to each topic, and add them together to form an ad-hoc category. The following example shows how this can work.

For this example, we will compare three topics:

- *Politics* = ⟨politics⟩ + ⟨vote⟩ + ⟨govern⟩ + ⟨elect⟩ + ⟨state⟩
- *Business* = ⟨business⟩ + ⟨company⟩ + ⟨industry⟩ + ⟨market⟩ + ⟨money⟩
- *Sports* = ⟨sport⟩ + ⟨game⟩ + ⟨ball⟩ + ⟨play⟩ + ⟨win⟩

To create a category from a text, we remove all of its stopwords, find all words and adjacent pairs of words that correspond to concepts in AnalogySpace, and use the sum of those concepts to form an ad-hoc category. This example uses three texts from news articles, found in the appropriate sections of Google News:

1. The eight Democrats seeking their party's U.S. presidential nomination sparred Sunday in a nationally televised debate. VOA's Paula Wolfson reports foreign policy issues topped the event, which was held in the key state of Iowa.[2]
2. Midwest Air Group Inc. said Thursday night that it will accept a cash offer to be acquired by private investment firm TPG Capital, rejecting a revived cash-and-stock offer from AirTran Holdings Inc.[3]
3. Jose Guillen's first-inning homer into the White Sox bullpen was a message for the Chicago relievers to be ready early. Guillen followed the two-run shot with a two-run single in the second, driving in five runs and sparking Seattle's offense in the Mariners' 11-5 romp of the White Sox on Sunday afternoon.[4]

[2]Paula Wolfson, *Voice of America*, August 19, 2007.
[3]Emily Fredrix, *Forbes*, August 17, 2007.
[4]Tim Booth, Associated Press, August 19, 2007.

| Article # | Politics | Business | Sports |
|---|---|---|---|
| 1 | **0.408** | 0.255 | 0.236 |
| 2 | 0.006 | **0.484** | 0.117 |
| 3 | 0.140 | -0.123 | **0.583** |

Table 6.1: AnalogySpace determines the topic areas of three news articles.

When these texts are turned into ad-hoc category vectors, and are compared with the topic vectors for their similarity in AnalogySpace, the correct topic shows up as the most similar in each case, as shown in Table 6.1. This shows the potential for this to be expanded into a more robust topic-detection system, and since little training seems to be necessary, this system could be provided with a set of topics adapted to the task at hand.

This method could also be used with the possible "topics" simply being the entries of ConceptNet, yielding a weighted list of concepts that are semantically associated with the input text, similar to the example in Section 4.6.5. This kind of information can be used to improve the results of predictive text entry and speech recognition [Lieberman *et al.*, 2005].

## 6.4 Semantic spectra

Because AnalogySpace is a continuous space, it can be used in ways that go beyond simply making a discrete choice, as the topic detection example above does. Many applications of common sense knowledge involve measuring the strength with which particular ideas occur in a text. As illustrated in Section 4.6.3 and the graphs in Appendix A, AnalogySpace is particularly good at representing concepts in a spectrum.

Common sense knowledge has frequently been applied to *affect sensing*, or detecting emotions in text [Liu *et al.*, 2003a]. Affect-sensing based on common sense knowledge has been applied in user interfaces that can empathize with the user [Liu *et al.*, 2003a, p. 6], as well as an interface for navigating text by emotion [Liu *et al.*, 2003b]. This is another task that could be facilitated by working in AnalogySpace, because emotions are an example of a semantic spectrum.

If we first identify directions in AnalogySpace that correspond to basic emotions, we can combine these directions into a space representing a range of possible emotions. In fact, we have already seen directions that are similar to emotions and are easy to express when examining the eigenconcepts: the $+\mathbf{e}_0$ direction is "pleasant", while $-\mathbf{e}_0$ is "unpleasant", $-\mathbf{e}_5$ is "enjoyable", and $-\mathbf{e}_1$ – the direction representing actions seen as atypical or infeasible – could be seen as "surprising". There is no reason to limit the vectors that define an emotional space to being eigenconcepts, though, except for familiarity.

With a set of emotional vectors defined, then concepts that occur in text can be projected into that emotional space in an operation as straightforward as matrix multiplication.

## 6.5 Aligning knowledge bases

While ConceptNet contains a wealth of information in both English and Portuguese, the two languages currently act as completely different semantic spaces. Given a concept or set of concepts in English, there is no clear way to find corresponding concepts in Portuguese.

With both languages' ConceptNets represented as vector spaces, however, it could be possible to align these vector spaces with each other. The first step would be to "anchor" a certain number of concepts by finding English and Portuguese concepts that are direct translations of each other. This kind of data could be collected from bilingual contributors as in GlobalMind [Chung, 2006], or it could be imported in large quantities (albeit somewhat noisily) from an electronic multilingual lexicon.

Concepts that are not anchored could then be translated by using nearby anchored concepts as a guide. Similarities could be propagated between languages using the tenet that if two concepts are similar in one language, they should be similar in another, and this would eventually build up a somewhat smooth, locally-consistent mapping between languages.

This could be applied to other tasks besides translation, as well: for example, it

could be used to augment the domain-general ConceptNet with some domain-specific knowledge. I feel that this is a very promising direction for future research using AnalogySpace.

## 6.6 Conclusion

The AnalogySpace representation allows the information in ConceptNet to be used in a number of new ways. The implementation of cumulative analogy as a vector operation within AnalogySpace is an elegant way to create new hypotheses that can be fed back into ConceptNet. Other operations on AnalogySpace can be used introspectively, to refine ConceptNet and improve its quality, or externally, to make the knowledge in ConceptNet available in a form that is useful to applications.

The power of AnalogySpace comes from the way it exploits large-scale patterns in ConceptNet. By identifying the principal components of ConceptNet, we find the "signal" representing the collective knowledge of thousands of contributors, while abstracting away the noise.

# Appendix A

# Parsing patterns

## A.1 Top-level English parsing patterns

In the process of building ConceptNet, statements in the Open Mind Common Sense corpus are checked against these patterns, in order, until one matches, so that earlier patterns take precedence over later ones. When a match is found, the constituent labeled with *:1* will become the first concept, and the constituent labeled with *:2* will become the second, of a predicate described by the given relation.

The expressions in italics require a portion of the text to be parsed as a phrase of that type. The parsing rules for phrases appear in the next section.

Table A.1: Top-level parsing patterns used to build ConceptNet in English.

| Pattern | Relation |
|---|---|
| The first thing you do when you *VP:1* is *VP:2* | HasFirstSubevent |
| The last thing you do when you *VP:1* is *VP:2* | HasLastSubevent |
| Something you need to do before you *VP:1* is *VP:2* | HasPrerequisite |
| *NP:1* requires *NP:2* | HasPrerequisite |
| If you want to *VP:1* then you should *VP:2* | HasPrerequisite |
| *NP:1 BE ADVP* made of *NP:2* | MadeOf |
| *NP:1 BE* a kind of *NP:2 TAG* | IsA |
| *NP:1 BE* a sort of *NP:2 TAG* | IsA |
| Something you might find *P NP:2* is *NP:1* | AtLocation |
| Something you find *P NP:2* is *NP:1* | AtLocation |
| Somewhere *NP:1* can be is *P NP:2* | AtLocation |
| You are likely to find *NP:1 P NP:2* | AtLocation |
| *NP:1 BE* used for *NP:2* | UsedFor |
| *NP:1 BE* used to *VP:2* | UsedFor |
| You can use *NP:1* to *VP:2* | UsedFor |
| People use *NP:1* to *VP:2* | UsedFor |
| People use *NP:1* for *NP:2* | UsedFor |
| *NP:1 BE ADVP* for *VP:2* | UsedFor |
| *NP:1 BE ADVP* for *NP:2* | UsedFor |
| *NP:1 BE* capable of *NP:2* | CapableOf |
| An activity *NP:1* can do is *VP:2* | CapableOf |
| An activity *NP:1* can do is *NP:2* | CapableOf |
| You would *VP:1* because you want to *VP:2* | MotivatedByGoal |
| You would *VP:1* because you want *NP:2* | MotivatedByGoal |
| *NP:1 ADVP* wants to *VP:2* | Desires |
| *NP:1 ADVP* wants *NP:2* | Desires |
| *NP:1 ADVP* want to *VP:2* | Desires |
| *NP:1 ADVP* want *NP:2* | Desires |
| *NP:1 BE* defined as *NP:2* | DefinedAs |
| *NP:1 BE* the *NP:2* | DefinedAs |
| *NP:1 BE DT* symbol of *NP:2* | SymbolOf |
| *NP:1* would make you want to *VP:2* | CausesDesire |
| You would *VP:2* because *XP:1* | CausesDesire |
| The effect of *XP:1* is that *S:2* | Causes |
| The effect of *XP:1* is *NP:2* | Causes |
| The consequence of *XP:1* is that *XP:2* | Causes |
| Something that might happen as a consequence of *XP:1* is that *XP:2* | Causes |
| Something that might happen as a consequence of *XP:1* is *XP:2* | Causes |
| *ADVP NP:1* causes you to *VP:2* | Causes |
| *ADVP NP:1* causes *NP:2* | Causes |
| One of the things you do when you *VP:1* is *XP:2* | HasSubevent |
| Something that might happen when you *VP:1* is *XP:2* | HasSubevent |

| Pattern | Relation |
|---|---|
| Something that might happen while *XP:1* is *XP:2* | HasSubevent |
| Something you might do while *XP:1* is *XP:2* | HasSubevent |
| *NP:1 BE* part of *NP:2* | PartOf |
| Something that might happen as a consequence of *XP:1* is *XP:2* | Causes |
| *ADVP NP:1* causes you to *VP:2* | Causes |
| *ADVP NP:1* causes *NP:2* | Causes |
| One of the things you do when you *VP:1* is *XP:2* | HasSubevent |
| Something that might happen when you *VP:1* is *XP:2* | HasSubevent |
| Something that might happen while *XP:1* is *XP:2* | HasSubevent |
| Something you might do while *XP:1* is *XP:2* | HasSubevent |
| *NP:1 BE* part of *NP:2* | PartOf |
| You make *NP:1* by *NP:2* | CreatedBy |
| *NP:1* is created by *NP:2* | CreatedBy |
| There *BE NP:1 P NP:2* | AtLocation |
| *NP:1 BE PASV:2* | ReceivesAction |
| You can *ADVP V:2 NP:1* | ReceivesAction |
| *NP:1 BE P NP:2* | AtLocation |
| *NP:1 BE ADVP AP:2* | HasProperty |
| *NP:2 ADVP* has *NP:1* | PartOf |
| *NP:2 ADVP* have *NP:1* | PartOf |
| *NP:1 BE ADVP NP:2 TAG* | IsA |
| *NP:1* ca n't *VP:2* | CapableOf |
| *NP:1* cannot *VP:2* | CapableOf |
| *NP:1* can *VP:2* | CapableOf |
| *NP:1 VP:2* | CapableOf |

## A.2 Phrase-level English parsing patterns

The following rules are used in a bottom-up chart parser to find all possible matching phrases. The set of possible matches then determines whether the top-level patterns, described above, can match.

Expressions in square brackets refer to tags assigned by the N-best Brill tagger [Brill, 1992], trained on the Brown corpus. Words can be assigned multiple possible tags, so the square-bracket expressions match if they match any one of the tags.

Table A.2: Phrase-level parsing patterns used to build ConceptNet in English.

| | | |
|---|---|---|
| *ADVP* | → | $\epsilon$ \| [RB] \| [RB] [RB] \| [MD] [RB] \| *DO* [RB] |
| *AP* | → | [JJ] \| [VBN] \| [PRP$] \| *AP AP* \| *AP* and *AP* \| *AP* , *AP* \| *NP* [POS] \| [JJR] \| [JJS] \| [CD] |
| *BE* | → | be \| is \| are \| was \| were \| being \| been \| [MD] be \| [MD] [RB] be \| 's \| 're \| 'm |
| *CHANGE* | → | get \| become \| gets \| becomes |
| *DO* | → | do \| does \| did |
| *NP* | → | [DT] *AP N'* \| *AP N'* \| [DT] *N'* \| *N'* \| *Npr* \| [VBG] \| [PRP] \| [VBG] *NP* \| [VBG] *NP* [RB] \| [VBG] *P* \| [VBG] *NP P* \| *NP PP* \| *NP* and *NP* |
| *Npr* | → | [NNP] \| [NNP] *Npr* |
| *N'* | → | [NN] \| [NNS] \| [NN] *N'* |
| *P* | → | [IN] \| [TO] |
| *PASV* | → | [VBN] \| [VBN] *PP* \| [VBN] *PP PP* |
| *PP* | → | *P NP* \| [TO] *VP* |
| *S* | → | *NP VP* |
| *TAG* | → | $\epsilon$ \| [VBN] *PP* \| [WDT] *VP* \| [WDT] *S* |
| *V* | → | [VB] \| [VBP] \| go [VB] \| go and [VB] \| [VBZ] |
| *VP* | → | *ADVP V* \| *ADVP V NP* \| *ADVP V PP* \| *BE NP* \| *BE AP* \| *CHANGE AP* \| *VP* [RB] |
| *XP* | → | *NP* \| *VP* \| *S* |

## A.3 Portuguese patterns

The Portuguese ConceptNet can be built out of a plain-Portuguese representation of the statements collected by *Open Mind Common Sense no Brasil* using only regular-expression matching, because all of its statements were collected through pre-defined templates. The regular expressions that follow are used similarly to the English top-level patterns, in that a predicate is created from the first one that matches.

Each pattern is also associated with a polarity that is assigned to the resulting predicate, and a number indicating which parenthesized group (1 or 2) is the first concept.

Table A.3: Regular-expression patterns used to build ConceptNet in Portuguese.

| Pattern | Relation | Group | Polarity |
|---|---|---|---|
| `^Um(a) (.+) é um tipo de (.+)$` | IsA | 1 | 1 |
| `^Uma coisa que você pode encontrar em um(a) (.+) é um(a) (.+)$` | AtLocation | 2 | 1 |
| `^Você (as vezes\|pode\|frequentemente\|muito frequentemente\|quase sempre\|geralmente) quer um(a) (.+) para (.+)$` | UsedFor | 1 | 1 |
| `^Você (quase nunca\|raramente\|geralmente não) quer um(a) (.+) para (.+)$` | UsedFor | 1 | -1 |
| `^Um(a) (.+) é usado(a) para (.+)$` | UsedFor | 1 | 1 |
| `^Você geralmente encontra um(a) (.+) em um(a) (.+)$` | AtLocation | 1 | 1 |
| `^Um lugar onde você geralmente encontra um(a) (.+) é em um(a) (.+)$` | AtLocation | 1 | 1 |
| `^Pessoas (.+) quando elas (.+)$` | HasSubevent | 2 | 1 |
| `^Um outro jeito de dizer (.+), é (.+)$` | DefinedAs | 1 | 1 |
| `^Quando se tenta (.+), um problema encontrado pode ser (.+)$` | ObstructedBy | 1 | 1 |
| `^Quando pessoas (.+), uma forma de ajudar é (.+)$` | MotivatedByGoal | 2 | 1 |
| `^Para (.+) deve-se (.+)$` | HasPrerequisite | 1 | 1 |
| `^Antes de (.+), deve-se (.+)$` | HasPrerequisite | 1 | 1 |
| `^Para poder (.+) é preciso (.+)$` | HasPrerequisite | 1 | 1 |
| `^Uma?  (.+) gosta de (.+)$` | Desires | 1 | 1 |
| `^Para conseguir (.+) deve-se (.+)$` | MotivatedByGoal | 2 | 1 |
| `^O que leva alguém a (.*) é o(a) (.+)$` | HasPrerequisite | 1 | 1 |
| `^As pessoas (cuidam de um doente em casa) para (.+)$` | MotivatedByGoal | 1 | 1 |
| `^(.+) pode ser um dos sintomas de (.+)$` | Causes | 2 | 1 |

# Appendix B

# User evaluation data

Table B.1: Results from the user evaluation described in Chapter 6. The four central columns refer to the four sources of statements, and the numbers in those columns are the average scores that each user assigned to each type of statement.

| Subject ID | Random | Inference1 | Inference2 | OMCS | # answered |
|---|---|---|---|---|---|
| 23180 | -0.60 | 0.27 | 1.27 | 1.07 | 60 |
| 23181 | -0.27 | 1.07 | 1.00 | 1.67 | 60 |
| 23182 | -0.79 | 0.73 | 1.00 | 0.69 | 57 |
| 23183 | -0.67 | 1.07 | 1.07 | 1.67 | 60 |
| 23184 | -0.87 | 0.93 | 1.40 | 1.20 | 59 |
| 23185 | -0.47 | 0.33 | 1.27 | 1.20 | 60 |
| 23187 | -0.47 | 1.13 | 1.07 | 1.53 | 60 |
| 23191 | 0.20 | 1.33 | 0.47 | 0.40 | 60 |
| 23196 | -0.71 | 0.92 | 0.46 | 1.36 | 53 |
| 23199 | -0.38 | 0.27 | 0.50 | 1.23 | 42 |
| 23200 | -0.43 | 0.47 | 0.54 | 1.46 | 55 |
| 23201 | -1.00 | 0.80 | 1.40 | 0.55 | 41 |
| 23206 | -0.47 | 1.07 | 1.00 | 1.73 | 59 |
| 23208 | -0.43 | 0.77 | 1.00 | 1.89 | 40 |
| 23211 | -0.93 | 0.69 | 0.73 | 1.07 | 58 |
| 23212 | -0.70 | 1.20 | 0.63 | 1.75 | 32 |
| 23213 | -0.42 | 1.40 | 0.83 | 1.36 | 45 |
| 23216 | -0.93 | 0.79 | 0.73 | 1.21 | 58 |
| 23218 | -0.80 | 0.87 | 0.87 | 1.80 | 60 |
| 23221 | -0.07 | 0.80 | 1.47 | 1.60 | 60 |
| 23222 | -0.87 | 0.40 | 0.73 | 1.79 | 59 |
| 23224 | -0.63 | -0.10 | 0.70 | 0.89 | 37 |
| 23225 | -0.67 | 1.07 | 1.47 | 1.33 | 60 |
| 23226 | -0.87 | 0.57 | 0.93 | 1.86 | 57 |
| 23227 | -1.00 | 1.00 | 0.87 | 1.27 | 60 |

| Subject ID | Random | Inference1 | Inference2 | OMCS | # answered |
|---|---|---|---|---|---|
| 23228 | 0.11 | 0.89 | 1.29 | 1.71 | 32 |
| 23229 | -0.87 | 0.54 | 0.86 | 1.62 | 55 |
| 23230 | -0.45 | 1.15 | 1.44 | 1.88 | 41 |
| 23233 | -0.60 | 0.47 | 0.80 | 1.73 | 60 |
| 23234 | -0.87 | 0.47 | 0.87 | 1.93 | 60 |
| 23236 | -0.86 | 0.64 | 0.33 | 1.38 | 45 |
| 23239 | -0.87 | 0.67 | 0.43 | 1.80 | 59 |
| 23241 | -0.86 | 1.33 | 1.08 | 1.47 | 57 |
| 23244 | -0.69 | 0.78 | 1.09 | 1.50 | 45 |
| 23245 | -1.00 | 0.38 | 0.67 | 1.29 | 40 |
| 23246 | -0.87 | 1.20 | 0.21 | 0.93 | 59 |
| 23248 | -0.50 | 0.86 | 0.86 | 1.67 | 55 |
| 23253 | -0.80 | 0.80 | 0.93 | 1.73 | 60 |
| 23258 | -0.21 | 0.67 | 1.14 | 1.53 | 58 |
| 23261 | -0.67 | 1.13 | 0.47 | 0.60 | 60 |
| 23264 | -0.80 | 1.00 | 1.07 | 1.27 | 60 |
| 23265 | -0.60 | 1.20 | 1.00 | 1.27 | 59 |
| 23267 | -0.80 | 0.64 | 0.27 | 0.80 | 59 |
| 23268 | -0.50 | 1.08 | 0.88 | 1.83 | 47 |
| 23271 | -1.00 | 1.40 | 1.00 | 1.73 | 60 |
| 23273 | -0.53 | 0.73 | 0.73 | 1.53 | 60 |
| 23274 | -0.47 | 0.47 | 0.67 | 1.13 | 60 |
| 23277 | -0.93 | 1.07 | 0.58 | 1.36 | 54 |
| 23278 | -0.80 | 0.13 | 0.80 | 1.60 | 60 |
| 23280 | -0.80 | 0.33 | 1.20 | 1.20 | 60 |
| 23282 | -0.53 | 1.00 | 0.29 | 0.93 | 59 |
| 23283 | -1.00 | 0.70 | 1.00 | 2.00 | 31 |
| 23286 | -0.27 | 0.93 | 0.93 | 1.07 | 60 |
| 23288 | -0.47 | 1.43 | 1.40 | 1.67 | 59 |
| 23289 | -0.80 | 0.67 | 1.00 | 1.53 | 60 |
| 23290 | -0.67 | 0.80 | 0.60 | 1.47 | 60 |
| 23291 | -0.53 | 0.33 | 0.86 | 1.20 | 59 |
| 23292 | -0.53 | 1.14 | 1.43 | 2.00 | 57 |
| 23293 | -1.00 | 0.79 | 0.43 | 1.07 | 57 |
| 23294 | -0.73 | 0.67 | 1.29 | 1.40 | 59 |
| 23297 | -0.80 | 0.27 | 0.73 | 1.60 | 60 |
| 23298 | -0.73 | 0.50 | 0.79 | 1.67 | 58 |
| 23299 | -0.87 | 0.80 | 0.67 | 0.40 | 60 |
| 23300 | -1.00 | 0.53 | 1.13 | 1.60 | 59 |
| 23301 | -0.40 | 0.93 | 0.87 | 1.27 | 60 |
| 23302 | -0.73 | 1.00 | 0.47 | 1.20 | 60 |
| 23305 | 0.20 | 1.00 | 1.27 | 1.40 | 60 |
| 23308 | -0.47 | 0.71 | 0.80 | 1.73 | 59 |
| 23309 | -0.60 | 1.27 | 1.47 | 1.53 | 60 |
| 23310 | -0.80 | 1.13 | 0.60 | 1.47 | 60 |
| 23311 | -0.40 | 0.40 | 0.47 | 1.53 | 60 |
| 23314 | -0.87 | 0.87 | 0.67 | 1.60 | 60 |

| Subject ID | Random | Inference1 | Inference2 | OMCS | # answered |
|---|---|---|---|---|---|
| 23315 | -1.00 | 0.40 | 0.47 | 0.93 | 60 |
| 23317 | -0.93 | -0.20 | 0.67 | 1.13 | 60 |
| 23318 | -0.60 | 0.27 | 0.60 | 1.73 | 60 |
| 23319 | -0.87 | 0.20 | 0.93 | 0.87 | 60 |
| 23320 | -0.87 | 1.20 | 0.40 | 1.20 | 60 |
| 23321 | -0.47 | 0.93 | 1.13 | 1.73 | 59 |
| 23322 | -1.00 | 0.40 | 0.43 | 1.00 | 59 |
| 23323 | -0.07 | 1.73 | 0.87 | 1.33 | 60 |
| 23324 | -0.47 | 0.36 | 0.33 | 1.47 | 59 |
| 23326 | -0.87 | 0.67 | 1.20 | 1.53 | 60 |
| 23327 | -0.53 | 0.47 | 0.60 | 1.60 | 60 |
| 23328 | -0.73 | 0.67 | 0.60 | 0.93 | 60 |
| 23329 | -0.20 | 0.86 | 0.67 | 1.60 | 59 |
| 23331 | -0.60 | 0.73 | 0.47 | 1.40 | 60 |
| 23333 | -0.53 | 0.40 | 0.64 | 1.20 | 59 |
| 23335 | -0.47 | 0.80 | 0.73 | 1.53 | 60 |
| 23336 | -0.87 | 1.27 | 0.80 | 1.00 | 60 |
| 23337 | -0.87 | 1.20 | 1.20 | 1.60 | 60 |
| 23338 | -0.44 | 0.67 | 0.50 | 1.29 | 35 |
| 23341 | -0.67 | 0.86 | 0.67 | 1.20 | 59 |
| 23343 | -1.00 | 0.43 | 0.33 | 1.07 | 59 |
| 23345 | 0.07 | 1.00 | 0.53 | 1.40 | 60 |
| 23346 | -0.40 | 0.93 | 1.21 | 1.73 | 59 |
| 23348 | -0.67 | 1.20 | 0.67 | 1.40 | 60 |
| 23351 | -0.27 | 0.93 | 1.40 | 1.60 | 60 |
| 23352 | -0.47 | 0.87 | 1.20 | 1.47 | 60 |
| 23354 | -0.93 | 0.86 | 0.47 | 1.27 | 59 |
| 23356 | -0.40 | 0.93 | 1.33 | 1.33 | 60 |
| 23357 | -0.53 | 1.13 | 1.13 | 1.33 | 60 |
| 23360 | -0.80 | 1.13 | 1.27 | 1.64 | 59 |
| 23362 | -0.80 | 0.60 | 0.60 | 1.87 | 60 |
| 23363 | -0.87 | 0.53 | 0.80 | 1.00 | 60 |
| 23365 | -0.67 | 1.27 | 1.13 | 1.07 | 60 |
| 23366 | -1.00 | 0.33 | 0.53 | 1.53 | 60 |
| 23367 | -0.53 | 0.67 | 0.60 | 1.27 | 60 |
| 23368 | -0.87 | 1.50 | 0.93 | 1.40 | 59 |
| 23369 | -0.73 | 0.60 | 1.29 | 1.40 | 59 |
| 23370 | -1.00 | 0.73 | 0.93 | 1.20 | 60 |
| 23373 | -0.80 | 0.29 | 1.00 | 1.11 | 36 |
| 23374 | -0.87 | 1.07 | 0.93 | 0.80 | 60 |
| 23375 | -0.87 | 1.13 | 1.07 | 1.33 | 60 |
| 23376 | -0.53 | 1.60 | 1.33 | 1.87 | 60 |
| 23377 | -0.73 | 0.60 | 0.73 | 1.13 | 60 |
| 23378 | -0.93 | 0.93 | 0.80 | 1.53 | 60 |
| 23379 | -1.00 | 1.07 | 1.00 | 1.20 | 60 |
| 23380 | -1.00 | 0.73 | 0.87 | 1.60 | 60 |
| 23381 | 0.53 | 1.07 | 1.40 | 2.00 | 60 |

| Subject ID | Random | Inference1 | Inference2 | OMCS | # answered |
|---|---|---|---|---|---|
| 23383 | -1.00 | 0.14 | 0.53 | 1.27 | 59 |
| 23385 | -0.79 | 1.07 | 1.33 | 1.73 | 59 |
| 23386 | -0.73 | 0.40 | 0.60 | 0.93 | 60 |
| 23387 | -0.87 | 1.07 | 0.47 | 1.13 | 60 |
| 23390 | -0.67 | 0.80 | 1.27 | 1.53 | 60 |
| 23391 | -0.87 | 1.07 | 0.80 | 0.87 | 60 |
| 23392 | -0.63 | 0.50 | 1.09 | 1.50 | 43 |
| 23393 | -0.53 | 0.20 | 0.71 | 1.27 | 59 |
| 23394 | -0.93 | 0.80 | 1.40 | 1.67 | 60 |
| 23395 | -0.53 | 0.73 | 1.40 | 0.80 | 60 |
| 23398 | -1.00 | 0.79 | 0.80 | 1.47 | 59 |
| 23399 | -0.93 | 0.93 | 0.80 | 1.33 | 60 |
| 23400 | -1.00 | 0.53 | 1.13 | 1.47 | 60 |
| 23401 | -1.00 | 0.93 | 0.73 | 1.13 | 59 |
| 23406 | -0.80 | 0.40 | -0.27 | 0.60 | 60 |
| 23407 | -0.80 | 0.47 | 0.67 | 0.80 | 60 |
| 23409 | -1.00 | 0.93 | 0.93 | 1.27 | 60 |
| 23411 | -1.00 | 0.60 | 0.73 | 1.40 | 60 |
| 23413 | -0.93 | 1.00 | 1.13 | 1.53 | 60 |
| 23414 | -0.67 | 1.20 | 1.47 | 1.00 | 60 |
| 23415 | -0.60 | 0.60 | 0.73 | 1.40 | 60 |
| 23417 | -0.93 | 0.87 | 0.00 | 1.33 | 60 |
| 23420 | -0.80 | 0.87 | 1.07 | 1.40 | 60 |
| 23421 | -0.67 | 1.20 | 1.00 | 1.93 | 60 |
| 23422 | -0.87 | 1.40 | 1.07 | 1.40 | 60 |
| 23423 | -0.73 | 0.67 | 0.64 | 1.73 | 59 |
| 23426 | -0.80 | 0.67 | 1.20 | 1.27 | 60 |
| 23427 | -1.00 | 1.20 | 1.07 | 1.67 | 60 |
| 23428 | -1.00 | 1.60 | 0.60 | 1.53 | 60 |
| 23429 | -0.73 | 1.07 | 1.13 | 1.13 | 60 |
| 23430 | 1.00 | 1.60 | 1.21 | 1.47 | 59 |
| 23431 | -0.87 | 1.00 | 0.47 | 1.13 | 60 |
| 23432 | -1.00 | -0.20 | 0.93 | 1.13 | 60 |
| 23435 | -0.93 | 0.29 | 0.67 | 1.40 | 59 |
| 23436 | -0.87 | 0.53 | 0.43 | 1.20 | 59 |
| 23437 | -0.60 | 1.07 | 0.79 | 1.13 | 59 |
| 23438 | -0.73 | 0.73 | 0.80 | 1.40 | 60 |
| 23439 | -0.87 | 0.43 | 0.73 | 0.80 | 59 |
| 23440 | -0.87 | 0.27 | 0.87 | 1.67 | 60 |
| 23444 | -0.93 | 0.93 | 1.53 | 1.27 | 59 |
| 23445 | -0.47 | 0.93 | 1.27 | 1.33 | 59 |
| 23448 | -0.27 | 1.07 | 1.20 | 1.13 | 60 |
| 23451 | -1.00 | 1.20 | 0.86 | 1.73 | 59 |
| 23452 | -0.73 | 0.80 | 1.33 | 1.60 | 60 |
| 23453 | -0.67 | 1.13 | 0.00 | 0.87 | 60 |
| 23456 | -1.00 | 1.00 | 0.60 | 1.20 | 60 |
| 23458 | -0.53 | 0.53 | 1.13 | 0.73 | 60 |

| Subject ID | Random | Inference1 | Inference2 | OMCS | # answered |
|---|---|---|---|---|---|
| 23460 | -1.00 | 0.73 | 0.40 | 1.27 | 60 |
| 23461 | -0.33 | 0.73 | 1.73 | 1.67 | 60 |
| 23462 | -0.67 | 1.43 | 1.60 | 1.40 | 59 |
| 23463 | -0.73 | 1.07 | 1.13 | 1.33 | 60 |
| 23464 | -0.87 | 0.67 | 1.20 | 1.47 | 60 |
| 23465 | -0.40 | 0.27 | 1.00 | 1.47 | 60 |
| 23466 | -0.87 | 1.20 | 0.93 | 1.47 | 59 |
| 23468 | -0.67 | 0.57 | 0.40 | 1.47 | 59 |
| 23469 | -1.00 | 1.33 | 0.53 | 1.73 | 60 |
| 23470 | -0.73 | 0.67 | 1.00 | 1.40 | 60 |
| 23472 | -0.60 | 0.47 | 0.93 | 1.47 | 60 |
| 23474 | -0.80 | 1.00 | 0.87 | 0.87 | 60 |
| 23476 | -0.80 | 0.53 | 0.27 | 1.67 | 60 |
| 23480 | -0.80 | 0.80 | 0.93 | 1.53 | 60 |
| 23482 | -0.73 | 0.93 | 0.67 | 0.53 | 60 |
| 23484 | -0.73 | 1.13 | 0.53 | 1.87 | 60 |
| 23490 | -0.53 | 0.93 | 0.53 | 1.10 | 57 |
| 23492 | -0.79 | 0.80 | 1.13 | 1.10 | 54 |
| 23493 | -0.60 | 0.87 | 0.53 | 1.00 | 60 |
| 23494 | -0.67 | 0.47 | 0.80 | 1.13 | 60 |
| 23495 | -0.87 | 1.33 | 0.67 | 0.20 | 60 |
| 23496 | -0.60 | 0.87 | 1.47 | 1.00 | 60 |
| 23499 | -0.53 | 1.00 | 1.27 | 1.07 | 60 |
| 23500 | -0.60 | 1.00 | 0.80 | 0.67 | 60 |
| 23501 | -0.80 | 0.67 | 0.73 | 0.53 | 60 |
| 23502 | -0.33 | 1.67 | 1.47 | 0.93 | 60 |
| 23503 | -0.67 | 1.07 | 1.33 | 1.00 | 60 |
| 23504 | -0.80 | 0.80 | 1.07 | 1.00 | 60 |
| 23505 | -0.87 | 1.07 | 0.93 | 0.73 | 60 |

# Appendix C

# Visualizations of AnalogySpace

The following figures represent the contents of AnalogySpace by plotting two selected components of all concepts and properties, corresponding to each concept or property's weight within two principal components ("eigenconcepts"). This representation is discussed in Chapter 4.

Concepts are indicated with the symbol + at their coordinates, and properties are indicated with ×. Additionally, in color versions of this document (such as the electronic copy in MIT's DSpace), concepts are colored red while properties are blue.

A subset of the points are labeled, chosen for legibility so that the labels do not often overlap. Some labels appear next to an arrow at the edge of a graph, as well; these indicate an important concept or property that would be plotted far beyond the bounds of the graph. Concepts are labeled with a common natural language representation of that concept, left properties are labeled with a concept, a slash, and a relation (such as "person/CapableOf"), and right properties are labeled like left properties but in the opposite order.

Concepts or properties that appear at the same angle from the origin are similar according to the two eigenconcepts being plotted.

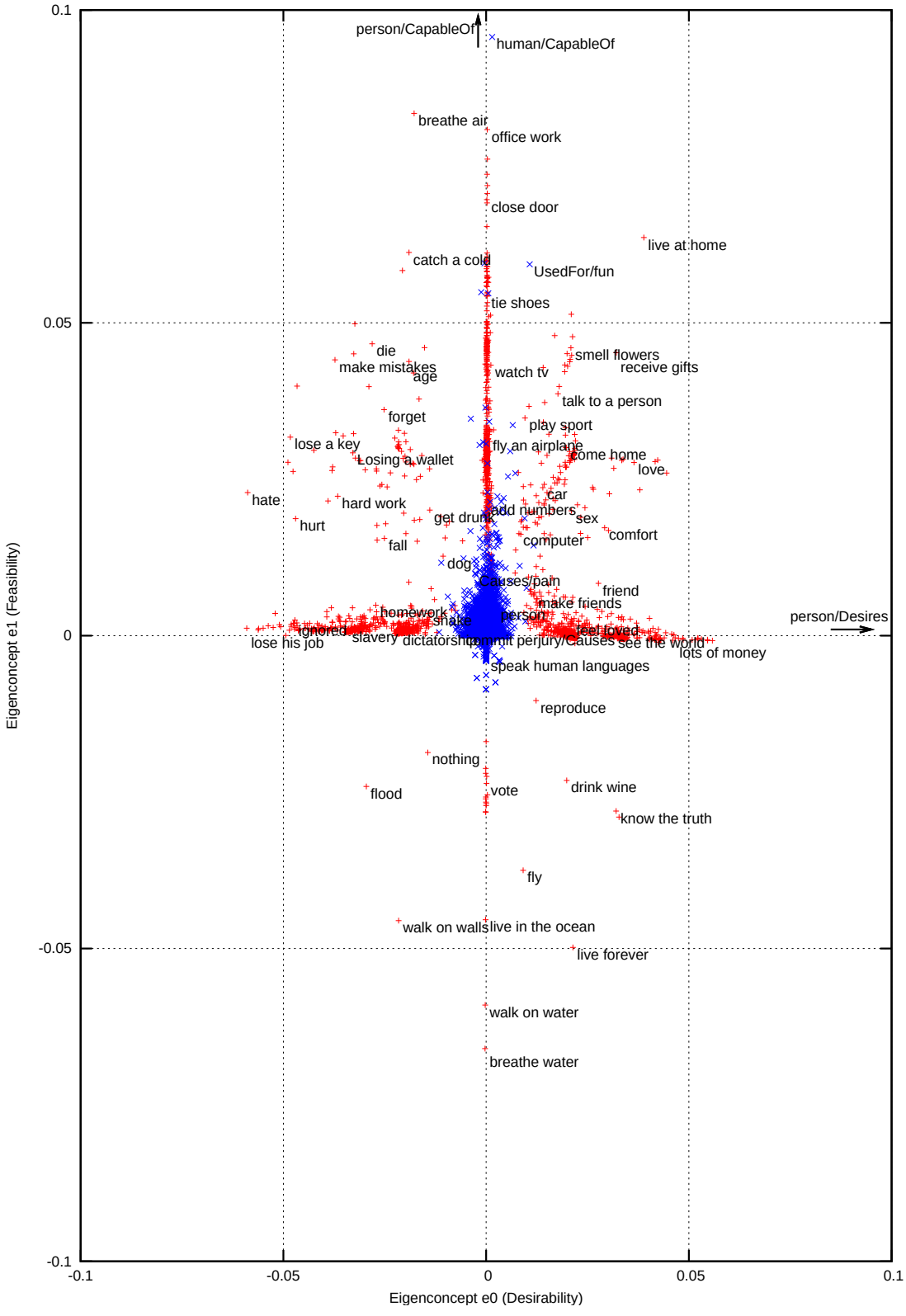


Figure C-1: $\mathbf{e}_0$ and $\mathbf{e}_1$ components of concepts and properties.

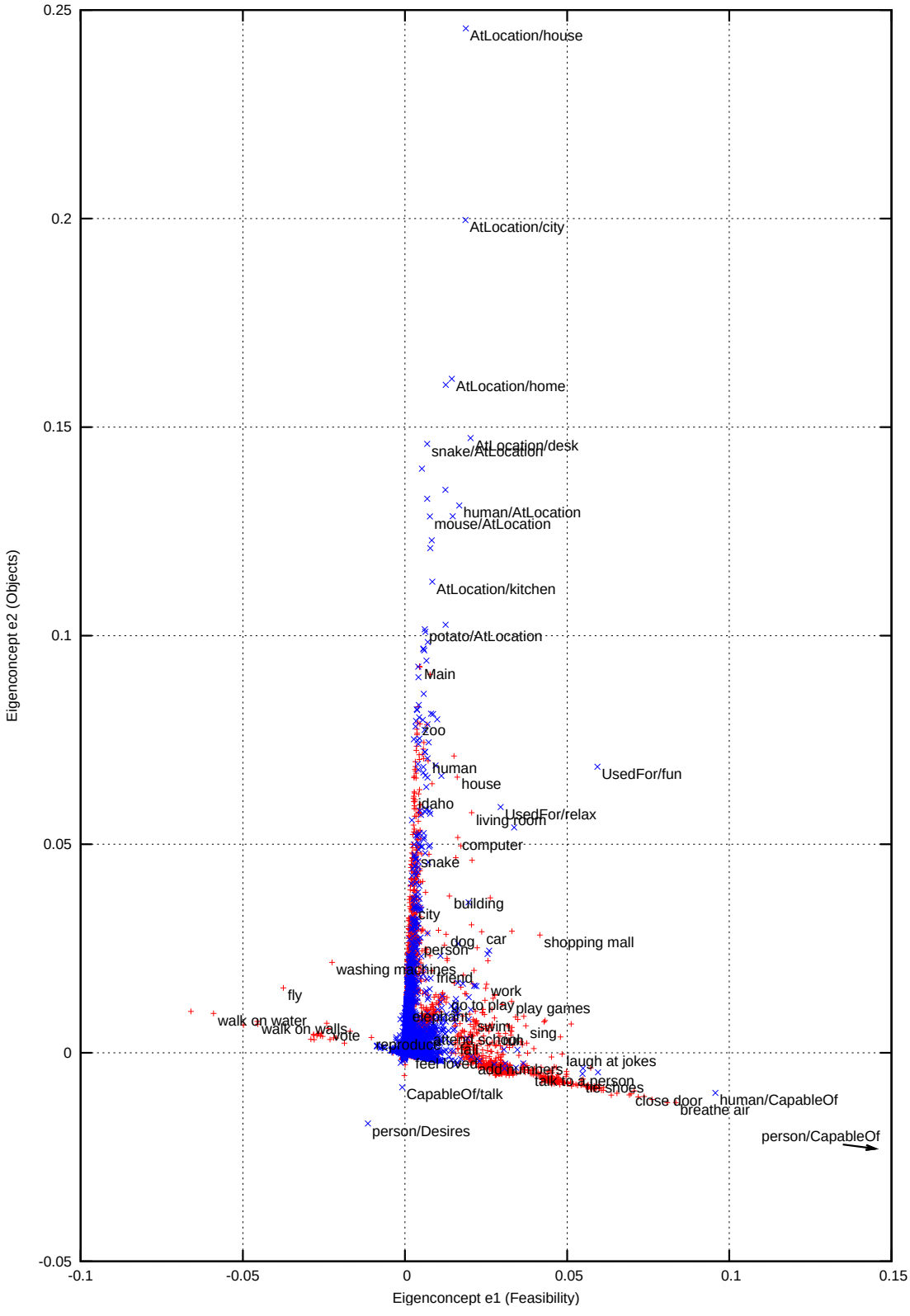


Figure C-2: $\mathbf{e}_1$ and $\mathbf{e}_2$ components of concepts and properties.

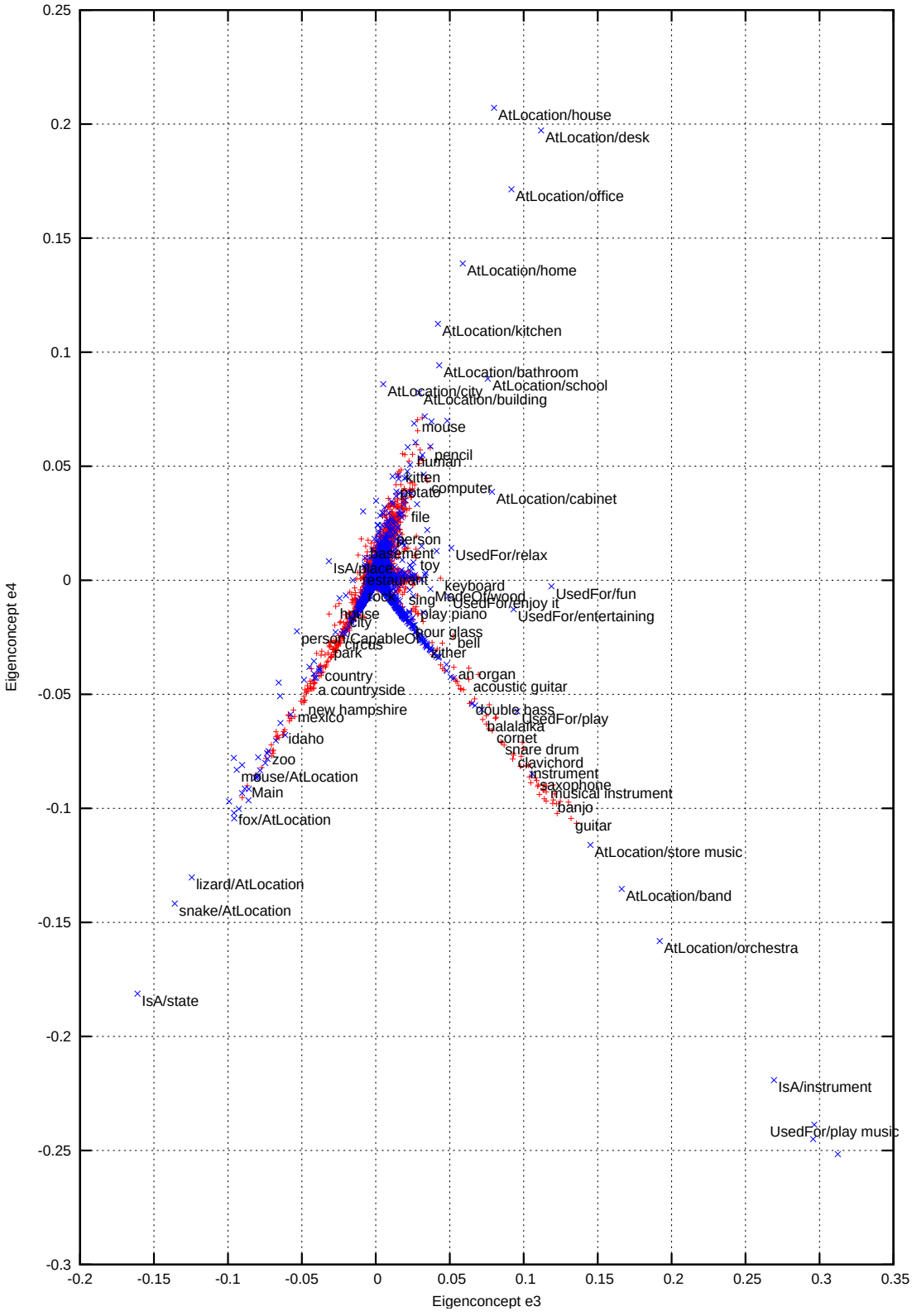


Figure C-3: $\mathbf{e}_3$ and $\mathbf{e}_4$ components of concepts and properties.

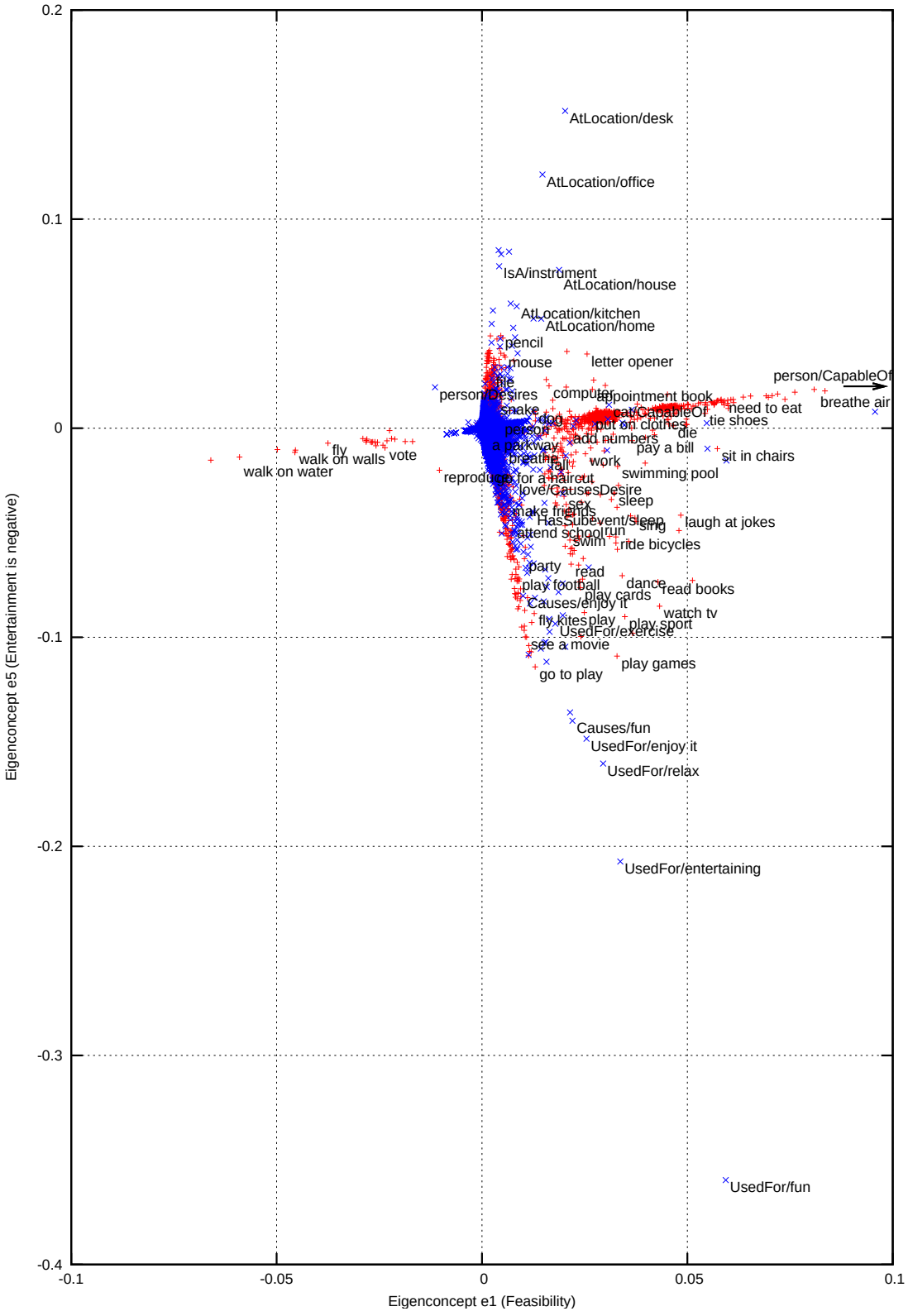


Figure C-4: $\mathbf{e}_1$ and $\mathbf{e}_5$ components of concepts and properties.

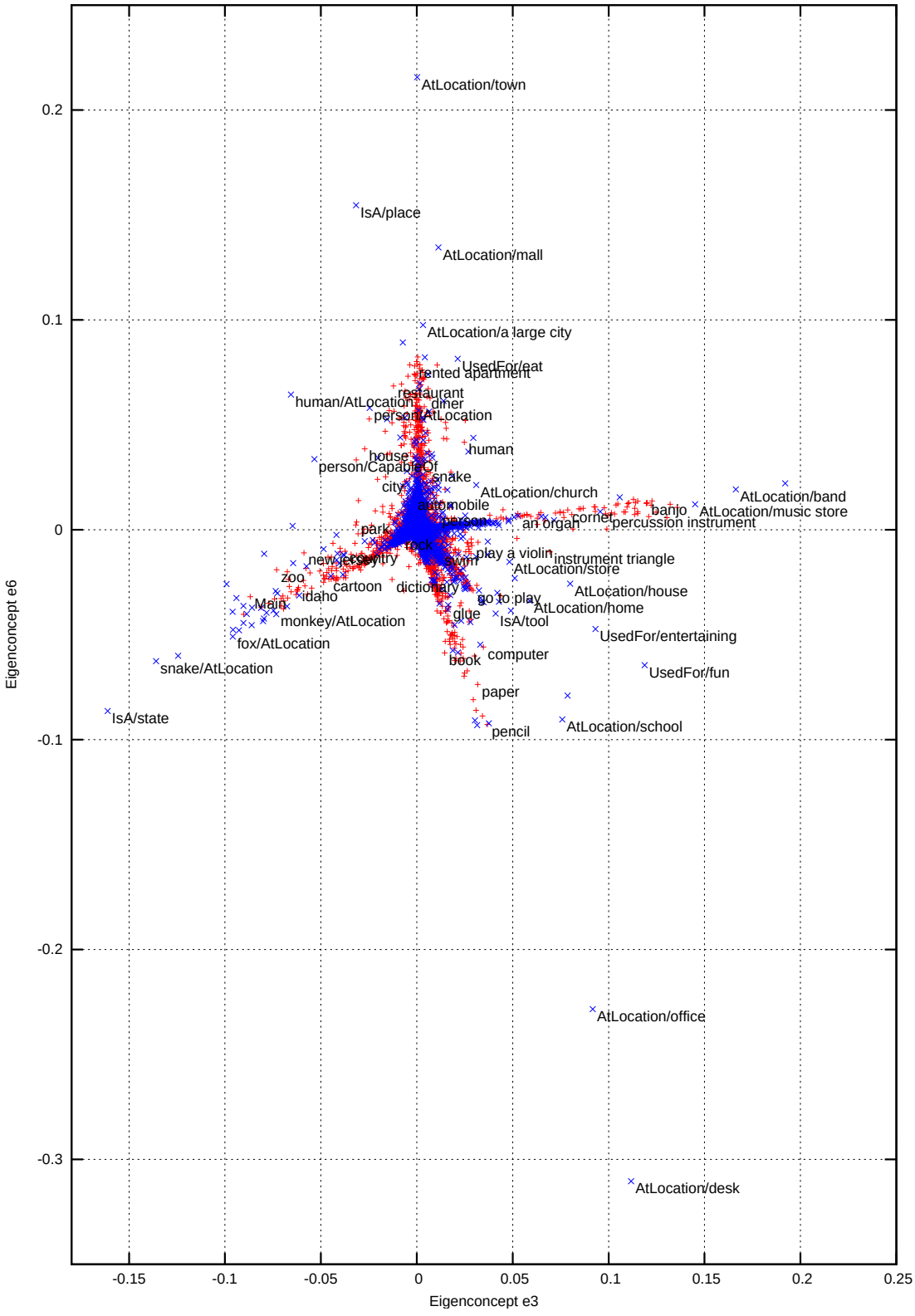


Figure C-5: $\mathbf{e}_3$ and $\mathbf{e}_6$ components of concepts and properties.

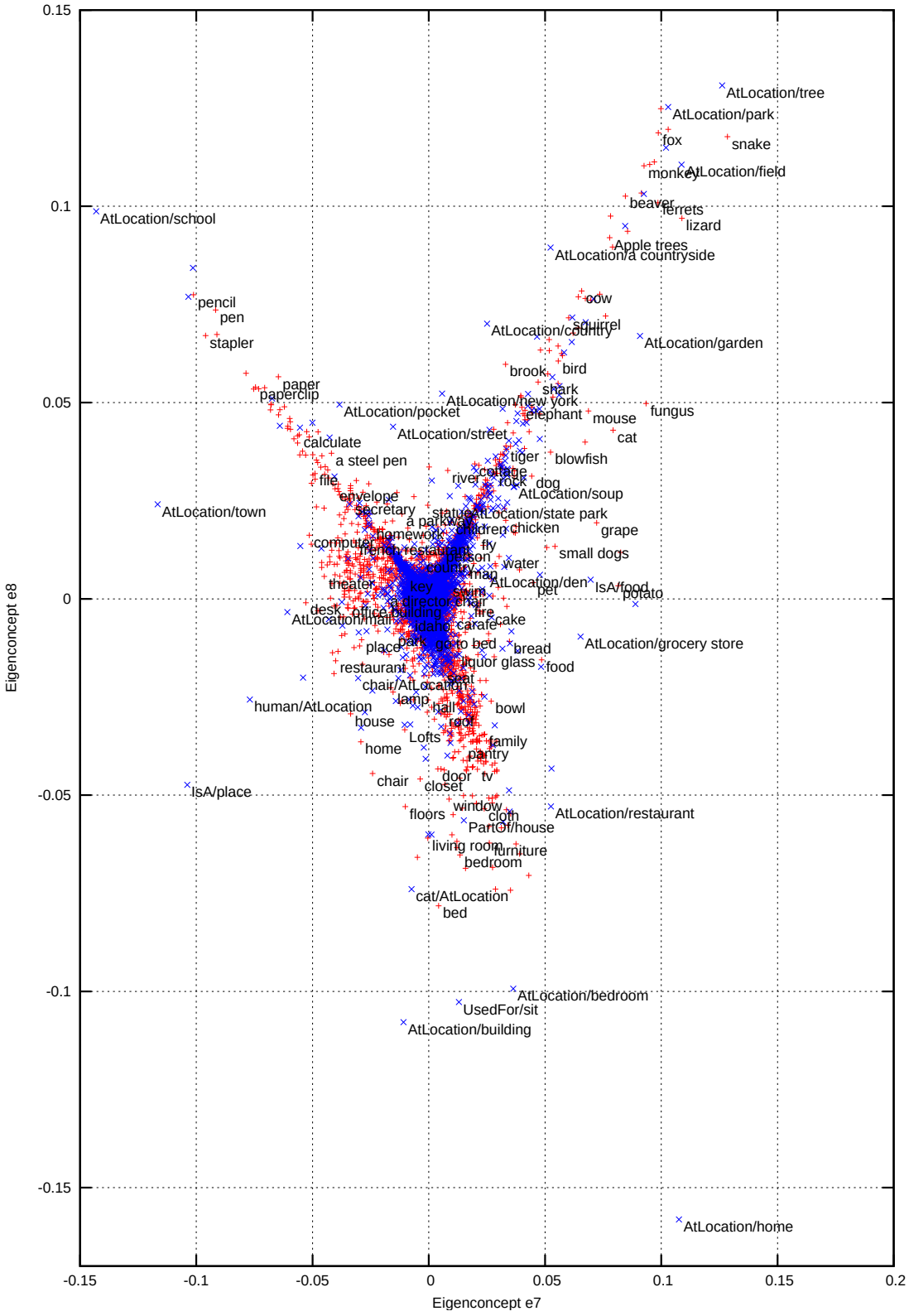


Figure C-6: $\mathbf{e}_7$ and $\mathbf{e}_8$ components of concepts and properties.

# Appendix D

# Downloadable resources

This document has depended on a large amount of code and data, which is available to be downloaded from: `http://web.media.mit.edu/~rspeer/thesis`.

From there, you will find links to the following items:

- A snapshot of the ConceptNet database
- The Python code (CSAMOA) that builds ConceptNet
- The Ruby code that runs the Open Mind Commons site and performs SVD operations
- An electronic, color copy of this thesis (also available from `dspace.mit.edu`)

If the link above should ever go dead, contact me by e-mail at `rspeer@alum.mit.edu` for the files.

# Bibliography

[Alonso, 2007] Jason Alonso. CSAMOA: A common sense application model of architecture. In *Proceedings of the Workshop on Common Sense and Intelligent User Interfaces*, Honolulu, Hawaii, 2007.

[Anacleto *et al.*, 2006] Junia Anacleto, Henry Lieberman, Marie Tsutsumi, Vnia Neris, Aparecido Carvalho, Jose Espinosa, and Silvia Zem-Mascarenhas. Can common sense uncover cultural differences in computer applications? In *World Computer Congress*, 2006. Available from: `http://www.media.mit.edu/~lieber/Publications/Cultural-Differences-IFIP06.pdf`.

[Banerjee and Pedersen, 2003] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness, 2003. Available from: `http://citeseer.ist.psu.edu/banerjee03extended.html`.

[Brill, 1992] Eric Brill. A simple rule-based part of speech tagger. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 112–116, Morristown, NJ, USA, 1992. Association for Computational Linguistics. Available from: `http://dx.doi.org/10.3115/1075527.1075553`.

[Chklovski, 2003] Timothy Chklovski. Learner: a system for acquiring commonsense knowledge by analogy. In *K-CAP '03: Proceedings of the 2nd International Conference on Knowledge Capture*, pages 4–12, New York, NY, USA, 2003. ACM Press. Available from: `http://portal.acm.org/citation.cfm?id=945645.945650`.

[Chung, 2006] Hyemin Chung. *GlobalMind — bridging the gap between different cultures and languages with common-sense computing*. PhD thesis, MIT Media Lab, 2006. Available from: `http://web.media.mit.edu/~lieber/Teaching/Common-Sense-Course/Hyemin-Thesis.pdf`.

[de Carvalho *et al.*, 2007] Aparecido F. de Carvalho, Junia C. Anacleto, Henry Lieberman, Muriel Godoi, and Silvia Zem-Mascarenhas. Using common sense for planning learning activities. In Catherine Havasi and Henry Lieberman, editors, *Workshop on Common Sense and Intelligent User Interfaces*, Honolulu, HI, January 2007. Intelligent User Interfaces Conference. Available from: `http://eurydice.cs.brandeis.edu/csiui/Papers/cs_education_iui.pdf`.

[Deerwester *et al.*, 1990] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic

analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990. Available from: `http://citeseer.ist.psu.edu/deerwester90indexing.html`.

[Fellbaum, 1998] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[Gupta and Kochenderfer, 2004] Rakesh Gupta and Mykel J. Kochenderfer. Using statistical techniques and WordNet to reason with noisy data. In *Workshop on Adaptive Text Extraction and Mining, Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, San Jose, California, July 2004. Available from: `http://web.media.mit.edu/~rgupta/p2.pdf`.

[Harris, 1975] Richard J. Harris. *A Primer of Multivariate Statistics*. Academic Press, London, 1975.

[Havasi *et al.*, 2007] Catherine Havasi, Robert Speer, and Jason Alonso. ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing* (to appear), Borovets, Bulgaria, September 2007. Available from: `http://web.mit.edu/~rspeer/www/research/cnet3.pdf`.

[Lenat, 1995] Doug Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 11:33–38, 1995.

[Lieberman *et al.*, 2005] Henry Lieberman, Alexander Faaborg, Waseem Daher, and José Espinosa. How to wreck a nice beach you sing calm incense. *Proceedings of the 10th International Conference on Intelligent User Interfaces*, 2005.

[Liu and Singh, 2004] Hugo Liu and Push Singh. ConceptNet: a practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226, October 2004. Available from: `http://portal.acm.org/citation.cfm?id=1031314.1031373`.

[Liu *et al.*, 2003a] Hugo Liu, Henry Lieberman, and Ted Selker. A model of textual affect sensing using real-world knowledge. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 125–132. ACM Press, 2003. Available from: `http://portal.acm.org/citation.cfm?id=604067`.

[Liu *et al.*, 2003b] Hugo Liu, Ted Selker, and Henry Lieberman. Visualizing the affective structure of a text document. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 740–741. ACM Press, 2003. Available from: `http://portal.acm.org/citation.cfm?id=765961`.

[Liu, 2004] Hugo Liu. MontyLingua: An end-to-end natural language processor with common sense. Technical report, MIT, 2004. Available from: `http://web.media.mit.edu/~hugo/montylingua`.

[Mueller, 1998] Erik T. Mueller. *Natural language processing with ThoughtTreasure*. New York, Signiform, 1998. Available from: `http://citeseer.ist.psu.edu/mueller98natural.html`.

[Mueller, 2000] Erik T. Mueller. A calendar with common sense. In *Intelligent User Interfaces*, pages 198–201, 2000. Available from: `http://citeseer.ist.psu.edu/316437.html`.

[Musa *et al.*, 2003] Rami Musa, Madleina Scheidegger, Andrea Kulas, and Yoan Anguilet. GloBuddy, a dynamic broad context phrase book. In *Modeling and Using Context*, pages 1026–1026. Springer, Berlin / Heidelberg, 2003. Available from: `http://www.springerlink.com/content/97eajmlg42xfpchj`.

[Patwardhan and Pedersen, 2006] Siddharth Patwardhan and Ted Pedersen. Using WordNet-based context vectors to estimate the semantic relatedness of concepts. In *EACL 2006 Workshop Making Sense of Sense—Bringing Computational Linguistics and Psycholinguistics Together*, pages 1–8, Trento, Italy, 2006. Available from: `http://acl.ldc.upenn.edu/W/W06/W06-2501.pdf`.

[Porter, 2001] Martin F. Porter. Snowball: a language for stemming algorithms. Snowball web site, 2001. `http://snowball.tartarus.org/texts/introduction.html`, accessed Jan. 31, 2007.

[Pustejovsky *et al.*, 2006] James Pustejovsky, Catherine Havasi, Roser Saurí, Patrick Hanks, and Anna Rumshisky. Towards a generative lexical resource: The Brandeis Semantic Ontology. *Proceedings of the Fifth Language Resource and Evaluation Conference*, 2006. Available from: `http://www.cs.brandeis.edu/~arum/publications/lrec-bso.pdf`.

[Pustejovsky, 1998] James Pustejovsky. *The Generative Lexicon*. MIT Press, Cambridge, MA, 1998.

[Rohde, 2001] Doug Rohde. Doug Rohde's SVD C library. SVDLIBC web site, 2001. `http://tedlab.mit.edu/~dr/SVDLIBC/`, accessed Aug. 18, 2007.

[Schierwagen, 2001] Andreas Schierwagen. Vision as computation, or: Does a computer vision system really assign meaning to images? In M. Matthies, H. Malchow, and J. Kriz, editors, *Integrative Systems Approaches to Natural and Social Dynamics*. Springer-Verlag, Berlin, 2001. Available from: `http://www.informatik.uni-leipzig.de/~schierwa/vision.pdf`.

[Singh *et al.*, 2002] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan L. Zhu. Open Mind Common Sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1223–1237, London, UK, 2002. Springer-Verlag. Available from: `http://www.media.mit.edu/~push/ODBASE2002.pdf`.

[Sleator and Temperley, 1993] D. D. Sleator and D. Temperley. Parsing english with a link grammar. In *Third International Workshop on Parsing Technologies*, 1993. Available from: `http://citeseer.ist.psu.edu/44375.html`.

[Smith, 2007] Dustin Smith. *EventMinder: A Personal Calendar Assistant That Recognizes Users' Goals*. PhD thesis, MIT Media Lab, 2007.

[Speer, 2007] Robert Speer. Open Mind Commons: An inquisitive approach to learning common sense. *Proceedings of the Workshop on Common Sense and Interactive Applications*, 2007. Available from: `http://web.mit.edu/~rspeer/www/research/commons.pdf`.

[Stocky *et al.*, 2004] Tom Stocky, Alexander Faaborg, and Henry Lieberman. A commonsense approach to predictive text entry. In *Conference on Human Factors in Computing Systems*, Vienna, Austria, April 2004. Available from: `http://agents.media.mit.edu/projects/textentry/CHI04_textEntry.pdf`.

[Turney, 2005] Peter D. Turney. Measuring semantic similarity by latent relational analysis, Aug 2005. Available from: `http://arxiv.org/abs/cs/0508053`.

[Wall *et al.*, 2003] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. *A Practical Approach to Microarray Data Analysis*, chapter 5, pages 91–109. Kluwel, Norwell, MA, Mar 2003.

[Wirén, 1987] Mats Wirén. A comparison of rule-invocation strategies in context-free chart parsing. In *Proceedings of the third conference on European chapter of the Association for Computational Linguistics*, pages 226–233, Morristown, NJ, USA, 1987. Association for Computational Linguistics.