Analogy in the Large

Kenneth B. Haase
MIT Media Laboratory
20 Ames Street, E15-485
Cambridge, MA 02142
haase@media.mit.edu
(617)253-7418 [Voice]
(617)258-6264 [Fax]

Abstract: This article discusses the use of analogy to index and organize large databases of information. We describe the design and implementation of an analogical database supporting tens to hundreds of thousands of cases. The contents of the database are parsed news articles represented as networks of grammatical relations with references into Wordnet for word meaning information. Efficient algorithms for indexing and matching in this database are described and briefly discussed and examples of their performance are discussed.

Keywords/Areas:
Analogy, Natural Language, AI Architectures

This paper has not already been accepted by and is not currently under review for a journal or another conference nor will it be submitted for such during IJCAI's review process.

Supporting Analogy in the Large

Kenneth B. Haase
MIT Media Laboratory

1 Introduction

This article discusses the design of databases which support dynamic analogies among thousands of descriptions. The databases consist of natural language text parsed into networks describing their grammatical relations and associated with references in a common database. Queries to the database retrieve networks with similar grammatical structure and match elements in order to isolate thematic elements. We call the database analogical because the mappings between elements are not determined by a priori canonical structures like case frames, scripts, or memory packets, but by drawing analogies between them.

The technical contributions of this work include algorithms for efficiently determining analogies. an analysis of the problems of indexing for analogy among thousands of descriptions, and an implemented indexing system based on the analysis. In addition, we discuss the use of Wordnet as a semantic background for analogizing and indexing.

2 The Databases

In 1993 and the first half of 1994, we constructed our first large database supporting analogical indexing and matching in the large. This database (henceforth the '94 database) consisted of roughly 900,000 words of parsed text taken from an online news service. Documents in this database were represented by nodes corresponding to words in the text and arcs describing (possible) grammatical relations between them. Words in different documents with the same stem were related to one another through reference to common elements of a shared knowledge base; in addition, words determined to be in particular classes (for instance, people or place names) were similarly connected with one another.

Experiments with the '94 database were promising. It proved possible to efficiently retrieve texts with similar grammatical structure and to easily identify corresponding elements of texts based on that structure. For relatively stylized texts, this made it possible to use the text itself as a 'canonical form' from which semantic roles could be extracted. For example, relatively stylized reports of finanical activity could be automatically converted into graphs by extracting the corresponding numbers from different reports. The main deficits of the database were its inability to deal with radical changes of grammatical structure (e.g. from active to passive voice) and its lack of knowledge of word meanings.

In the fall of 1994, we began working with a 2.5 million word corpus provided by Gannett Co. to build a new database (the '95 database) to address these deficits. The Gannett corpus consists of one paragraph news summaries from 1987 to the present: we are hoping to use this historical range to press the generality of the approach. Initial attempts to process the new database were stymied by the sluggishness of the underlying database infrastructure as the size of the database passed the 5 million frame mark. In November, we switched to a new infrastructure and began to address the real deficits of the first database.

In the '95 database, new frames are constructed only for the atomic phrases, not for every word. In addition, the parser has been enhanced to be more reliable and to do simple regularization on passive sentences. Most significantly, different words are now connected with one another through the Wordnet database; this allows matching and indexing to use some notion of word meaning in its determinations.

The remainder of this article describes the structures and algorithms used in matching and indexing and their performance in both the '94 and '95 databases.

3 Representational Structure

Descriptions in our database consist of sets of individuals connected by two kinds of relations:

- *micro-relations* connect individuals to either other individuals in the same description or literal values and reference points in a global namespace;
- *associations* connect individuals to either individuals in other descriptions or to reference points in the global namespace

Micro-relations provide a *pre-semantic structural representation* which is neither canonical (there is no promise that semantically equivalent descriptions have the same micro-relational structure) nor entirely correct (some micro-relations may be accidental or artifactual). Instead of being an intepretation, micro-relations are designed to constrain interpretations.

Associations provide an *ambiguous meaning representation* such that if two individuals have associations in common, they are taken as having some semantic similarity. Like micro-relations, associations may be ambiguous and partially incorrect. The association relation is transitive (if x is associated to y and y to z, then z is associated to z), but usually not symmetric (if x is associated to y, y is probably not associated to x). The *immediate associations* of an individual constitute a set out of which all of its associations can be generated.

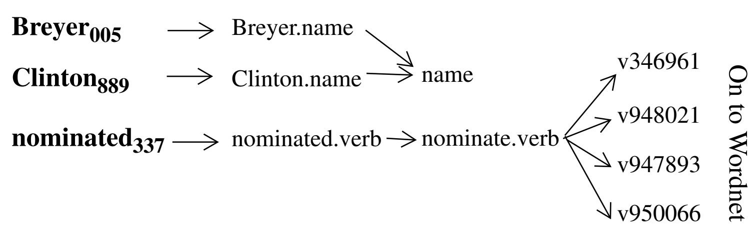
Micro-relations and associations are central to the matching and indexing processes: the matcher uses associational structure to provide base-level matching and micro-relational structure to determine higher level matches. The indexer combines associational and micro-relational structure to construct signatures for descriptions which can be indexed for purposes of finding good analogical matches.

3.1 Representing Text

For example, in our text database, a sentence like 'President Clinton finally nominated Stephen Breyer' is translated into a description consisting of three individuals with the following micro-relational structure:

Individual:	**Clinton$_{889}$**	Individual:	**nominated$_{337}$**	Individual:	**Breyer$_{005}$**
association:	Clinton.name	*association*	nominated.verb	*association:*	Breyer.name
phrase:	President.adj	phrase:	finally.adv	phrase:	Stephen.adj
	Clinton.name		nominated.verb		Breyer.name
subject-of:	nominated$_{337}$	subject:	Clinton$_{889}$	object-of:	nominated$_{337}$
		object:	Breyer$_{005}$		

some of the micro-relations (`subject`, `object-of`, etc) connect individuals in the description to each other; some (e.g. `phrase`) connect individuals to references in the global database (e.g. `nominated.verb`). Each of the individual descriptions also has an immediate association with an individual in the global database, and these individuals may have associations of their own:



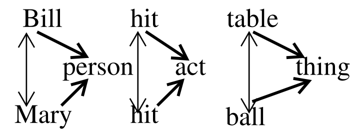
The use of Wordnet to provide the associational structure of the global namespace is the principal difference between the '94 and '95 databases.

4 Base Level Matching

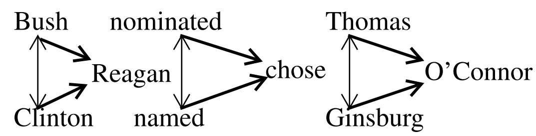
The chief innovation of our matcher is its use of a network of associations to determine base-level similarity. Analogical matchers like SME[Falkenhainer89] have a basic and fixed level of symbolic description at which the matching process 'grounds out'. In our matcher, the base level consists of a network of associations which grows as new descriptions are indexed and matched. One important feature of this sort of base level is that criteria of similarity can be changed without changing the implementation of the matcher. A second feature is that the introduction of new elements to the base level occurs naturally. This second feature distinguishes it from the ACME matcher, which uses an associational base level, but implements it with a relatively fixed network.

The base level similarity metric for our matching mechanism applies to the individuals which constitute descriptions and is based on the identification of unique common associations between them. Two individuals are cognates (i.e. similar) if they share some unique common association z which is shared by no other pairs of individuals from the two descriptions. The cognate relation has two interesting special cases:

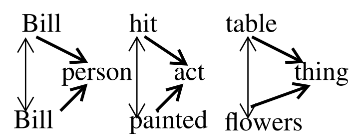
Type-unique cognates occur when two heterogenous descriptions are compared and the associations which make the elements distinct become the foundation for cognate relations between the descriptions.



The second interesting case of the cognate relation is triangle cognates, where individuals are cognates by virtue of a common association in a third description. If two descriptions already have a set of associations in a third description, these can provide the basis for cognate relations between them.



Of course, when presented with a pair of descriptions whose common associations are very general, this will produce fanciful mappings. However, the goal of this base level of matching is not to generate *only* sensible mappings but to generate the most sensible mapping given the possibilities.



The second interesting case of the cognate relation is triangle cognates, where individuals are cognates by virtue of a common association in a third description. If two descriptions already have a set of associations in a third description, these can provide the basis for cognate relations between them.

4.1 Computing Cognates

Given two descriptions ***C*** and ***D*** and some $x \in C$, the cognate of ***x*** in ***D*** can be determined in $O(mn)$ time, where n is the size of ***C*** and ***D*** and m is bounded by the maximum number of associations over the individuals in ***C*** and ***D***. Interestingly, it is also possible to determine the cognates for all of the elements in ***C*** in the same $O(mn)$ time.

The following algorithm determines a single cognate relation:

```
To find the unique associations (cognates) in D of k (in C)
    Create a set of individuals called Candidates
            and a set of pairs of individualscalled Answers
    Add all the associations of k to Candidates
    For each x in C, if x != k
        remove all associations(x) from Candidates
    For each y in D
        for each a in associations(y),
          if a is in Candidates, remove associations(a) from Candidates
          if there is some <z,a> in Answers,
            then remove <z.a> from Answers and remove a from Candidates
        for each a in associations(y) (again),
          if a is still in Candidates then add <y,a> to Answers
    Finally, for each pair <y,a> in Answers,
        return y
```

The core of the algorithm is a competition between the individuals in each description for common associations. The individual ***x*** originally 'claims' all of its associations; its siblings in ***C*** then remove all of their associations from this set. The individuals in ***D*** are then examined for associations in common with and unique to ***x***; whenever one is found, it suggests a mapping and also rules out all of its common associations..

By using bits on each individual to represent the set Candidates and another bit to represent removal of pairs from the set Answers, this algorithm achieves its O(mn) time bounds. In a subset of the '95 database which is currently parsed, m ranged from 2 to 42 with an average value of 10.2. Note that if m remains relatively small, cognate determination becomes effectively linear in the size of the descriptions being matched.

The 'all cognates' algorithm uses a pair of fields on each individual to combine the searches for individual cognates:

```
To find the unique associations (cognates) between C and D
  Create a set of individuals called Conflicts
  Create a set of pairs of individuals called Answers
  Create a mapping from individuals into C called Cmap
  Create a mapping from individuals into D called Dmap
  For each x in C for all the associations a of x
    if Cmap(a) is defined, add a to the set Conflicts
    otherwise set Cmap(a)=x
  For each y in D
    for all the associations a of y
      if Dmap(a) is defined, add a to the set Conflicts
      otherwise set Dmap(a)=y
  For each x in C
    for each association a of x
      if a is not in Conflicts and Dmap(a) is defined
      add the pair <x,Dmap(a)> to Answers
  return Answers
```

Like the single cognate case, the algorithm iterates over each element of ***C*** and ***D*** and for each interation, iterates over the element's associations, giving an $O(mn)$ time bound.

Cognate matching provides a base matching level for analogy which is both flexible and efficient. It also, in the case of triangle cognates, allows the reuse of associations determined in the past to generate new mappings, essentially memoizing past analogical work. Cognate matching will fail to find a mappings for one of two reasons:

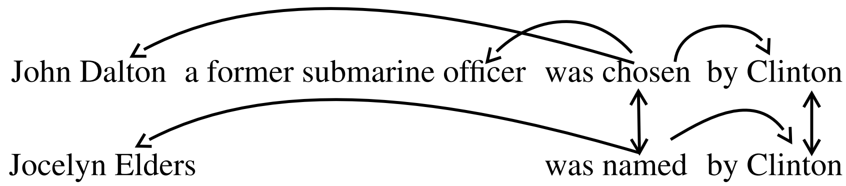
1. there are no common associations between them
2. there are no *unique* common associations between them.

Case (1) is relatively uncommon when the database has a rich associational structure (such as Wordnet). Case (2) is more common and we resolve it though the use of structure matching to combine associations and micro-relations to resolve ambiguities.

5 Structural Matching

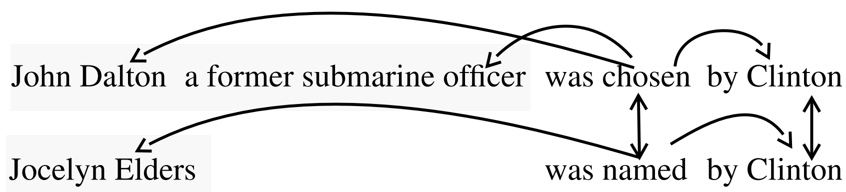
Structure matching in our database starts with a set of initial mappings derived by cognate matching and extends this mapping based on the micro-relational structure of the description. Unlike SME, our algorithm does not explicitly represent spaces of possible mappings: instead it generates a single map which will be ambiguous if there is no one systematic mapping.

The algorithm starts from an initial set of pairings determined on the basis of cognate relations; it then considers each pair and uses the micro relations of the paired individuals to specify smaller sets out of which it attempts to resolve cognates. For instance, in matching the following networks, whose micro relations are phrase ordering, we can determine two cognate relations:



the relation between 'named' and 'chosen' is found through common associations in Wordnet; the relation between the two "Clinton"'s is based on their common root (namely, "Clinton"). However, no mapping between Dalton and Elders can be found because its common association (name) is also common with Clinton.

Structure matching, however, combines the mapping between "chosen" and "named" and the micro-relation 'occurs before' to select a subset to compare for cognates:



Restricted to this subset, the conflict with "Clinton" does not exist, and the mapping between "Dalton" is easily generated "Elders" is easily generated.

This algorithm structurally matches two descriptions ***C*** and ***D*** in $O(mn^2)$ time:

```
To generate extended mappings between C and D
    Initialize I to be the cognate mappings between C and D
    for each pair <x,y> in I
        expand the pairing <x,y> by
          for each micro-relation R of x
            let E be the cognate mappings between R(x) and R(y)
            for each pairing <p,q> in E
                add the pairing <p,q> to I and
                expand the pairing <p,q> (recursively)
    Finally, return the extended set of mappings I
```

Because this algorithm relies on cognate relations as the base level for matching, it is prone to the same sort of fanciful results as cognate matching. For instance, a description of two people kissing one another and two people hitting one another would be matched based on the fact that both actions have a common synset in Wordnet (e.g. contact or touch). Again, the goal of matching is not to produce only sensible matches to to produce a reasonably sensible match given the possibilities. The task of maintaining sensibility falls on the coverage of the database and the indexing mechanisms which retrieve descriptions for possible matching.

6 What Makes a Good Match?

How do we decide what makes a good match for a description? How do we find such matches without trying to match with every description in the database? In this section, we describe how our databases are indexed, starting with a characterization of what makes a good match and then describing the indexing scheme for identifying such matches without examining the entire database.

We define a good match to have two interrelated features: systematicity and solidity. The concern for systematicity is common with nearly all work on analogy. [Falkenhainer89], [Holyoak89], and [Mitchell91] all seek systematicity, albeit in different ways.

But, as we saw above, the structural matcher is also perfectly happy to generate fanciful and unresaonable systematic matches. We introduce the criteria of *match solidity* to reflect this concern. The solidity of a particular mapping is a function of the number of unique common associations which support it: in the case of the fanciful match between 'Bill hit the table' and 'Bill painted the flower', the match is supported by a slender thread of association through the Wordnet synsets for actions and things. In contrast, a match between 'Bill hit table' and 'Bill kicked the ball' would be supported by many more unique common associations.

Another way to think about solidity is to consider it's relation to the stability of mappings when extraneous elements are added to descriptions. If we extended 'Bill hit the table' to be 'Bill grabbed the table and hit it', the match to 'Bill painted the flower' would be lost because the common association between 'hit' and 'painted' is no longer unique with another action to compete with it. On the other hand, the match between 'Bill grabbed the table and hit it' and 'Bill kicked the ball' would survive the addition of the distracting elements.

7 Indexing for Good Matches

Given this informal account of systematicity and solidity, wemove on to the question of how we use the criteria of systematicity and solidity in constructing an index for analogical matching.

We start with the fact that a match is systematic if corresponding elements are related to one another by the same micro-relations; that match is solid if the corresponding elements also have a lot of unique common associations. Uniqueness, however, is a contextual property which requires looking at the descriptions being matched. Because of this, our signature must be based just on common associations and not on unique common associations.

Given these two concerns of systematicity and solidity, we define a 'full signature' for a description ***D***, such that similar signatures indicate the potential for solid and systematic matches, as computed in this way:

```
To generate a full signature for a description D
    Create a set of keys K
```

```
for each individual i in D,
    for each micro-relation R defined for i
      for each individual j in R(i)
        for each association aj of j
        for each association ai of i
            add the key <aj,r,ai> to K
Finally return K
```

For every systematic relation carried over in a mapping between descriptions ***C*** and ***D***, there will be at least one common key in ***K***. If the systematic relation is supported by more common associations, there will be more keys in common. Given the way keys are genrated, sharing keys constitutes a neccessary but not sufficient condition for a systematic mapping; the sufficient condition is a contextual one: whether or not the common associations described by the keys are in fact unique.

The chief technical problem with this approach is that the size of the signature for a description is a function of the square of the depth of the association network times the number of elements and micro-relations in the description. This amounts to several thousand keys for even small descriptions. For practical purposes, we have found it neccessary to trim this full signature but the details of how to trim the signature are still an open problem. The general character of the solution is to trim by using a subset of each individual’s associations, rather than the whole set, to construct the set of keys. We are experimenting with several possible simplification strategies:

- terminal associations (these are the associations which have no associations themselves; this ensures that you won’t miss any matches, but ignores the solidity of the possible match)
- immediate associations (i.e. the *n* nearest associations; this handles the solidity well, but will miss some matches)
- immediate and terminal associations combined (this set is larger but avoids the blind spots of either approach; however, there is no argument that the terminal or nearest are really the best)
- ‘basic level’ associations (some interior associations (between immediate and terminal) may be best for constructing keys; such associations might correspond to the basic level categories found in human categorization behavior; or, they might be computed based on empirical analysis of either the space of examples or the space of associations)

As mentioned above, the best indexing strategy is still an open research question; in the ‘94 database, we used a combination of all associations for the ai iteration and terminal associations for the aj iteration. In the ‘95 database, with its richer associational structure, this reliably produced fanciful matches of the kind we were hoping to avoid. In our current efforts, we are using immediate associations (basically the immediate synsets in Wordnet) at the same time that we explore the possibility of technological solutions to dealing with really large index sets.

Our use of flat indexing is similar to that of ARCS [Thagard90] and MAC/FAC [Gentner91] , but differs in using a key spacethat reflects relational structure.

8 Generating the database

This section describes the generation of the datbase from input text. As mentioned above, there are really two databases: the '94 database consisting of 900,00 words of parsed news summaries with an impoverished association network; and the '95 database, partially constructed with 3 million words and a rich association network derived from Wordnet. This description describes the '95 database with some reference to our experience with the '94 database. After describing how the database is generated, we give some descriptions of matching and indexing in it.

The text database is generated from input text in a four phase process:

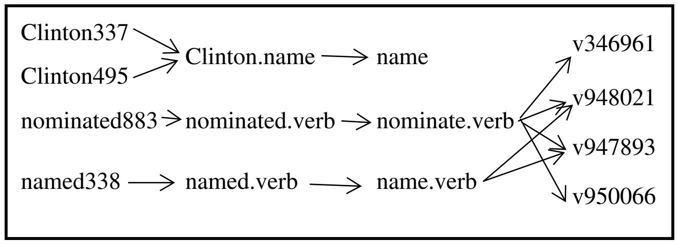
1. Tagging breaks the document into words and determines parts of speech, with an eye towards phrasing;
2. Phrasing identifies atomic phrases in the text and their heads;
3. Grounding creates new frames for head nouns and verbs in the document and connects these frames to frames in the global database;
4. Linking hooks up the local frames for a document based on possible grammatical relations between the phrases they describe.

The goal of this process is not to produce any one interpretation of the text but to generate a set of structures which will constrain and guide indexing and matching. The process is intentionally over-generative in producing multiple attachments for prepositions and ambiguous sense references. The rest of this section highlights different steps of the process:

Tagging. After the text is segmented into sentences and words (with some merging of colocations like "follow suit"), it is tagged by a probabilistic finite state parser using a hand-coded grammar and parts of speech extracted from Wordnet. The grammar associates weights with different paths through the finite state machine and the parser selects the path with the lowest weight. The associations of weights and paths allow the system to handle unusual constructions or senses while biasing against them. In tests over a subset of the Brown corpus, the tagger has demonstrated an accuracy of 95% on nouns and verbs without any initial training.

Phrasing. The phrasing process takes a sequence of tagged words and groups them into atomic phrases. An atomic phrase is a phrase which has no other phrases within it: for instance, a noun phrase without prepositions, a verb phrase with neither prepositions or objects, etc. In addition, phrasal combining cues such as complementizers and conjunctions are not included in these atomic phrases.

Tagging for Phrasing. The tagger subcategorizes some word classes in order to make it easier to extract atomic phrases from the tagged output. For instance, the category 'adverb' is subcategorized into 'ending-adverb' (adverb ending a verb phrase), 'adjverb' (adverb modifying an adjective), and 'padverb' (adverb modifying a preposition). This subcategorization also allows individuals words to be identified as more or less likely to serve particular narrow grammatical roles.



Grounding. In both databases, words which are spelled the same have a common association; however, in the '95 database, connections between words go further. Words are associated with a global frame based on the word and its part of speech, which is in turn related to a frame for the root of the word (if it has one). The frame for the root word is then related (in turn) with some number of frames representing Wordnet synsets; these frames are further associated with other synset frames based on Wordnet's 'hypernym' relations.

Linking. Once global frames are established for the words in the document and local frames constructed for the verbs and nouns, these local frames are linked together to reflect possible grammatical relations within the text. This linkage is done by an ad-hoc collection of procedures, for example:

- nouns not in prepositions look to their left for verbs of which they might be objects; if there are none, they connect to all the verbs on their right as potential subjects, otherwise, they connect to the first verb on their right as a potential subject
- pronouns look backwards for nouns and, if any are found, executes the linkage rules for that noun 'as if' it were in the place of the pronoun. With no references, they are treated as nouns.
- Every local frame also points to the frames immediately before and after it, in an effort to capture proximity information
- as well as many others...

Error and amibiguity. As mentioned earlier, the point of the linkage process is not to produce a single accurate interpretation but to capture a superset for purposes of guiding indexing and matching. So, for instance, except for the special treatment of the 'of' preposition, prepositions attach to just about everything; likewise, many erroneous subject/verb relations are introduced in order to avoid missing complicated relations involving the resolution of subjects for relative clauses.

Performance. The parser operates at roughly 750 words/min when running on a single machine; the tagging and phrasing is done by a Lisp program while the grounding and linking takes place in Scheme compiled to C [Bartlett]. The modules communicate via a LISP-based remote procedure call protocol.

9 Performance

In describing the matching performance, we will start with a discussion of certain easy cases in order to clarify how the algorithms of Section 3 apply to actual texts. After this discussion, we turn to some of the harder cases and the system's success or confusion with them. This division also corresponds to the capabilities of the '94 and '95 databases; the easier cases involved stylized phrasings which the ’94 database easily handled while the more complicated cases include the kinds of variations which many of the changes in the ’95 database were intended to handle.

When phrase structure and word choice is very stylized and similar between texts, cognate and structural matching usually has an easy time determining correspondences between texts. For instance, in the daily market reports included in the ’94 database, the following texts were easily matched:

The Dow Jones average of 30 industrials opens at a record 3734.53 Thursday after closing up 15.65 Wednesday The NASDAQ OTC composite opens at 767.89, down 1.46.	The Dow Jones average of 30 industrials opens at 3685.4 Friday after closing down 19.01 Thursday. The NASDAQ OTC composite opens at 754.34, down 8.02.

where the underlined words are all unambiguous cognates of words in the other text. These are the initial mappings determined in the first phase of the mapping procedure. In this initial set, the two occurrences of the verb ‘open’ are ambiguous, but extending the pairing of the two occurrences of ‘average’ and ‘composite’ easily determines which occurrences of ‘opens’ are analogous to each other. Extending this mapping (recursively) resolves the ambiguous mappings between the numbers on both sides, identifying the number 3734.53 on one side with 3685.4 on the other. By applying this technique to a set of document retrieved from the database, the numeric information embedded in the text can be automatically transformed into a graph and mapped against the date of the report.

In this particular case, the chief virtue of our matcher is its ability to operate on the text without priming. While it would be straightforward to construct a regular expression to extract that particular numeric value from that particular class of daily report, the structures and algorithms we have described do so directly, without any external intervention.

Both the ‘94 and ‘95 databases use the indexing scheme described above; Once the ‘94 database was loaded into memory, all 441 of the Dow Jones reports could be retrieved out of thousands of articles, based on their grammatical relations, in under 3.5 seconds on an IBM RS/6000 running AIX and using matching and indexing algorithms coded in C. The structure matcher was then able to match them all to one selected case in under a minute; if this effort was converted into associations, the mapping could be done in less than 10 seconds (since it was all just cognate relations established by previous association).

Indexing performance on the ‘95 database is not yet known, but should be available by publication time. The case is more complicated, since the associational structure is richer and the signature for each document larger, but we expect to find straightforward engineering solutions to the problems of scale.

9.1 Harder Matches

Of course, the chief reason for the easy success in the above case was that its phrasing and wording were highly stylized. Among the goals for the ’95 database were to expand matching to less stylized cases by a combination of some phrasal canonicalization (while retaining ambiguity) and the use of Wordnet to represent knowledge about word meanings. In this example, we collected various reports on administration appointments and produced representations for them which were then matched. The results were satisfying. The following four sentences, despite differences in wording and spelling, can have many of their thematic elements mapped to one another automatically without any introduction of special representations or encodings:

1. "President Clinton nominated$_1$ outspoken Jocelyn Elders$_2$ to be his surgeon general$_9$ on Thursday."
2. "Clinton named William Perry, deputy secretary under Aspin to the post"
3. "President Clinton$_3$ fired embattled FBI chief$_{10}$ William Sessions Monday and is ready to nominate$_4$ Louis Freeh, a federal judge in New York City and former FBI agent"
4. "San Antonio banker John Dalton$_5$, a former Navy submarine officer$_6$ and Democratic fund-raiser$_7$, was chosen$_8$ Wednesday by President Clinton to be Navy secretary"

In sentences 1+2, the phrase structure is exactly the same, and the connection through Wordnet handles the variation in word choice. In sentence 3, a policy of projecting subjects forward to capture embedded clauses (when there is not a conflicting subject), connects the 'President Clinton$_3$' firing Sessions to the expected nomination$_4$ of Freeh. In sentence 4, the rules for transforming the passive sorts out subject and object, allowing Dalton$_5$ to match the corresponding elements of the other sentences.

Sentence 4 also demonstrates the advantage of representing ambiguity explicitly.The sentence's representation is explicitly ambiguous about which phrase "Dalton$_5$", "submarine officer$_6$", or "fund-raiser$_7$" is the actual subject of chosen$_8$. However, when asked to determine a match with a particular second text, the unique common prototype relation pulls up the person's name in one relational context as analogous to the person's name in the other. But the representation of matching allows us to both simplify the parsing process (by postponing resolution) and simplify the matching process (by not having to consider parser errors).

While the system can do a pretty good job of figuring out *who* was nominated, only in sentences 1 and 4 is it able to figure out what position they were being nominated to fill.It misses out on 2 and 3 for two different reasons.

- For sentence 2, it does not resolve the anaphoric referent of 'the post'. Solving this requiressome mechanism for intersentential anaphora; we do not currently have one, but expect thatwe will be able to take advantage of the same representation of ambiguity used for sense and attachment to represent a space of possible referents.
- For sentence 3, the reason for the system's ignorance is the common sense or conventional inference an astute reader makes that if the sentence mentions someone being fired from a position and then describes a planned nomination, that it's likely to be a nomination to that position.

There can, of course, be no general solution to the problem of conventional knowledge which is contingent and cultural by its very nature. One possible way to allow the system to accquire this kind of knowledge would be a framework where new sentences were automatically associated --- upon arrival --- with existing sentences having similar structure. Different structures with similar meaning could then be associated with one another, and through these common associations, the new sentences would likewise be associated with each other despite the differences in phrasal conventions.

For instance, if sentences 1 and 3 were aligned to demonstrate meaning equivalence (e.g. “surgeon general$_9$” associated with “FBI chief$_{10}$”) and a subsequent sentence arrived:

5. President Clinton fired embattled surgeon general$_{11}$ Jocelyn Elders and is ready to nominate....

Its description would be aligned with Sentence 3 above and by common association, general$_{11}$ would be associated with chief$_{10}$ and the corresponding elements of other texts (of course, Jocelyn Elders would not be associated with Jocelyn Elders, but that’s how Washington works...)

Of course, it remains to be seen whether this approach to accquiring this sort of knowledge is effective; it might be that either variations are too large or their natural structure too confusing to allow this approach to succeed. One of our hopes is that the historical breadth of the ’95 database (7 or 8 years of news) will allow us to provide examples of phrasal variation over one period of time and then examine how well those examples cover other periods.

10 Ongoing and Future Work

We expect to have the ‘95 database parsed, indexed, and matching by early Spring. Along the way, we will undertake a more systematic study of indexing strategies, particularly what subsets of the full signature suffice for effective matching. In the summer, we expect to report on wider scale experiments and empirical analyses of the indexing behavior.

We are also planning to apply the structures and algorithms described here to non-textual domains, including the description of images. In this domain, individuals will correspond to salient blobs of color and texture, with micro-relations describing their geometric and topological relationships to one another. This may prove an additional challenge to our matching and indexing algorithms, since the number of micro-relations will likely be quite large (compare to the text case, where it is of the same order as the number of individual words).

11 References

[Falkenhainer89] Falkenhainer, B., Forbus, K., and Gentner,D., “The Structure-Mapping Engine: Algorithms and Examples”, *Artificial Intelligence* (41), 1989

[Holyoak89] Holyoak, K. and Thagard, P. “Acomputational model of analogical problem solving,” In *Similarity and analogical reasoining*, edited by S. Vosniadou and A. Ortony, Cambridge University Press 1989

[Thagard90] Thagard, P., Holyoak, K., Nelson, G., and Gochfeld, D., “Analog Retrieval by Constraint Satisfaction”, *Artificial Intelligence* (46).

[Lakoff87] Lakoff, G., *Women, Fire, and Dangerous Things*, University of Chicago Press, 1987

[Gentner91] Gentner, D and Forbus, K., "MAC/FAC: A model of similarity-based retrieval", Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Lawrence Earlbaum 1991