MULTILINGUAL SEMANTIC MATCHING OPERATOR IN SQL

A. Kumaran Jayant R. Haritsa

Technical Report
TR-2004-03

Database Systems Lab
Supercomputer Education and Research Centre
Indian Institute of Science
Bangalore 560012, India

`http://dsl.serc.iisc.ernet.in`

Multilingual Semantic Matching Operator in SQL

A. Kumaran Jayant R. Haritsa
Department of Computer Science and Automation
Indian Institute of Science, Bangalore, INDIA
{kumaran,haritsa}@dsl.serc.iisc.ernet.in

ABSTRACT

In an increasingly multilingual world, it is critical that information management tools organically support the simultaneous use of multiple natural languages. An essential prerequisite for efficiently achieving this goal is that the underlying database engines must provide seamless matching of attribute data across languages. We propose here a new SQL operator, called SemEQUAL, that supports simple semantic matching of multilingual attribute data. SemEQUAL leverages standard linguistic resources, such as the WordNet taxonomic hierarchy, that are available in multiple languages. We define this operator and outline its implementation using standard SQL:1999 features, but a performance evaluation on a suite of commercial database systems indicates unacceptably slow response times. However, by tuning the schema and index choices to match typical linguistic features, the performance is improved to a level commensurate with online user interaction.

1. INTRODUCTION

In an increasingly multilingual digital world[1], it is critical that information management tools, such as web search engines, *e-Commerce* portals and *e-Governance* applications, support the simultaneous use of multiple natural languages. An essential pre-requisite is that the underlying database engines (typically relational), provide the same functionality and efficiency for multi-lingual data as that associated with processing unilingual data, for which they are well-known. Unfortunately, as described below, the state-of-the-art falls short of these requirements on several counts, motivating our research on multi-lingual database systems.

From the efficiency perspective, we recently profiled in [18] the performance of standard relational operators (e.g. Select, Join) applied on multilingual data and proposed efficient storage formats to make the operators *language-neutral*. Subsequently, from the functionality perspective, we introduced a new SQL multilingual operator called LexEQUAL [19], for *phonetic* matching of specific types of attribute data across languages, and proposed techniques to optimize its performance along the lines of those that are used in monolingual world [13]. In this paper, we take the next logical step in supporting multilingual functionality, by proposing SemEQUAL, a *semantic* operator for matching text attribute data across languages based on *meaning*. For example, to automatically match the English noun *mathematics*, with *mathématiques* in French or கணிதம் (transliterated as *kanitham*) in Tamil.

1.1 The SemEQUAL Operator

To determine semantic equivalence of word-forms across languages and to define the SemEQUAL operator, we take recourse to *WordNet* [30], a standard linguistic resource that is available in multiple languages and, very importantly from our perspective, features inter-language semantic linkages. After integrating WordNet with the database platform, two alternatives arise with regard to the SemEQUAL implementation: a *derived-operator* approach, wherein SemEQUAL is expressed in terms of standard SQL scripts, or a *core-operator* implementation, where SemEQUAL is internally visible to the database engine. The latter approach has its obvious benefits in terms of improved efficiency, but requires an involved and time-consuming software engineering exercise, making it feasible only in the long-term. In contrast, the derived-operator approach can be used immediately if the performance can be made acceptably fast – we investigate this possibility here.

Specifically, we first analyse the performance of SemEQUAL, expressed using standard SQL:1999 features, in relational database systems. A direct implementation on three commercial database systems indicates that supporting multilingual semantic processing is unacceptably slow. However, by tuning the schema and access structures to match the characteristics of WordNet, we are able to bring the response times down to *a few milliseconds*, which we expect to be sufficient for most applications. We emphasize that our focus in this paper is to demonstrate the efficient implementation of SemEQUAL using existing database technologies.

In short, we quantitatively demonstrate that multilingual semantic matching may be implemented on today's database systems, using standard language features, achieving performance levels commensurate with online user interaction.

[1]Two-thirds of current internet users are non-native English speakers [23] and it is predicted that the majority of web-data will be multilingual by 2010 [29].

1.2 A Multilingual *e-Commerce* Example

Consider a hypothetical *Books.com*, with a sample multilingual product catalog, as shown in Figure 1, where books in different languages are featured.

Author	Author_FN	Title	Price	Category
Descartes	René	Les Méditations Metaphysiques	€ 49.00	Philosophie
நேரு	ஜவஹர்லால்	ஆசிய ஜோதி	INR 250	சரித்திரம்
Adams	Laurie S.	Arte Di Rinascita Italiana	€ 75.00	Arti Fini
Lebrun	François	L'Histoire De La France	€ 19.95	Histoire
Durant	Will/Ariel	History of Civilization	$ 149.95	History
नेहरू	जवाहरलाल	भारत एक खोज	INR 175	इतिहास
Franklin	Benjamin	Un Américain Autobiographie	€ 25.00	Autobiographie
Gilderhus	Mark T.	History and Historians	$ 49.95	Historiography
காந்தி	மோகன்தாஸ்	சத்திய சோதனை	INR 250	சுயசரிதம்

Figure 1: A Multilingual *Books.com*

Currently, a query with (Category = 'History') selection condition, would return only those books that have *Category* as `History` in English, although the catalog also contains history books in French, Hindi and Tamil. A multilingual user may be better served, however, if all the history books in all the languages (or more likely, in a set of languages specified by her) are returned. A sample SQL query using the proposed SemEQUAL operator and the corresponding result set, as given in Figure 2, would therefore appear to be desirable[2].

```
SELECT Author,Title,Category FROM Books
WHERE Category SemEQUAL 'History'
InLanguages {English, French, Tamil}
```

Author	Title	Category
Durant	History of Civilization	History
Lebrun	L'Histoire De La France	Histoire
நேரு	ஆசிய ஜோதி	சரித்திரம்

Figure 2: Basic Semantic Selection

Further, the SemEQUAL operator may be generalized to return not just the tuples that are equivalent in meaning, but also with respect to semantic relationships, such as *generalization*. For example, consider a variation of the operator, specified as SemEQUAL_2, where the user may specify retrieval of all `History` books, including those under the subclassifications of `History`, as shown in Figure 3. Note that in addition to the original results, three additional tuples are also reported in the output[3].

In the following sections, we take the generalized operator – SemEQUAL_2, as the multilingual semantic matching operator, as will be discussed in Section 2.2.

As a final note, in this paper we focus only on multilingual domain, though such an operator may be applicable in any domain that has a well-specified taxonomic hierarchy.

[2]The third record in the result set is a Tamil book, with (transliterated) category value *Charitram*, meaning `History`.

[3]Both `Historiography` (*the art and science of history making*) and `Autobiography` are considered specialized branches of `History`. The last record in the result set is a Tamil book, with (transliterated) category value as *Suyacharitam*, meaning `Autobiography`.

```
SELECT Author,Title,Category FROM Books
WHERE Category SemEQUAL2 'History'
InLanguages {English, French, Tamil}
```

Author	Title	Category
Durant	History of Civilization	History
Franklin	Un Américain Autobiographie	Autobiographie
Gilderhus	History and Historians	Historiography
Lebrun	L'Histoire De La France	Histoire
நேரு	ஆசிய ஜோதி	சரித்திரம்
காந்தி	சத்திய சோதனை	சுயசரிதம்

Figure 3: Generalized Semantic Selection

1.3 Our Contributions

To summarize, our main contributions in this paper are:

- Motivating and formalizing the notion of multilingual semantic equality at the granularity of database attributes, based on the WordNet multilingual resource.
- Integration of WordNet with relational systems for query processing and a *derived-operator* implementation of SemEQUAL, using standard SQL features.
- Optimizing the performance of the SemEQUAL operator on commercial database systems, based on WordNet linguistic features, to a level that appears sufficient for *e-Commerce* deployments.

1.4 Organization of the Paper

The remainder of this paper is organized as follows: Section 2 details our definition for semantic matching operator and its implementation. In Sections 3 and 4, we present our experimental evaluation and the results, respectively. Section 5 provides a survey of related research and Section 6 concludes the paper, highlighting our results and future research avenues. Finally, a review of WordNet is provided in Appendix – A (which may be skipped by an informed reader).

2. MULTILINGUAL SEMANTIC MATCHING

In this section, we specify the semantics of the SemEQUAL operator and describe our strategy for implementing the operator, in our derived-operator approach, using standard SQL constructs.

2.1 Some Basic Definitions

Let the database contain tuples that include attributes that are earmarked for semantic matching. Let $\mathcal{D}$ be the domain with atomic values (text strings) from which the values of attribute are taken. Let $\mathcal{H}$ be a well defined taxonomic hierarchy (a collection of directed acyclic graphs) that define *is-a* relationships among the atomic values of the domain $\mathcal{D}$. Given an atom x and a taxonomic hierarchy $\mathcal{H}$, the transitive closure of x in $\mathcal{H}$ is unique, and is denoted by $\mathcal{TC}_{\mathcal{H}}(x)$. Based on the above, we provide the following definitions, to express *semantic matching* in a domain.

Definition 1: Given a taxonomic hierarchy $\mathcal{H}$ in domain $\mathcal{D}$ and two nodes A and B in $\mathcal{D}$, we define A *is-a* B, iff $A \in \mathcal{TC}_{\mathcal{H}}(B)$.

Definition 2: Given a taxonomic hierarchy $\mathcal{H}$ in domain $\mathcal{D}$ and two sets of nodes $\mathcal{A}$ and $\mathcal{B}$ in $\mathcal{D}$, we define $\mathcal{A}$ *is-a* $\mathcal{B}$, iff $\mathcal{A} \subseteq \mathcal{TC}_{\mathcal{H}}(\mathcal{B})$.

Definition 3: Given a taxonomic hierarchy $\mathcal{H}$ in domain $\mathcal{D}$ and two sets of nodes $\mathcal{A}$ and $\mathcal{B}$ in a domain $\mathcal{D}$, we say $\mathcal{A}$ *is-possibly-a* B, iff $\mathcal{A} \cap \mathcal{TC}_{\mathcal{H}}(\mathcal{B}) \neq \phi$.

If the atomic values have unique semantics in the domain $\mathcal{D}$ and taxonomical hierarchy $\mathcal{H}$ comprises only of trees then definition 2 provides an unambiguous semantic match between the domain elements. When one or more of the above conditions fail to hold, then the definition 3 provides a weaker notion of equality[4]. In our implementation of SemEQUAL, we use the weaker definition 3, as the linguistic atomic values are not unique in their meaning.

2.2 Definition of SemEQUAL

In this section, we define and outline implementation of Multilingual semantic matching using the SemEQUAL operator, leveraging on common linguistic resource (specifically, WordNet) for mapping text strings to a set of canonical semantic primitives. For the discussions in the rest of this paper, familiarity with WordNet is assumed; we provide basics of WordNet in Appendix A, and refer interested readers to [8, 30] for further details. For following the query processing issues, it is suffice to understand that WordNet contains a lexical matrix that maps a text string to a set of canonical sementic primitives and a taxonomical hierarchy for all noun semantic primitives, modeled as a collection of directed acyclic graphs. Further, efforts are underway to link WordNets of different languages, by linking corresponding semantic primitives.

Consider a canonical SemEQUAL query predicate[5]

$$\{Attribute\}\ \text{SemEQUAL}\ \{\texttt{Constant}\}\ \text{InLanguages}\ L_1, L_2, \ldots, L_N$$

Let L_{in} denote the language in which the Constant is specified, $S^c_{L_{in}}$ denote the set of synsets of Constant `c` in language L_{in}, $S^c_{L_{out}}$ denote the set of matching synsets of $S_{L_{in}}$ in target language L_{out}, and $TC(S^c_{L_{out}})$ denote the transitive closure of $S^c_{L_{out}}$ in the WordNet of language L_{out}. Then, $\bigcup_{out} TC(S^c_{L_{out}})$ denotes the union of all synsets in the transitive closures of $S^c_{L_{out}}$ in the respective WordNet of the target languages. Further, let the value of the Attribute, in the database tuple currently under consideration, be denoted by *data*, its language by L_{data}, and the set of synsets of *data* w.r.t. L_{data} by $S_{L_{data}}$.

With this notation, the SemEQUAL operator returns true only if $S_{L_{data}} \bigcap (\bigcup_{out} TC(S^c_{L_{out}})) \neq 0$.

The SemEQUAL operator has the following properties:

Property 1: SemEQUAL is not reflexive.

Property 2: SemEQUAL commutes with selection, projection or join operators. It also commutes with aggregate operators, as long as the aggregation is defined on the semantic attribute that is being compared.

The property 1 follows from the asymmetry of the operator and the property 2 follows from the fact that the operator does not modify the record, but merely acts as a filter. These properties may be exploited by the optimizer to select an efficient query plan. Due to the lack of space, the details are deferred to a forthcoming technical report.

The SemEQUAL operator may be implemented similar to the popular *equijoin* operator, though the operator implementation involves two distinct steps: computation of the transitive closure of constant `C` and testing if any of the values of $S_{L_{data}}$ is a member of the set $\bigcup_{out} TC(S^c_{L_{out}})$. Computing the transitive closure of the constant `C` in a relation system is expensive, but may be implemented using the standard SQL:1999 recursive SQL constructs or tree traversal algorithms. After computing the transitive closure of the constant `C` the operator would cycle through the inner table (the LHS operand), outputting all records for which SemEQUAL returns a value `true`. This second step may be implemented using well-known techniques, such as, building a hash-tables for elements of the closure set and the set-membership of a value may be found in one hash probe.

2.3 Implementing Multilingual SemEQUAL

Our strategy for matching multilingual attributes hinges on converting the query string and data strings to *sets of WordNet synsets* in a canonical form, using appropriate linguistic resources (WordNet, in our case) and processing the set-membership predicate efficiently.

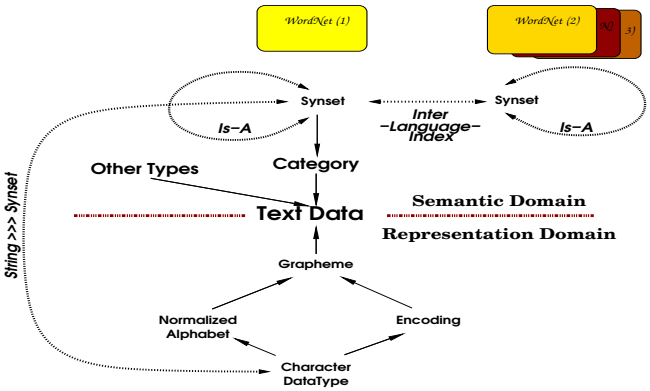

Figure 4: Framework for Semantic Matching

For the first step, the framework we use for conversion is shown in Figure 4. The text strings stored in the databases (represented by the lower half of the figure) are converted into synsets, using WordNet lexical matrix function. Though the database attributes can vary from simple attributes to full documents, we consider only matching of simple multilingual attributes storing noun word-forms.

Once transformed to synsets, the semantic primitives may be augmented with specializations, by traversing the taxo-

[4]While the definition 2 provides strong semantic matches (such as, *Floppy Disk Drive* is a *Computer Peripheral*), the definition 3 provides only for possible semantic equality under some specific interpretations (such as, *Mouse* could possibly be a *Computer Peripheral*).

[5]We consider only multilingual selection queries here – the extensions for join are elaborated in a forthcoming technical report.

nomical hierarchies of WordNet that are stored in the database. In addition to the intra-language (`Is-A`) relationships available to specialize within a language, the inter-language (`Inter-Language-Index`) relationships are also available in the interlinked WordNets[6], to move to taxonomic hierarchies of target languages. Thus the rich semantic interrelationships between the synsets may be used for computing the transitive closure of a given synset, spread over a set of user-specified target languages.

2.4 Semantic Matching Function

The SemEQUAL function to match a pair of multilingual strings is outlined in Figure 5.

SemEQUAL ($String_{Data}$, $String_{Query}$, $\mathcal{T}_\mathcal{L}$)

Input: Strings $String_{Data}$, $String_{Query}$
Set of Target Languages $\mathcal{T}_\mathcal{L}$

Output: `TRUE` or `FALSE`
[*Optional*] Gloss of Matched Synset

1. (L_D,L_Q) ← LangOf ($String_{Data}$, $String_{Query}$);
2. ($\mathcal{W}_\mathcal{D}$,$\mathcal{W}_\mathcal{Q}$) ← WordNetOf (L_D,L_Q);
3. $\mathcal{S}_\mathcal{D}$ ← Synset of $String_{Data}$ in $\mathcal{W}_\mathcal{D}$;
   $\mathcal{S}_\mathcal{Q}$ ← Synset of $String_{Query}$ in $\mathcal{W}_\mathcal{Q}$;
4. $\mathcal{TC}_\mathcal{Q}$ ← TransitiveClosure($\mathcal{S}_\mathcal{Q}, \mathcal{T}_\mathcal{L}$);
5. **if** $\mathcal{TC}_\mathcal{Q} \cap \mathcal{S}_\mathcal{D}$ **is not empty then**
   return `TRUE` **else return** `FALSE`;
6. [Opt.] **return** Gloss of the Matched Synset;

TransitiveClosure (S, $\mathcal{T}_\mathcal{L}$)

Input: String S, Target Language Set $\mathcal{T}_\mathcal{L}$

Output: The specializations of $\mathcal{S}$

1. L_S ← Language of String S;
2. $\mathcal{W}_\mathcal{L}$ ← WordNet of Language L_S;
3. $\mathcal{S} \leftarrow \mathcal{S}_\mathcal{C}$ ← Synsets of S in $\mathcal{W}_\mathcal{L}$; $\mathcal{S}_\mathcal{N} \leftarrow \phi$;
4. **repeat until no change in** $\mathcal{S}$:
5. **for every element** s **in** $\mathcal{S}_\mathcal{C}$
6. $\mathcal{S}_\mathcal{N} \leftarrow \mathcal{S}_\mathcal{N} \cup$ hypernyms of s
   $\cup$ Synsets linked to s through
   InterLangIndex to $L \in \mathcal{T}_\mathcal{L}$ *that are not yet traversed to*;
7. $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_\mathcal{N}$; $S_C \leftarrow \mathcal{S}_\mathcal{N}$; $\mathcal{S}_\mathcal{N} \leftarrow \phi$;
8. **return** $\mathcal{S}$

Figure 5: Semantic Matching Algorithm

The SemEQUAL function takes as input, strings $String_{Data}$ and $String_{Query}$. The transitive closures are computed in the taxonomic network obtained from crosslinked WordNet hierarchies, using the TransitiveClosure function. Once the transitive closure is computed, set-processing routines are used for computing intersections. The output is `TRUE` if the specified matching condition is met, else `FALSE`. Since the query string, $String_{Query}$, may match on any one of the several synsets (which are possible semantics of the same word form), SemEQUAL may be made optionally to return the *Gloss* of the synset on which the $String_{Query}$ is matched.

The TransitiveClosure function computes the transitive closure of a given string, by computing all the sub-classes of a synset node corresponding to the input string. The subclasses are computed using `Is-A` relationships within a language and using `Inter-Language-Index` across languages. In the implementation, only the WordNets corresponding to the target languages specified in the query are traversed.[7] Further, once a traversal to a target language (in line 6 of TransitiveClosure algorithm) has been made, back traversals to WordNets of any other languages are not permitted, in order to avoid unnecessary repetitions. Due to such restrictions, the algorithm scales linearly with the number of WordNets added to the query[8].

Note that a full transitive closure computation may not be essential for testing non-empty intersection of sets of synsets corresponding to the input multilingual strings. Specifically, during the computation of the closure of $\mathcal{TC}_\mathcal{Q}$ (in Line 6 of TransitiveClosure algorithm), as soon as a node that is in S_D is produced in the closure, the matching may be stopped, outputting a `TRUE`. However, if a large table of values is checked for semantic equality with a given string $String_{Data}$, then apriori computing the full closure of $String_{Data}$ may help to reduce the overall response time since it is computed only once for the query.

A related issue here is that we match the data string against the the *union* of the transitive closures of the synsets of the query string in all target languages. This is indeed wasteful since, at least for text attributes, potential matches for a data string in a particular language can only be with the synsets in *that* language. The problem, however, is that current technology does not easily support automatic language identification for attribute data, and therefore, the matching has to be perforce done in the indicated manner.

2.5 Following through with an Example

We present a simple *derived-operator* implementation methodology for matching multilingual data to implement the SemEQUAL function. The linguistic resources (WordNets in multiple languages) are first stored in the database tables. Then, SQL:1999 recursive constructs are used to compute the transitive closure of the synsets corresponding to the data string. The set membership of the query string in the transitive closure of the data string is specified using the SQL `IN` predicate.

[6]Several efforts are underway to link up WordNet taxonomic hierarchies of different languages. The *European WordNet* (EWN) [6] – a major initiative that includes all major European languages, keeps the basic taxonomic hierarchies the same, and has defined links to map synsets in one language to those in the other. Similarly, the *Indo-WordNet* (IWN) [17, 3] is being developed in a common framework, with a stated goal of sharing the same taxonomic structure in all 15 of the official Indian languages, and with additional links to the English WordNet.

[7]To specify all languages, a wildcard `*` is used in the `InLanguages` clause, in which case all installed WordNets are utilized.

[8]While it is possible to compute the *fix-point* of all cross-lingual traversals, the complexity of the algorithm becomes extremely high; in addition, the result sets could become more fuzzy due to the unevenness between the taxonomic hierarchies of different languages.

For example, the user query to retrieve all `History` books including its subclassifications, shown below,

```
SELECT Author, Title from Books
WHERE Category SemEQUAL ALL 'History'
InLanguages {English, French, Tamil}
```

is mapped to the query:

```
WITH Descendants (child, lang)
  (SELECT WL.sub, WL.lang
   FROM WordNet WL
   WHERE WL.super = 'History'
   AND ( WL.lang = 'English'
      OR WL.lang = 'French'
      OR WL.lang = 'Tamil')
  UNION ALL
   SELECT WL.sub, WL.lang
   FROM WordNet WL, Descendants Dec
   WHERE WL.parent = Dec.child
   AND WL.lang = Dec.lang)
SELECT Author, Title from Books
WHERE Category IN
  (SELECT child FROM Descendants)
```

The user query effectively translates to the following SQL query, where the IN clause has been expanded to indicate the computed transitive closure.

```
SELECT Author, Title from Books
WHERE Category IN {
    'History','Autobiography','Memoir',...
    'Histoire','Autobiographie','Mémoire',... }
    'சரித்திரம்','சுயசரிதம்',...}
```

Here, the values `Autobiography`, `Memoir`, etc., are a few of the 76 subclasses of `History`, in English WordNet, and `Autobiographie`, `Mémoire`, சரித்திரம், சுயசரிதம் etc., are equivalent synsets in the French and Tamil WordNets. Note that any conjunction (disjunction, respectively) of SemEQUAL predicates can be handled by computing the intersection (union, respectively) of closures for the IN predicate.

We would like to emphasize that the size of the closure thus computed depends on the query string and the characteristics of the WordNet taxonomic hierarchy. The problem of computation of closures in relational systems has been analyzed in [15, 16], where the authors show poor performance of the relational database systems in computing closures. Though a variety of sophisticated algorithms for transitive closure have since been presented in the literature (e.g. [1]), the current implementations of transitive closure algorithms in relational database systems are still recognized to be generally slow. Once computed, the closure set is passed on to the IN predicate, which is well optimized in RDBMS. This operator contributes very little to the overall processing time of the SemEQUAL query (less than 1% of the query run time in our experiments). Thus, the overall performance of the SemEQUAL query primarily depends on the speed of computing the recursive SQL operator.

3. EXPERIMENTAL STUDY

In this section, we describe our experimental setup to measure the performance of the SemEQUAL operator, on a suite of commercial database systems.

3.1 System Setup

For the performance experiments, a standard Pentium IV workstation with 512 MB memory running Windows NT operating system, was setup. Three commercial database systems, IBM DB2 Universal Server (version 7.1.0), Microsoft SQL Server (version 8.00.194), and Oracle 9i (version 9.0.1), were installed with default configurations[9]. Of these three, DB2 and Oracle support recursive SQL natively, while the functionality is simulated in SQL Server, using scripts. In subsequent sections, the database platforms are identified randomly as *System A*, *System B* and *System C*, to protect the identities of the systems.

3.2 WordNet Storage

The WordNet data was loaded in the database systems using the simple hierarchy table method (with `(Parent, Child)` relationships) for the storage of the WordNet taxonomic relationships. We calculated the storage space requirements of each WordNet to be about 2.5 MB, based on the profile of English Wordnet (shown in Table 1) and assuming that the WordNet of each language will be similar to English WordNet when fully developed. Storing index structures takes about 1.5 MB of additional storage space, raising the total to 4 MB of storage space for each language. Further, it should be noted that the WordNet for a language based on non-Latin scripts has to be stored in Unicode [27], essentially doubling the storage requirement[10]. Therefore, we estimate that, in general, any WordNet can be accommodated within 10 MB of disk space.

3.2.1 Profile of WordNets

The entire set of noun taxonomic hierarchies of WordNet (Version 1.5), totaling about 110,000 *word forms*, 80,000 *word senses* and about 140,000 relationships between them, was loaded on the database systems. In addition, a sample of Euro WordNet was downloaded and stored. The content statistics associated with the Euro-WordNet and Indo-WordNet, were obtained and used for analysis.

Though the WordNets of different languages are at different stages of development, The available WordNet data were analyzed to profile the structural and storage characteristics of each. The salient statistics are given in Table 1 for the respective WordNets. Clearly, the English WordNet is the most developed, followed by the European WordNets, and finally the Hindi WordNet. As can be seen in Table 1, the statistics of the existing hierarchies (such as, *Average Fan-out*, *Average Word-Forms per Synset*, etc.) indicate a very close match between English and European WordNets, arising out of similar graph characteristics among these WordNet hierarchies. The Hindi WordNet has an average fan-out statistic that is nearly double that of English, perhaps due

[9]The public-domain database systems, MySQL and PostgreSQL, were not considered since they do not support transitive closure computation.

[10]Though compression techniques [18] may be used to reduce the space usage, for this study, we used only basic Unicode.

Characteristic	English	French	German	Spanish	Hindi
Word Forms (Words)	114,648	32,809	20,453	50,526	22,522
Word Sense (Synsets)	80,000	22,745	15,132	23,378	7,868
Average Fan-out	2.236	2.176	2.301	2.360	3.889
Average Word Forms per Synset	1.985	1.442	1.352	2.162	2.286
Equivalence Relations per Synset(to English Synsets)	1.000	0.999	1.080	0.908	Not Available

Table 1: Statistical Profile of WordNets [3, 28, 7]

to clustering of word-senses on some parent nodes during the development period.

Another interesting fact is the *Equivalence Relationships per synset* to English, in Euro WordNets; there exists a near identity relationship between number of synsets of a specific language and the number of equivalence relationships, indicating near-identical sets of synsets in those WordNets, with respect to English WordNet. Such statistics confirm our intuition that the development of WordNets closely follow the English WordNet, structurally and semantically. In addition, since both Euro-WordNet and Indo-WordNet have conformance to English WordNet as their stated design goal, it is reasonable to expect their structures to be similar to that of English WordNet, when fully developed.

3.2.2 Simulating the Crosslinked-WordNets

To profile the performance of SemEQUAL working with *fully developed* linguistic resources, we simulated the crosslinking between WordNets by assuming that every synset in a non-English language is connected to a corresponding synset in English. Supporting such a methodology is the near identical taxonomic hierarchies among Euro WordNets and their near identity relationship with taxonomic hierarchy of English WordNet, as mentioned above. Thus, the English WordNet is replicated in Unicode (to model the Unicode representation of non-Latin-script data) and a `inter-language-index` is created between every pair of corresponding synsets between the original English WordNet and its Unicode replica. The resulting taxonomic hierarchy is used in the following experiments.

3.3 Queries Performed

Since the closure computation takes more than 99% of the runtime of the SemEQUAL query, we used queries that compute the transitive closures on the above taxonomic hierarchy, as the base query for performance measurements. A SQL:1999 transitive closure query (as shown in Section 2.5) was used, with different query strings representing closures of different sizes in the taxonomic hierarchy.

3.4 Metrics Measured

In all the experiments, we measured the wall-clock runtime of a given query on the given data set. The queries were run in an SQL or a programming language environment (C or PL/SQL), as appropriate. The test machine was quiesced except for the database system under study and the queries were run cold. The average runtime from several identical runs was taken as the runtime of a specific query (the graphs show mean values with relative half-widths about the mean of less than 5% at the 90% confidence interval).

4. RESULTS AND ANALYSIS

In this section, we focus on the performance of the different commercial database systems in computing the closure. We show the magnitude of inefficiencies and subsequently outline two performance optimization techniques.

4.1 Closure Computation – Baseline

In the first suite of experiments, the interlinked WordNet taxonomic hierarchy (in Unicode format to simulate multilingual environments) was stored using the hierarchy table method, as mentioned previously in Section 3.2. The query strings for the experiment were chosen such that they have closures of varying sizes, from very small to nearly half of the noun forms. Such selection provides a sufficiently wide range for calibrating the performance of the closure computation. Sample closure sizes (that is, the cardinality of the result set) for selected query strings are given in Table 2.

Semantic Primes	Size of Closure
Time	155
Shape	585
Process	2041
Fauna	4126
Knowledge	5340

Table 2: Closure Size for Semantic Primes

The *SQL-Baseline* performance (in seconds) for the basic closure computation is given in Figure 6 (the graph is shown in *log-log* scale). The performance of the query, both without and with indexes on the attributes are provided. For those experiments with index, B-Tree indexes on the parent and child attributes were created. As can be observed here, the closure computations for all the systems take in the order of tens to hundreds of seconds without index and between a few hundredths of a second to a few seconds even with index, making the performance unsuitable for *e-Commerce* deployments, if the size of the closure exceeds a few hundred items. We observed that the differences in run-times in different database systems are primarily due to differences in the implementation of transitive closure algorithms. For example, two systems used *breadth-first-search* for expanding the result set, while the third used *depth-first-search*. One of the systems detected cycles during traversal and exited gracefully, while the other two ran indefinitely.

While the slow performance is expected due to the repeated scan of the table for every element in the *in-progress* closure set, we found that the query plans always used *nested-loops* join, irrespective of the size or profile of the data (such as, size of the table containing the taxonomical hierarchy, expected size of the result set, etc.), or user-provided optimizer hints. While using indexes, in all the systems, the query

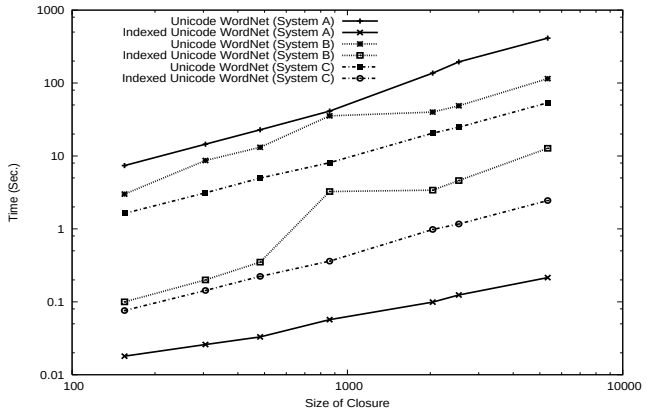

Figure 6: Performance of Computing Closure

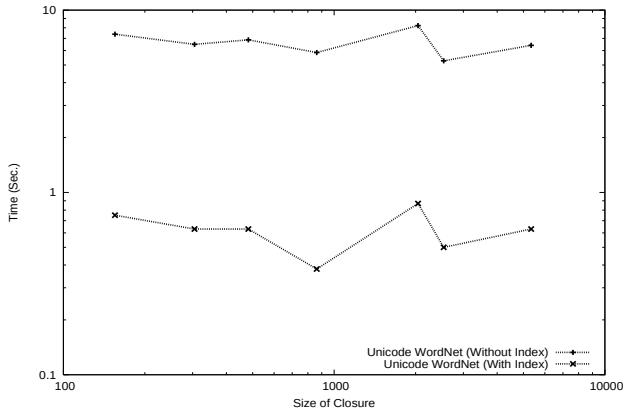

Figure 7: Performance with Precomputation

execution plans indicated that the index on the taxonomic table was made use of for scanning the outer (taxonomic) table, but no optimizations (such as, sorting, maintaining incremental views etc.) were used for efficient scanning of the inner (temporary) table.

Thus, we observe that the standard storage and indexing of WordNet in the database system is not sufficient to support the performance needed for online deployments. In the following sections, we outline two optimizations that improve the closure computation performance by another 1 to 2 orders of magnitude. We hasten to add that though the optimizations are not novel, our objective is to demonstrate that SemEQUAL may be efficiently implemented with standard SQL features on currently available commercial database systems. In the subsequent experiments, we focus on only *System B*, which exhibited the worst performance in the experiments so far.

4.2 Optimization #1: Precomputed Closure

We used the following technique for our first optimization – *pre-computing and storing the closures* of every element in WordNet explicitly, as the immediate children of corresponding element, so that the closures could be found with a simple linear scan of the enhanced table. However, any improvement in performance is achieved with a significant overhead in storage space – for English WordNet hierarchy, the storage of the taxonomic tables are increased by about 50 times, to roughly 120 MB, to store the precomputed closure with each element. We ran the transitive closure query on the resulting data set, and the performance is presented in Figure 7 (the graph is shown in *log-log* scale). We observe here an improvement in performance, to about 4 seconds for English WordNet, and about 7 seconds for the Unicode WordNet. Though the runtimes are now reduced by two orders of magnitude from the Baseline, such run times are still unacceptable for on-line interactions. Understandably, the closure computation takes approximately the same time for all sizes of the closure, since they all involve only a single scan of the table.

Further, we built an index on the parent attribute on the pre-computed table with the expectation that the performance would improve tremendously, since with one index scan, the closure can be computed. The B-Tree index built on the parent attribute took a further 45 MB of storage space. When the performance experiments were repeated with the index (also shown in Figure 7), as expected, the runtime of closure computation, for both the English and the Unicode WordNets, were reduced by 3 orders of magnitude, to the order of milliseconds for Latin-script based languages and to just under one second for Unicode based languages.

In summary, while there is tremendous improvement in the performance, these gains come at an enormous storage cost. Also, the runtimes are still not sufficiently small for Unicode based WordNets.

4.3 Optimization #2: Reorganizing Schema

In this section, we outline an alternative optimization strategy for improving the performance of closure computation, without the large space overheads of pre-computed closures. Our strategy is based on leveraging the *distribution* of synsets in the WordNet hierarchy to reduce the calls to the expensive recursive SQL statements.

We first computed the fan-out of subclasses for every parent node in English WordNet, as shown in Figure 8 (the graph is drawn to a *log-log* scale). The plot of the fan-out exhibits a characteristic *power-law* distribution with an exponent of -2.75. Further analysis indicate that only a small number of synsets (*less than* 10%) have a large number of children (*more than* 16), with the large majority having only a few children[11].

The above distribution suggests a new, more efficient organization of WordNet hierarchy, where a certain number of sub-classes may be in-lined. We chose to inline upto 16 subclasses of a given synset in the parent table, reducing the number of records in the taxonomic table to about a

[11]It is interesting to observe that the plot of the fan-outs in Hindi WordNet (also shown in Figure 8) exhibits a very similar profile to English WordNet, differing only in scale. Such similarity across widely different languages suggests the applicability of power-laws in linguistic domains. Further, we expect similar distributions for the other European languages as well.

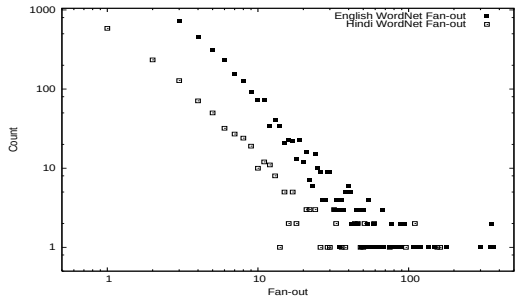

Figure 8: Histogram of WordNet Fanout

tenth of the original table. All synsets with greater than 16 subclasses are left in the original table. The closure computation algorithm is modified to access the inlined table (for all classes with less than 16 subclasses), or the original table (otherwise). The overall size of the table (in terms of number of tuples) reduces by about 90%, though the storage size remains the same as the Baseline[12]. However, the children of a given synset may be found with a few accesses, with a clustered index on the table, as against a table scan in Baseline. With the new WordNet storage organization and query execution semantics, we repeated the performance experiments on the reorganized tables, in two sets – first without an index on the hierarchy table, and next, with a clustered index on the parent synset attribute.

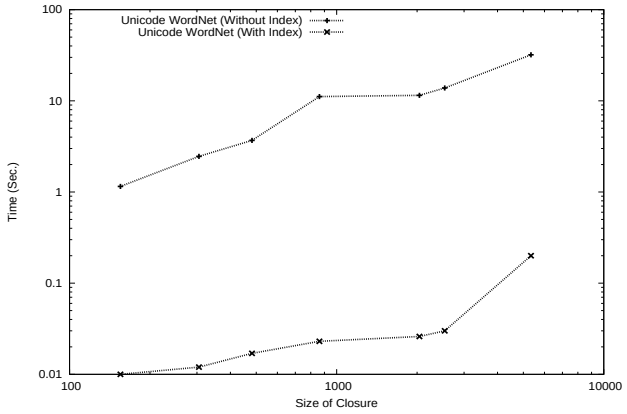

Figure 9: Performance with Reorganization

The performance of both sets of queries are shown in Figure 9 (the graph is drawn to a *log-log* scale). The performance of Baseline is speeded up by 2 orders of magnitude on plain tables and by 3 orders of magnitude on indexed tables. Specifically, we find that the time to compute the closure with index on re-organized tables is in the order of *a few milliseconds both for the Latin-based and Unicode WordNets* for closures of size upto $2,500$, well suited for any *e-Commerce* deployments.

More significantly, the storage requirements do not go up with the reorganized schema, as the same contents are reorganized in different schema format, but not replicated or deleted. Though there are slight overheads in accessing two tables, these are insignificant compared with computing closures on a large original table. Thus, we show that closures can be computed efficiently on WordNet taxonomic hierarchies, without excessive space tradeoffs.

[12]The required storage is about 4 to 5 MB for ASCII-based languages and about 10 MB for Unicode-based languages.

4.4 The Typical Closure Size

In this section, we establish the typical size of the closure, to justify the viability of the above performance figures. We selected a combination of the top-100 most used nouns in English [2] and the top-50 nouns that are used in popular web-search engines [31] and computed the size of *their* closures in English WordNet. The average size of closures of such noun forms provide a reasonable estimate on a typical closure size that would be computed for a user semantic query. Due to space limitations, we provide only a partial list of the query nouns and their closures sizes in Table 3.

Common Query Nouns	Size of Closure
Baby, Children, Kids	107
Business, Company, Organization	488
Education, Training	969
Food, Drink	2,570
Sex	78
Music, Song	548
Travel, Holiday	404
Average Closure Size	625

Table 3: Closure Size for Common Query Nouns

As can be seen, the average closure size for the most used noun forms is about 625. Hence, it is realistic to use the computation of a closure of size about 2,000, assuming that in the multilingual world, users would want answers to be computed in at most 3 or 4 languages. We observe that for computing a closure of size $\approx 2,000$, it takes about 100 seconds (without index) and upto 5 seconds (with index) for baseline performance. With the additional optimizations, namely *pre-computed closures* and *re-organized tables*, the performance of the closure computation in the same range is brought down to *a few milliseconds*.

4.5 Scaling of Performance with Languages

In this section, we explore how the performance degrades with the number of languages being considered for query processing. In the following experiments, we used *System B* for computing the transitive closure of a node that has a closure of size of ≈ 600. The interlinked WordNet hierarchy of multiple languages is simulated by replicating English WordNet for each language with `inter-language-indexes` between each corresponding synsets.

The runtimes for the typical query under different methodologies are given in Figure 10. While the baseline performance increases quadratically with *number of languages*, a near-linear increase is observed in both *pre-computed closure* and *re-organized tables* methodologies. Further, even with about 8 languages, the runtimes for the typical query remained within a few tens of milliseconds.

Thus, we show that a new semantic multilingual matching functionality may be added to the relational database systems by integrating standard linguistic resources, and lever-

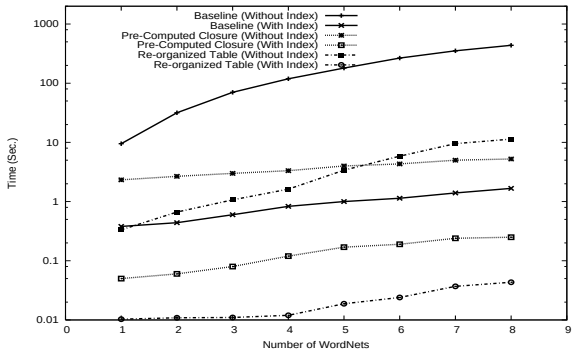

Figure 10: Scaling of Computing Closure

aging only on existing SQL features. Further, we show the performance of such matching may be optimized to support online-user interactions.

5. RELATED RESEARCH

To the best of our knowledge, our approach to multilingual semantic matching of attribute data – by integrating the linguistic ontological resources with the database engine and fine tuning it for OLTP type environments, has not been discussed, previously in the literature.

Existing Support in Relational Databases
Since no semantic processing requirements are specified in the SQL standards, each vendor has taken a different approach based on a suite of NLP techniques. While these techniques are effective with large documents, they are not well-suited for attribute granularity semantic processing, or for OLTP environments. The SQL `LIKE` operator relies on a restricted form of regular expression matching, and requires the text being matched to be from a single script, making it unsuitable in multilingual (or semantic) world.

Graph Database Systems
While customized graph database platforms have been developed for specific application domains [14, 21], they are yet to be adopted as general purpose solutions. Our focus is to define a general semantic operator and profile and optimize its performance in the relational systems.

Information Retrieval Research
There are vast amounts of literature in the Information Research community in the areas of Knowledge-based and Crosslingual information retrieval. The techniques employed are varied, ranging from syntactic and morphological analysis [9], Machine Translation [5], statistical techniques [10], and Latent Semantic Indexing [4] for semantic querying in a single language. Further, paired dictionaries, thesauri and statistical mapping techniques are used for handling cross-language querying. We refer to the Multilingual Information Retrieval Track of the ACM SIGIR conference for a survey of current techniques. However, this research focuses on specialized NLP techniques to work on a large corpora of text and is not well suited for attribute granularity data. The closest technique to ours uses paired thesauri [25], but does not provide generalization techniques. Though WordNet based approach was used for semantic information retrieval in [24], and for crosslingual information retrieval in [11], the emphasis of these papers has been on *quality* of the results and not performance. In contrast, we focus on query processing on relational systems in an OLTP environment.

Semantic Web
Relating to the semantic processing of data, the Semantic Web [26] proposed and promoted by W3C Consortium, extends the current web data by annotating it with semantic metadata information. Such annotations are more appropriate for web domain, and not for database query processing. However, the ontological hierarchies captured in such a formalism may be used in our methodology.

6. CONCLUSIONS

Given that the global deployment of *e-Commerce* applications and tools need support for seamless multilingual text data processing based on their semantics, we proposed a new SQL operator – SemEQUAL, intended for matching multilingual text attribute data based on their meanings. Our proposal outlines a light-weight, yet robust approach, for implementing this feature by adopting and integrating the WordNet linguistic resource in the database system. Leveraging the rich taxonomic hierarchies in WordNets and cross-linking between them, multilingual text attribute data may be matched, by transforming them to a canonical semantic form. As a side effect, such a methodology provides a repeatable and consistent results set for a given data set across different database systems.

We outlined a *derived-operator* implementation for SemEQUAL. Our experiments with WordNet on three commercial database systems, confirmed the utility of the SemEQUAL operator, but underscored the inefficiencies in computing transitive closure, an essential component for semantic matching. The runtimes are in the order of tens of seconds, unsuitable for any practical deployments. By tuning the storage and access structures to match the characteristics of resources in the linguistic domain, we speeded up the closure computation by 2 to 3 orders of magnitude – to *a few milliseconds* – making the operator viable for supporting user online query processing. Thus, we show the viability of such an operator to support semantic matching. In the long-term, we expect a *core-operator* implementation would exhibit even further performance improvements.

Acknowledgement We thank Dr. Pushpak Bhattacharyya, Professor and the Coordinator of Center for Indian Language Technology at IIT-Bombay, for providing us with details and data on Hindi WordNet.

7. REFERENCES

[1] R. Agrawal, S. Dar and H. V. Jagadish. Direct Transitive Closure Algorithms: Design and Performance Evaluation. *ACM Trans. on Database Systems*, 1990.

[2] The British National Corpus, Oxford University Press. *http://www.comp.lancs.ac.uk/*.

[3] Centre for Indian Language Technology, IIT-Bombay. *http://www.cfilt.iitb.ac.in*.

[4] S. Deerwester, S. T. Dumais and W. C. Ogden. Indexing by Latent Semantic Analysis. *Jour. of American Soc. of Information Sciences*, September 1990.

[5] The EuroSpider. *http://www.eurospider.ch/*.

[6] The Euro-WordNet. *http://www.illc.uva.nl/EuroWordNet/.*

[7] The Euro-WordNet – Final Results Report. *http://www.illc.uva.nl/EWN/finalresults-ewn.html.*

[8] C. Fellbaum and G. A. Miller. WordNet: An electronic lexical database (language, speech and communication). *MIT Press*, 1998.

[9] C. Fluhr *et al.* Multilingual Database and Crosslingual Interrogation in a Real Internet Application. *AAAI Sym. on Crosslanguage Text and Speech Retrieval*, 1997.

[10] F. Gey, A. Chen, M. Buckland and R. Larson. Translingual Vocabulary Mapping for Multilingual Information Access. *Proc. of 25th ACM SIGIR Conf.*, 2002.

[11] J. Gilarranz, J. Gonzalo and F. Verdejo. An Approach to Conceptual Text Retrieval using the Euro-WordNet Multilingual Semantic Database. *Proc. of AAAI Conf. on Crosslanguage Text and Speech Retrieval*, 1997.

[12] The Global WordNet Association. *http://www.globalwordnet.org/.*

[13] L. Gravano *et al.* Approximate String Joins in a Database (almost) for Free. *Proc. of the 27th VLDB Conference, Rome, Italy*, 2001.

[14] M. Graves, E. R. Bergeman and C. B. Lawrence. Graph Database Systems. *IEEE Engineering in Medicine and Biology Magazine*, December 1995.

[15] J. Han et al. Some Performance Results on Recursive Query Processing in Relational Database Systems. *Proc. of 2nd ICDE Conf.*, 1986.

[16] Y. Ioannidis. On the Computation of TC of Relational Operators. *Proc. of 12th VLDB Conf.*, 1986.

[17] B. D. Jayaram and P. Bhattacharyya. Report on Indo-WordNet Workshop. *Central Institute of Indian Languages*, January 1999.

[18] A. Kumaran and J. R. Haritsa. On Multilingual Performance of Database Systems. *Proc. of 29th VLDB Conf.*, 2003.

[19] A. Kumaran and J. R. Haritsa. Supporting Multiscript Matching in Database Systems. *Prof. of 9th EDBT Conf.*, 2004.

[20] A. Kumaran and J. R. Haritsa. LexEQUAL: Multilexical Matching Operator in SQL. *Proc. of 23rd ACM SIGMOD Intl. Conf. on Mgmt. of Data*, 2004.

[21] H. S. Kunii. DBMS with Graph Datamodel for Knowledge Handling. *Proc. of the Joint Comp. Conf. on Exploring technology: today and tomorrow*, 1987.

[22] M. Liberman and K. Church. Text Analysis and Word Pronunciation in TTS Synthesis. *Advances in Speech Processing*, 1992.

[23] The Computer Scope Ltd. *http://www.NUA.ie/Surveys.*

[24] R. Richardson and A. F. Smeaton. Using WordNet in a Knowledge-based Approach to Information Retrieval. *Working Paper CA-0395, Dublin City University*, 1999.

[25] D. Soergel. Multilingual thesauri in cross-language text and speech retrieval. *AAAI Sym. on Cross-Language Text and Speech Retrieval*, March 1997.

[26] The Semantic Web. *http://www.w3.org/2001/sw.*

[27] The Unicode Consortium. The Unicode Standard. *Addison-Wesley*, 2000.

[28] P. Vossen. EuroWordNet: Final Report. *University of Amsterdam*, October, 1999.

[29] The WebFountain. *http://www.almaden.ibm.com/WebFountain.*

[30] The WordNet. *http://www.cogsci.princeton.edu/w̃n.*

[31] Word Discover. *http://www.worddiscover.com.*

[32] The W3C's Ontology Language. *http://www.w3c.org/TR/owl-ref/.*

8. APPENDIX – A: WORDNET

WordNet [30] is a linguistic reference system, organized based on the meanings and semantic relationships. We provide a brief overview in this section and refer the interested reader to [8, 30] for further details.

8.1 Word Form and Word Sense

A word may be thought of as a lexicalized concept; simply, it is the written form of a mental concept that may be an object, action, description, relationship, etc. Formally, it is referred to as a *Word-form*. The concept that it stands for is referred to as *Word-sense*, or in WordNet parlance, *Synset*. The defining philosophy in the design of WordNet is that a synset is sufficient to identify a concept for the user. A short description, similar to the dictionary meaning, called the *Gloss* is provided with synsets, for human understanding. Two words are said to be synonymous, *or semantically the same*, if they have the same synset and hence map to the same concept. WordNet organizes all relationships between the concepts of a language as a semantic network between synsets. A lexical matrix that maps word forms to word senses forms the basis for mapping a word-form to a synset. For example, the word-form `mouse` corresponds to several different synsets, two of which are {*rodent, a gnawing animal*} and {*computer peripheral, an electronic device*}.

The synsets are divided into five distinct categories – *nouns*, *verbs*, *adjectives*, *adverbs* and *functional words*, with each of the groups giving rise to different types of semantic relationships between the synsets. We explore below the Nouns category, as *about a fifth of normal text corpora and majority of query strings are nouni-form words* [22].

8.2 Nouns

The nouns in English WordNet are grouped under approximately twenty-five distinct *Semantic Primes* [8], covering distinct conceptual domains, such as *Animal*, *Artifact*, etc. Under each of the semantic primes, the nouns are organized in a taxonomic hierarchy, as shown in Figure 11, with *Hyponyms* links signifying the `is-a` relationships and the reverse *Hypernyms* signifying the `has-sub-class` relationships. The `is-a` relationships define a *semantic taxonomic hierarchy* that is leveraged for matching on semantics, in our strategy. Additionally, a `part-of` hierarchy is also interwoven into the above taxonomic hierarchy. The specializations of a given noun are the synsets that occur in the transitive closure – that is, those nodes reachable in the taxonomic hierarchy from the node corresponding to the synset of the given noun.

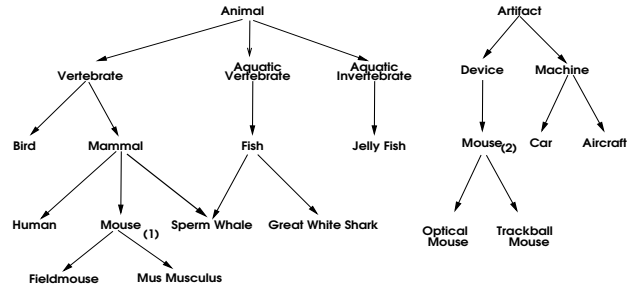

Figure 11: Sample *WordNet* Noun Hierarchy