SemMemDB: In-Database Knowledge Activation

Yang Chen
University of Florida
Gainesville, FL 32611, USA
yang@cise.ufl.edu

Milenko Petrovic and **Micah H. Clark**
Florida Institute for Human & Machine Cognition
Ocala, FL 34471, USA
{mpetrovic, mclark} @ihmc.us

Abstract

Semantic networks are a popular way of simulating human memory in ACT-R-like cognitive architectures. However, existing implementations fall short in their ability to efficiently work with very large networks required for full-scale simulations of human memories. In this paper, we present SemMemDB, an in-database realization of semantic networks and spreading activation. We describe a relational representation for semantic networks and an efficient SQL-based spreading activation algorithm. We provide a simple interface for users to invoke retrieval queries. The key benefits of our approach are: (1) Databases have mature query engines and optimizers that generate efficient query plans for memory activation and retrieval; (2) Databases can provide massive storage capacity to potentially support human-scale memories; (3) Spreading activation is implemented in SQL, a widely-used query language for big data analytics. We evaluate SemMemDB in a comprehensive experimental study using DBPedia, a web-scale ontology constructed from the Wikipedia corpus. The results show that our system runs over 500 times faster than previous works.

Introduction

This paper introduces SemMemDB, a module for efficient in-database computation of spreading activation over semantic networks. Semantic networks are broadly applicable to associative information retrieval tasks (Crestani 1997), though we are principally motivated by the popularity of semantic networks and spreading activation to simulate aspects of human memory in cognitive architectures, specifically ACT-R (Anderson et al. 2004; Anderson 2007). Insofar as cognitive architectures aim toward codification of unified theories of cognition and full-scale simulation of artificial humans, they must ultimately support human-scale memories, which at present they do not. We are also motivated by the desire for a scalable, standalone, cognitive model of human memory free from the architectural and theoretical commitments of a complete cognitive architecture.

Our position is that human-scale associative memory can be best achieved by leveraging the extensive investments and continuing advancements in structured databases and big data systems. For example, relational databases already provide effective means to manage and query massive structured data and their commonly supported operations, such as grouping and aggregation, are sufficient and well-suited for efficient implementation of spreading activation. To defend this position, we design a relational data model for semantic networks and an efficient SQL-based, in-database implementation of network activation (i.e., SemMemDB).

The main benefits of SemMemDB and our in-database approach are: 1) Exploits query optimizer and execution engines that dynamically generate efficient execution plans for activation and retrieval queries, which is far better than manually implementing a particular fixed algorithm. 2) Uses database technology for both storage and computation, thus avoiding the complexity and communication overhead incurred by employing separate modules for storage versus computation. 3) Implements spreading activation in SQL, a widely-used query language for big data which is supported by various analytics frameworks, including traditional databases (e.g., PostgreSQL), massive parallel processing (MPP) databases (e.g., Greenplum (EMC 2010)), the MapReduce stack (e.g., Hive), etc.

In summary, this paper makes the following contributions:

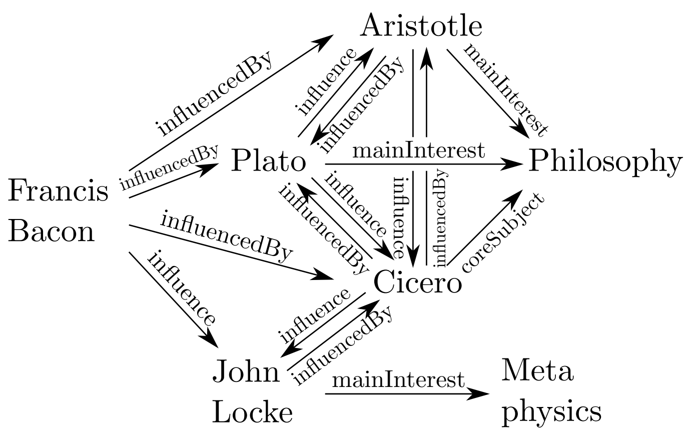
- A relational model for semantic networks and an efficient, scalable SQL-based spreading activation algorithm.
- A comprehensive evaluation using DBPedia showing orders of magnitude speed-up over previous works.

In the remainder of this paper: (i) we provide preliminaries explaining semantic networks and activation; (ii) we discuss related work regarding semantic networks and ACT-R's associative memory system; (iii) we describe the implementation of SemMemDB; (iv) we evaluate SemMemDB using DBPedia (Auer et al. 2007), a web-scale ontology constructed from the Wikipedia corpus. Our experiment results show several orders of magnitude of improvement in execution time in comparison to results reported in the related work.

Preliminaries

Semantic memory refers to the subcomponent of human memory that is responsible for the acquisition, representation, and processing of conceptual information (Saumier and Chertkow 2002). Various representation models for semantic memory have been proposed; in this paper, we use the *semantic network* model (Sowa 2006). A semantic network

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a)

N	i	p	j
	FB	IB	A
	FB	IB	P
	FB	IB	C
	FB	I	JL
	A	MI	Ph
	P	MI	Ph
	C	CS	Ph
	JL	MI	M
		···	

(b)

H	i	t
	FB	0
	A	0
	P	0
	C	0
	Ph	0
	M	0
	FB	1
	FB	3
	P	3
	A	6

(c)

Q_1	i	w
	FB	1.0

(d)

`Ans`$_1$	i	s
	A	0.4161
	P	0.2941
	C	-0.0610
	JL	-0.0610

(e)

Q_2	i	w
	FB	1.0
	A	0.4161
	P	0.2941
	C	-0.0610
	JL	-0.0610

(f)

`Ans`$_2$	i	s
	A	0.5728
	P	0.5256
	C	0.3199
	Ph	-0.0395
	JL	-0.0848
	M	-0.5065

(g)

Figure 1: (a) Semantic network fragment showing the relationships between scientists and their interests. Each node represents a DBPedia entity; each directed edge represents a relationship between the entities. (b) Database table N that stores the network depicted in (a). Each row (i, p, j) represents a directed edge between (i, j) with predicate p. We use abbreviations here for illustration (e.g., "FB" for "Francis Bacon"). (c) History table H recording the presentation history for each node. Zero means creation time. (d) and (f) are two example queries; (e) and (g) are the results of (d) and (f), respectively, ranked by activation scores. The precise definitions of N, H, Q tables are given in the Using SemMemDB Section.

consists of a set of nodes representing entities and a set of directed edges representing relationships between the entities. Figure 1(a) shows an example semantic network. It is constructed from a small fragment of DBPedia, an ontology extracted from Wikipedia. In this example, we show several scientists and their research topics. The edges in this network indicate how the scientists influence each other and their main interests.

Processing in a semantic network takes the form of spreading activation (Collins and Loftus 1975). Given a set of source (query) nodes Q with weights, the spreading activation algorithm retrieves the top-K most relevant nodes to Q. For example, to retrieve the most relevant nodes to "Francis Bacon," we set Q={(Francis Bacon, 1.0)} as shown in Figure 1(d). The algorithm returns {Aristotle, Plato, Cicero, John Locke} ranked by their activation scores as shown in Figure 1(e). These activation scores measure relevance to the query node(s) and are explained shortly. Figure 1(f) shows another example query, a second iteration of the previous query formed by merging the original query with its result. Figure 1(g) shows the result of this second iteration.

As shown in the above examples, the spreading activation algorithm assigns an *activation* score to each result node measuring its relevance to the query nodes in Q. The activation score A_i of a node i is related to the node's history and its associations with other nodes. It is defined as

$$A_i = B_i + S_i. \tag{1}$$

The B_i and S_i terms are *base-level activation* and *spreading activation*, respectively. The base-level activation term reflects recency and frequency of use while the spreading activation term reflects relevance to the current context or query. Formally, the base-level activation B_i is defined as

$$B_i = \ln\left(\sum_{k=1}^{n} t_k^{-d}\right), \tag{2}$$

where t_k is the time since the kth presentation of the node and d is a constant rate of activation decay. (In the ACT-R community, $d = 0.5$ is typical.) In the case of DBPedia for example, t_k values might be derived from the retrieval times of Wikipedia pages. The resultant B_i value predicts the need to retrieve node i based on its presentation history. The spreading activation S_i is defined as

$$S_i = \sum_{j \in Q} W_j S_{ji}. \tag{3}$$

W_j is the weight of source (query) node j; if weights are not specified in a query then a default value of $1/n$ is used, where n is the total number of source nodes. S_{ji} is the strength of association from node j to node i. It is set to $S_{ji} = S - \ln(\text{fan}_{ji})$, where S is a constant parameter and

$$\text{fan}_{ji} = \frac{1 + \text{outedges}_j}{\text{edges}_{ji}}.$$

The values of outedges_j and edges_{ji} are the number of edges from node j and the number of edges from node j to node i, respectively.

Equations (2) and (3) are easily implemented using the native grouping and aggregation operations supported by relational database systems. Thus, in only a few lines of SQL we are able to implement the relatively complex task of spreading activation. Moreover, database systems are able to generate very efficient query plans based on the table statistics, query optimization, etc. Thus, we believe it is better to rely on the databases to dynamically select the most efficient algorithm rather than to manually develop a fixed one as is done in previous works.

Related Work

Unsurprisingly, ACT-R's (Anderson, Matessa, and Lebiere 1997) original Lisp-based implementation of spreading activation in semantic networks does not scale to large associative memories. Researchers have thus investigated various ways to augment ACT-R's memory subsystem to achieve scalability. These investigations include outsourcing the storage of associative memories to database management systems (Douglass, Ball, and Rodgers 2009) and concurrently computing activations using Erlang (Douglass and Myers 2010). We observe that in (Douglass, Ball, and Rodgers 2009), databases are used only as a storage medium; activation computations are performed serially outside of the database, which is unnecessarily inefficient and incurs the significant communication overhead of data transfer in and out of the database. In SemMemDB, we try to leverage the full computational power of databases by performing all activation calculations within the database itself, using a SQL-based implementation of the spreading activation algorithm.

Semantic network spreading activation has also been explored using Hadoop and the MapReduce paradigm (Lorenzo, Gayo, and Rodríguez 2013). However, MapReduce-based solutions are batch oriented and not generally appropriate for dealing with ad hoc queries. In terms of simulating an agent's memory (our principle motivating use-case), queries against the semantic network are ad hoc and real-time, which are the types of queries better managed by relational database systems.

Using SemMemDB

We refer to Figure 1 to illustrate how users are expected to interact with the SemMemDB module. Specifically, a user defines a *query table* Q, a *network table* N, and a *history table* H. Having done so, activation and retrieval of the top-K nodes is initiated by a simple SQL query:

```
SELECT * FROM activate()
ORDER BY A DESC LIMIT K;
```

The `activate` function is a database stored procedure. Its parameters are implicitly Q, N, H, so it is to be understood as `activate(Q,N,H)`. The `ORDER BY` and `LIMIT` components are optional; they instruct the database to rank nodes by their activation scores and to return only the top-K results. The result is a table of at most K activated nodes with, and ranked by, their activation scores.

Tables Q, N, H are defined by the following:

- Table Q contains a tuple (i, w) for each query node i with numeric weight w.
- Table N contains a tuple (i, p, j) for each directed edge from node i to node j with predicate p.
- Table H contains a tuple (i, t) for every node i presented at numeric time t. A node was 'presented' if it was created, queried, or a query result at time t (users may choose other criteria). The specific measure of time (e.g., time-stamp, logical time) is defined by the user. H is typically updated (automatically) when a query is returned by insertion of the query and result nodes with the current time.

Tables Q, N, and H are allowed to be *views* defined as queries over other tables, so long as they conform to the described schema. This allows users to specify more complex data models and queries using the same syntax.

Example. *For the semantic network shown in Figure 1(a), the corresponding network table N is shown in Figure 1(b). Figure 1(c) shows one possible history table H; node creation time is assumed to be 0. Figures 1(d) and 1(f) correspond to the queries for* {*"Francis Bacon"*}, *and* {*"Francis Bacon", "Aristotle", "Plato", "Cicero", "John Locke"*}, *respectively, with the weights listed in the w columns. Finally, Figures 1(e) and 1(g) show the results of those queries.*

Base-Level Activation Calculation

The base-level activation B_i defined by (2) corresponds to a grouping and summation operation. Assuming the current time is T, the following SQL query computes the base-level activations of all nodes:

```
SELECT i, log(SUM(power(T-t,-d))) AS b
FROM H
GROUP BY i;
```

In Figure 1, "Aristotle" was presented most recently at time 6, next "Plato" at time 3, while "Cicero" and "John Locke" have not presented. In response to Q_1, "Aristotle" is judged most relevant to "Francis Bacon" (see Figure 1(e)) despite the fact that all of these nodes have the same number of edges (viz., 1) connecting them to "Francis Bacon." This is because of the differences between their base-level activations.

Spreading Activation Calculation

The spreading activation S_i defined by (3) is decomposed into two components, W_j, which is query dependent, and S_{ji}, which is network dependent but query independent. Since S_{ji} is query independent, an effective way to speed

Listing 1: Activation Procedure

```
1  CREATE OR REPLACE FUNCTION activate()
2  RETURNS TABLE(node INT, s DOUBLE
       PRECISION) AS $$
3  BEGIN
4    RETURN QUERY
5    WITH Spreading AS (
6      SELECT Assoc.j AS i,
           SUM(Q.w*Assoc.l) AS s
7      FROM Q NATURAL JOIN Assoc
8      GROUP BY Assoc.j
9    ), Base AS (
10     SELECT H.i AS i,
           log(SUM(power(T-t,-d))) as b
11     FROM H NATURAL JOIN Spreading
12     GROUP BY H.i
13   )
14   SELECT Base.i AS i, Base.b+Spreading.s
         AS A
15   FROM Base NATURAL JOIN Spreading;
16 END;
17 $$ LANGUAGE plpgsql;
```

up calculation is to precompute S_{ji} values in *materialized views*. These views store precomputed results in intermediate tables so that they are available during query execution. First, we compute the number of edges from each node i:

```
CREATE MATERIALIZED VIEW OutEdges AS
SELECT i, COUNT(*) AS l FROM N
GROUP BY i;
```

Then, we compute the actual S_{ij} values:

```
CREATE MATERIALIZED VIEW Assoc AS
SELECT i, j, S-ln((1+OutEdges.l)/COUNT(*))
    AS l
FROM N NATURAL JOIN OutEdges
GROUP BY (i, j, OutEdges.l);
```

Though a fair amount of computation happens here, we emphasize that it is done *once*, thereafter the resultant values are used by all queries against the semantic network.

Given the above definition of `Assoc` and a query Q, we compute the spreading activation S_i as follows:

```
SELECT j AS i, SUM(Q.w*Assoc.l) AS s
FROM Q NATURAL JOIN Assoc
GROUP BY j;
```

Activation Score Calculation

The complete SQL procedure for computing activation scores is given in Listing 1. In the `activate()` procedure, we start by computing the S_i terms in the `WITH Spreading AS` clause. The result of this subquery (`Spreading`) is often small so that history look-up can be optimized by joining H and `Spreading` in Line 11. In this way, only the relevant portion of history is retrieved. The final activation scores A_i are computed by joining `Base` and `Spreading` and adding up the corresponding B_i and H_i terms in Lines 14-15.

Evaluation

In this section, we evaluate the performance of SemMemDB using PostgreSQL 9.2, an open-source database management system. We run all the experiments on a two-core machine with 8GB RAM running Ubuntu Linux 12.04.

Data Set

We use the English DBPedia[1] ontology as the data set for evaluation. The DBPedia ontology contains entities, object properties, and data properties. Entities and object properties correspond to nodes and edges in the semantic network. Data properties only associate with single nodes, so they do not affect spreading activation and hereafter are ignored. We generate a pseudo-history for every node in the semantic network based on the assumption that 'past retrievals' follow a Poisson process, where the rate parameter of a node is determined by the number incoming edges.[2] The statistics of our DBPedia semantic network data set are listed in Table 1.

# nodes (entities)	3,933,174
# edges (object properties)	13,842,295
# histories (pseudo-data)	7,869,462

Table 1: DBPedia data set statistics.

For comparison, Table 2 lists the statistics of the Moby Thesaurus II data set, which Douglass and Myers (2010) used to evaluate their semantic network implementation.

# nodes (root words)	30,260
# edges (synonyms)	2,520,264

Table 2: Moby Thesaurus II data set statistics.

Performance Overview

In the first experiment, we run three queries against the entire DBPedia semantic network data set. The initial queries are listed in Table 3(a) where each contains three nodes. For each, we execute three iterations where the query for each iteration is based on the result of the previous iteration starting with the initial query. We measure the execution time by executing each iteration ten times and taking the average. The execution times and result sizes are listed in Table 3(b). Note that the history table is not modified during experiments.

All the queries complete within tens of milliseconds. We informally compare this result to those in (Douglass and Myers 2010). Douglass and Myers evaluate their semantic network implementation using the Moby Thesaurus II data set, which is only a tenth of the size of the DBPedia data set (see Table 2). Their average execution time is 10.9 seconds. This is more than 500 times slower than SemMemDB using for comparison Q_1, Iter. 2 at 22.02 ms, which has a larger fan than any query used in (Douglass and Myers 2010). Though

[1] `http://wiki.dbpedia.org/Downloads39`

[2] We assume that nodes with greater connectivity are retrieved more often, but this choice is arbitrary.

Query / Node i	Q_1	Q_2	Q_3
1	Aristotle	United States	Google
2	Plato	Canada	Apple
3	John Locke	Japan	Facebook

(a)

	Query	Iter. 1	Iter. 2	iter. 3
Q_1	time/ms	5.16	22.02	63.22
	result size	125	890	3981
Q_2	time/ms	1.77	5.69	14.60
	result size	23	121	477
Q_3	time/ms	2.58	6.15	13.61
	result size	36	132	381

(b)

Table 3: (a) Experiment 1 initial queries. (b) Avg. execution times and result sizes for queries by iteration.

Proportion	20%	25%	33%	50%	100%
# nodes	2,607,952	2,846,850	3,130,706	2,012,183	3,933,174
# edges	2,768,119	3,463,240	4,616,433	6,035,162	13,842,295
# histories	1,988,400	2,345,531	2,903,576	3,906,121	7,869,462
time/ms	35.05	38.47	42.52	45.02	57.63

Table 4: Experiment 2 semantic network sizes and avg. execution times for single iteration queries of 1000 nodes.

informal, this result illustrates the performance benefits offered by the in-database architecture of SemMemDB.

Effect of Semantic Network Sizes

In the second experiment, we evaluate the scalability of SemMemDB by executing queries against semantic networks of increasing size. These semantic networks are produced by using 20%, 25%, 33%, 50%, and 100% of the DBPedia data set. Query size is fixed at 1000 nodes and queries are generated by taking random subsets of DBPedia entities. The execution times and network sizes are listed in Table 4.

It is perhaps a surprising yet highly desirable result that execution time grows more slowly than network size. This scalability is due to the high *selectivity* of the join queries. Since the query size is much smaller than the network size, the database is able to efficiently select query-relevant tuples using indexes on the join columns (see Figure 2(b), left plan). As a result, execution time is *not* much affected by the total size of the semantic network, only the retrieved sub-network and the index size matter.

Effect of Query Sizes

In the third experiment, we evaluate the scalability of SemMemDB with respect to query size. We execute queries ranging in size from 1 to 10^4 nodes against the entire DBPedia semantic network data set. The queries are generated by taking random subsets of DBPedia entities. The execution times, query sizes, and result sizes are listed in Table 5.

The results indicate that execution time scales linearly with query size when query size is small ($\leq 10^3$). Under these conditions, join selectivity is high and indexing accelerates query-relevant tuple retrieval. This effect is illustrated in the query plans shown in Figure 2(b) for the sample query in Figure 2(a). When the query size is small, the index scan on `Assoc` only retrieves a small number of nodes, hence an

Query Size	1	10	10^2	10^3	10^4
time/ms	1.33	2.45	11.27	57.63	2922.51
result size	4	30	321	2428	19,007

Table 5: Experiment 3 avg. execution times and result sizes for single iteration queries of varying sizes.

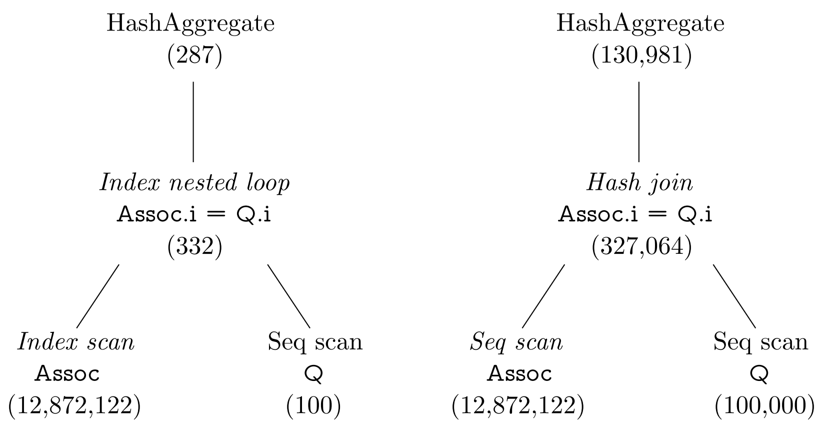
index nested loop is chosen by the database query planner. When the query size is large (e.g., 10^4), a large portion of the `Assoc` needs to be accessed (the index is not of much help), so the database chooses a hash join for better performance. This dynamic selection of the 'best' algorithm exemplifies why we feel it is better to rely on the database to efficiently plan and execute activation queries rather than to manually implement a fixed algorithm.

Conclusion

In this paper, we introduced SemMemDB, a module for efficient in-database computation of spreading activation over semantic networks. We presented its relational data model and its scalable spreading activation algorithm. SemMemDB is applicable in many areas including cognitive architectures and information retrieval, however our presentation was tailored to those seeking a scalable, standalone cognitive model of human memory. We evaluated SemMemDB on the English DBPedia data set, a web-scale ontology constructed from the Wikipedia corpus. The experiment results show more than 500 times performance improvement over previous implementations of ad hoc spreading activation retrieval queries over semantic networks. What we have reported here is an early stage development toward a scalable simulation of human memory. Planned future work includes the use of MPP databases and support for more complex queries and models as described in (Bothell 2007).

```
SELECT j AS i,
    SUM(Q.w*Assoc.l) AS A
FROM Q NATURAL JOIN Assoc
GROUP BY j;
```

(a)



(b)

Figure 2: (a) Sample query. (b) Query plans for (a) given that Q is small (left) or large (right). Numbers in parentheses indicate table sizes. Italics indicate differences between the two plans.

Acknowledgments

This work was sponsored by the Department of the Navy, Office of Naval Research under award no. N00014-13-1-0225, and by DARPA under contract no. FA8750-12-2-0348-2.

We thank Dr. Daisy Zhe Wang and Dr. Tom Eskridge along with four anonymous reviewers for their insightful comments on this manuscript.

References

Anderson, J. R.; Bothell, D.; Byrne, M. D.; Douglass, S.; Lebiere, C.; and Qin, Y. 2004. An integrated theory of the mind. *Psychological Review* 111(4):1036.

Anderson, J. R.; Matessa, M.; and Lebiere, C. 1997. ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction* 12(4):439–462.

Anderson, J. R. 2007. *How can the human mind occur in the physical universe?* Oxford University Press.

Auer, S.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; and Ives, Z. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*. Springer. 722–735.

Bothell, D. 2007. *ACT-R 6.0 Reference Manual*.

Collins, A. M., and Loftus, E. F. 1975. A spreading-activation theory of semantic processing. *Psychological Review* 82(6):407.

Crestani, F. 1997. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review* 11(6):453–482.

Douglass, S. A., and Myers, C. W. 2010. Concurrent knowledge activation calculation in large declarative memories. In *Proceedings of the 10th International Conference on Cognitive Modeling*, 55–60.

Douglass, S.; Ball, J.; and Rodgers, S. 2009. Large declarative memories in ACT-R. In *Proceedings of the 9th International Conference of Cognitive Modeling*.

EMC. 2010. Greenplum Database: Critical Mass Innovation. Technical report, EMC.

Lorenzo, J. G.; Gayo, J. E. L.; and Rodríguez, J. M. Á. 2013. Applying MapReduce to Spreading Activation Algorithm on Large RDF Graphs. In *Information Systems, E-learning, and Knowledge Management Research*. Springer. 601–611.

Saumier, D., and Chertkow, H. 2002. Semantic Memory. *Current Neurology and Neuroscience Reports* 2(6):516–522.

Sowa, J. F. 2006. Semantic Networks. In *Encyclopedia of Cognitive Science*. John Wiley & Sons, Ltd.