HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Semantic Search

An introduction to Semantic Search in Microsoft SQL Server 2012

Seminar Thesis

Master of Science in Engineering
Major Software and Systems
HSR Hochschule für Technik Rapperswil
www.hsr.ch/mse

Advisor: **Prof. Hansjörg Huser**
Author: **Rico Suter**

Rapperswil, January 2013

Version 1.0

Abstract

The purpose of this paper is to deliver a complete guide on how to use the semantic search feature in Microsoft SQL Server 2012 effectively. After reading the paper the reader should know how to setup the semantic search feature, use it and – most importantly – knows how it works under the hood which is required to effectively use it. The paper starts by introducing the reader into the world of semantic search. After this introduction the rest of the paper only discusses Microsoft SQL Server 2012's semantic search feature. Beside some detailed guides on how to install and use the database there is also an analysis section which will introduce the reader into some insights by explaining the underlying mechanisms. This is useful to demonstrate the database's strengths, weaknesses and limitations. The paper ends with notes about when to use the Microsoft SQL Server 2012 semantic search feature, a conclusion and the writer's opinion.

Keywords

Semantic search, Mircosoft SQL Server 2012, SQL, Queries, Analysis

Contents

1 Introduction to semantic search

1.1 What is semantic search

Often, traditional full-text search does not provide good enough results for an average user. This happens mostly with queries which cannot simply searched by finding an exact match in texts. Often a search engine has to "understand" a query. This is where semantic search comes into play. Semantic search tries to improve search by understanding the contextual meaning of the terms and tries to provide the most accurate answer from a given document repository. The technologies behind semantic search are mostly used to access unstructured data. Most users use semantic search in modern search engines like Google's well known search engine. Web site search engines are a good sample of accessing unstructured data because common HTML pages do not have a lot of structuring information as they are formatted text with links to other articles. Looking at structured data there are semantic databases which store objects or concepts with their dependencies. An example is freebase [1] a semantic database of everything which can be accessed via MQL (Metaweb Query Language), a JSON based query language. However this paper will give you an introduction to Microsoft SQL Server 2012 semantic search which is used to access unstructured data.

1.2 Products

As mentioned before, most semantic search engines cannot be bought nor installed. They can be found on popular websites like the search page from Google, Wolfram Alpha and many more. However there are some standalone applications. This chapter outlines some of the available products, however the rest of the paper will only cover semantic search with Microsoft SQL Server 2012.

Online search engines

There are several search engines with semantic search features. The following list contains some interesting search engines:

- **Google**'s search engine [2]: Well-known search engine which uses statistical frequencies to query unstructured data contained in web sites.
- **Wolfram Alpha** [3]: A search engine which is mainly used for statistical and mathematical queries. The search engine has been advertised as using artificial intelligence to answer to users questions.
- **Freebase** [1]: Database with extremely structured data to search in. The database is a collaborative metadata repository which is gathered from various sources like Wikipedia contributions.
- **Hakia** and **Powerset**: These two companies try to implement a search engine like freebase but based on the unstructured data found on web sites.

Apache Lucene / Solr

Apache Lucene is an open source search engine written in the Java programming language. There are several plugins which can be used to extend Lucene to support semantic search. Apache Solr is an Apache Lucene sub-project with more advanced features like extended full-text search, faceted search or support for more document formats like PDF or Microsoft Word files.

Microsoft SQL Server 2012

Microsoft SQL Server 2012 is a proprietary relational database management system released in March 2012. It has been written in C++ and runs on x86 or x64 platforms. The semantic search feature has been added in the version 2012 as an extension to full-text search which was available in previous versions.

1.3 How does semantic search work

There are several techniques to implement semantic search [4]. It is evident to distinguish between structured and unstructured data. Unstructured data is usually a text which does not contain any machine readable semantic information. The best search or indexing method depends on the structure of your data.

One type of structured data are ontologies. An **ontology** formally describes available concepts in a specific domain. It describes relations between words like “dogs are enemies of cats” or “dog is an animal”. With this information it is possible for a search engine to “understand” a query by travelling an available ontology.

One possibility to query structured data is **conceptual graph matching** [5]. In this method each query and the data are represented as trees of concepts (ontologies). The search engine compares the query with each tree in the database and finds the best matching tree.

One way to represent such a data structure is by using **RDF**. RDF (Resource Description Framework) is a framework used to describe data. It can be used to describe data structures much like a domain class model. Using this net of connecting concepts, it is possible that the search engine understands the search context and can retrieve more accurate search results. Using this conceptual network it is also possible to do **word sense disambiguation** [6]. If the user searches for a word with multiple meanings, the search engine chooses the most probable meaning by examining the other words in the query and the available concepts.

The problem of the previous mentioned techniques is that they require structured data – they are not suitable to access unstructured data like text documents. In a lot of common scenarios, there are a lot of text documents like web pages, word files or text columns in a database. Because Microsoft SQL Server does not have the metadata information, we will no further look into techniques which require them.

To query unstructured data, the search engine has to index all documents, split it up into keywords and score them depending on statistical analysis. Most search engines index all the documents with a **term extraction** or **a phrase extraction** algorithm. Phrase extractions will bundle multiple words whereas a term extraction algorithm simply divides the text into single word chunks.

The next step is to remove noise words. This are words which are too common and which do not contain usable information. Some example words are “in” or “the”. These words are also called **stop words**. Usually a search engine has a stop word database per language. Some search engines do not remove stop words to better support exact phrase search.

Another important thing in semantic search is **inflection** and **stemming**. Inflection is the modification of a word to another form – for example another tense, case, etc. Transforming a word back to its stem is called **stemming**. Using word stemming the search engine only uses the stems of the words and therefore can match words even if the user searches with words in the wrong form. There is for example past-tense stemming which is used to help match verbs in various conjugations.

A search engine should also support **synonyms** and **replacements**. The synonymic noun database is used to match nouns with the same meaning whereas replacements are used to match abbreviations or wrong spelled words.

Using a **statistical database** every keyword will get a score depending on its relevance in the language. The scores are precalculated from a big pool of random texts like books, magazines, web pages, etc. Using this database words with high appearances in a language are scored lower than rare words.

Microsoft SQL Server 2012

Microsoft SQL Server 2012 in the current version supports only unigrams. A unigram is a single word, whereas n-grams are word combinations (also known as term and phrase extraction respectively). Therefore words like “Internet Explorer” are not supported.

The server creates two internal indexes for semantic search columns or file tables. The **tag index** uses a variation of the TF-IDF (term frequency - inverse document frequency) [4] algorithm to determine the importance of the keywords. The TF-IDF value is proportionally based on the number of times a word appears in the text and inverse proportionally on the global occurrence of the word (retrieved from the statistical database). This is to minimize the significance of common words like the word “the”.

The second index is the **document similarity index** which uses a variation of the cosine similarity algorithm [5]. The cosine similarity algorithm determines the angle between two vectors. In the domain of semantic search each document has a vector with a vector component for each keyword and its significance as the vector length. More similar documents will have a smaller angle between their keyword vectors.

The extracted key phrases are not stored as ontology on the database engine. But they are stored rationally and are accessible through the table-valued functions which are discussed later in this paper.

2 Usage

This chapter explains how to install the server with semantic search. In the second part, we will setup a semantic search index over a column and on files from a file table. The last part of the chapter shows some queries which can be done using the created semantic search indexes.

2.1 Installation

First we have to install the Microsoft SQL Server 2012 with semantic search capabilities. The setup needs three steps:

1. Installing the server with semantic search feature
2. Install additional filter packs
3. Install a Semantic Language Database

Install Microsoft SQL Server

During the "Select Feature" step of the Microsoft SQL Server installation we only have to enable the feature "Full-Text and Semantic Extractions for Search" in the "Database Engine Services" group.

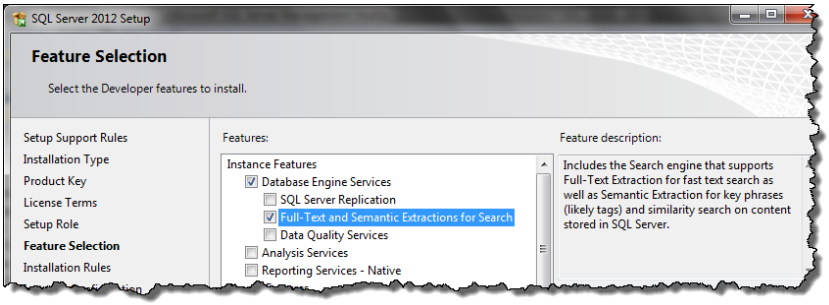

Figure 1: Enable semantic search

If you need file tables you have to enable the following options during the "Database Engine Configuration" step:

- **Enable FILESTREAM for Transact-SQL access**
- **Enable FILESTREAM for file I/O access** (this creates a new shared folder which can be used to manage the files in the file table).

After completing all steps we should be able to connect to the database engine using the Microsoft Management Studio.

Install additional filter packs

The default installation of Microsoft SQL Server only supports the most important file formats. For example, the newer Microsoft Word format ".docx" is not supported by default. To use more file formats, we need to install filter packs which add support for additional file types.

The filter packs can be downloaded here:

- **Microsoft Office 2010 Filter Packs**
  http://www.microsoft.com/en-us/download/details.aspx?id=17062
- **Microsoft Office 2010 Filter Packs SP1**
  http://www.microsoft.com/download/en/details.aspx?id=26606 (32-bit)
  http://www.microsoft.com/download/en/details.aspx?id=26604 (64-bit)

After installing the filter packs we have to execute some SQL queries to enable them on the server. The following queries force the server to update the internal filter list.

```
USE master;
EXEC sp_fulltext_service @action='load_os_resources', @value=1;
EXEC sp_fulltext_service 'update_languages';
EXEC sp_fulltext_service 'restart_all_fdhosts';
```

To retrieve a list of supported formats execute the following query:

```
EXEC sp_help_fulltext_system_components 'filter'
```

Install a Semantic Language Database

As noted in the introductory chapter, semantic search is based on statistical analysis. The database engine uses a database with statistical base data to classify the keywords found in the indexed documents. This database is called **Semantic Language Database** and is not installed by default.

The database can be downloaded here:

- **Microsoft® SQL Server® 2012 Semantic Language Statistics**
  http://www.microsoft.com/en-us/download/details.aspx?id=29069

After installing the MSI package we should find the .mdf- and .ldf-files of the database in the following directory:

- **C:\Program Files\Microsoft Semantic Language Database**

After moving the two files to the desired location in the file system, we have to call the following queries to attach and register the semantic database:

```
CREATE DATABASE SemanticsDB
       ON ( FILENAME = 'Path\To\semanticsdb.mdf' )
       LOG ON ( FILENAME = 'Path\To\semanticsdb_log.ldf' )
       FOR ATTACH
GO
EXEC sp_fulltext_semantic_register_language_statistics_db @dbname = N'semanticsdb'
GO
```

With the following query we can check whether the Semantic Language Statistics database is installed:

```
SELECT * FROM sys.fulltext_semantic_language_statistics_database
```

2.2 Semantic Search on Tables and Columns

To use semantic search on a text column of a regular table, we have to create a new table with a primary key (column ID) and a text column:

```
CREATE DATABASE MyDatabase

USE MyDatabase
CREATE TABLE MyTable (
	ID int NOT NULL,
	Name varchar(50),
	Text varchar(max),
	CONSTRAINT PK_MyTable PRIMARY KEY (ID ASC)
)
```

The next step is to create a semantic search index for the text column. It is important to use the correct language code which corresponds to the language of the indexed text. The next figure shows a list of common languages and their codes.

Language	Code
German	1031
English (US)	1033
French	1036

Figure 2: Common language codes

The installed and available languages can be retrieved with this query:

```
SELECT * FROM sys.fulltext_semantic_languages
```

The following query creates an index for the “Text” column in the previously created table:

```
CREATE FULLTEXT CATALOG MyTableCatalog WITH ACCENT_SENSITIVITY = ON;

CREATE FULLTEXT INDEX ON MyTable (
	Text
	Language 1033
	Statistical_Semantics
)
KEY INDEX PK_MyTable
ON MyTableCatalog
WITH STOPLIST = SYSTEM
```

Now we can start inserting “documents” into the table:

```
INSERT INTO MyTable VALUES (1, 'MyName', 'This is a sample text.')
```

Using the **semantickeyphrasetable** method we can retrieve a list of the indexed keywords:

```
SELECT * FROM semantickeyphrasetable(MyTable, *)
```

For more information on “Semantic Search on Tables and Columns” read the corresponding MSDN article [9].

2.3 Semantic Search on File Tables

To use semantic search directly on files in the file system we have create a new file stream enabled table also known as file tables:

```
CREATE DATABASE MyDatabase ON PRIMARY (
    NAME = MyDatabaseDB,
    FILENAME = 'C:\MyDatabase\MyDatabaseDB.mdf'
),
FILEGROUP MyDatabaseFS CONTAINS FILESTREAM(
    NAME = MyFSDatabaseFS,
    FILENAME = 'C:\MyDatabase\MyDatabaseFS')
LOG ON (
    NAME = MyFSDatabaseLOG,
    FILENAME = 'C:\MyDatabase\MyDatabaseLOG.ldf')
GO
```

After the database has been created, we have to enable non-transactional access at the database level:

```
ALTER DATABASE MyDatabase SET FILESTREAM
       (Non_Transacted_Access = FULL, Directory_name = N'MyDatabase')
```

Now we can “transform” the table into a file table with the following query:

```
USE MyDatabase
CREATE TABLE MyFileTable AS FILETABLE WITH (FileTable_Directory = N'MyFiles')
```

The files in the file table can be accessed via a Windows Samba share at the following location:

- \\<MACHINENAME>\<SQLINSTANCENAME>\MyDatabase\MyFiles

If the share is accessible we have successfully created a new file table. Using the following query you can see a list of all files in the file table:

```
SELECT * FROM dbo.MyFileTable
```

After creating the database and the file table we have to setup a semantic search index on the file table. The following query creates the index. The primary key (“MyPK” in the listing) has to be replaced with the primary key name from previously created file table. If required replace the language with the desired language code. It is important to match the index language with the languages in the documents.

```
CREATE FULLTEXT CATALOG MyFileTableCatalog WITH ACCENT_SENSITIVITY = ON;
CREATE FULLTEXT INDEX ON MyFileTable
(
       name LANGUAGE 1033 STATISTICAL_SEMANTICS,
       file_type LANGUAGE 1033 STATISTICAL_SEMANTICS,
       file_stream TYPE COLUMN file_type LANGUAGE 1033 STATISTICAL_SEMANTICS
)
KEY INDEX MyPK
ON MyFileTableCatalog WITH CHANGE_TRACKING AUTO, STOPLIST = SYSTEM;
```

To test the created index, we copy some files into the file stream's Windows share and wait some time until the index has been built. The following query shows all keywords from all documents ordered by their scores:

```
SELECT name, document_key, keyphrase, score
FROM semantickeyphrasetable(MyFileTable, *)
INNER JOIN MyFileTable ON path_locator = document_key
ORDER BY name, score DESC
```

2.4 Queries

The semantic search feature in Microsoft SQL Server 2012 allows you to create three types of queries::

- **Key phrases:** The engine lets you access the statistically significant phrases in each document.
- **Similar documents:** This type of query helps you find similar or related documents or rows based on the key phrases contained in the documents.
- **Similarity explanation:** This query returns the key phrases that explain why two rows or documents were identified as similar.

The following chapters explain each of the queries in detail.. All samples are based on the same test data. The following SQL statement inserts the test data:

```
INSERT INTO MyTable VALUES
(1, 'a', 'This file is about SQL Server 2012 and Visual Studio 2012'),
(2, 'b', 'This file is about SQL Server 2012'),
(3, 'c', 'Evaluation of SQL Server and Visual Studio'),
(4, 'd', 'This file is about Netbeans'),
(5, 'e', 'This file is about Eclipse'),
(6, 'f', 'This file is about Visual Studio 2012')
```

Find key phrases in a document (semantickeyphrasetable)

Using a semantic search index it is possible to retrieve all indexed keywords and their score:

```
SELECT * FROM semantickeyphrasetable(MyTable, *) ORDER BY score DESC
```

Using a SQL join we can add the document name in the result:

```
SELECT ID, Name, keyphrase, score
FROM semantickeyphrasetable(MyTable, *)
INNER JOIN MyTable ON ID = document_key ORDER BY score DESC
```

With the following query we can retrieve a list of all documents which contain the keyword "SQL":

```
SELECT Name FROM semantickeyphrasetable (MyTable, *)
INNER JOIN MyTable ON ID = document_key WHERE keyphrase = 'sql'
```

Another example is the following query which counts the number of documents with a specific keyword:

```
SELECT COUNT(*) AS 'Number of SQL documents'
FROM semantickeyphrasetable (MyTable, *)
WHERE keyphrase = 'sql'
```

Find similar or related documents (semanticsimilaritytable)

With the procedure **semanticsimilaritytable** we can retrieve a list of similar documents for a given document. The third parameter of the procedure ("1" in the listing below) specifies for which document or row the related documents should be searched.

```
SELECT ID, Name, Text, Score
FROM semanticsimilaritytable(MyTable, *, 1)
INNER JOIN MyTable ON ID = matched_document_key
ORDER BY score DESC
```

The query yields a list of related documents and their score (first row shows the sample document):

ID	Name	Text	Score
1	a	This file is about SQL Server 2012 and Visual Studio 2012	-
3	c	Evaluation of SQL Server and Visual Studio	0.8554052
2	b	This file is about SQL Server 2012	0.8195257
6	f	This file is about Visual Studio 2012	0.6681765
4	d	This file is about Netbeans	0.3436309
5	e	This file is about Eclipse	0.2038287

As we can see the document "c" has the highest score because it contains both keywords "SQL Server" and "Visual Studio". The next two documents contain either "SQL Server" or "Visual Studio" and thus are lower rated.

Find the key phrases that make documents similar (semanticsimilaritydetailstable)

The procedure **semanticsimilaritydetailstable** retrieves the keywords that make two documents similar. To see the most important key phrases which make the previous sample documents (document "a" with PK 1 and document "c" with PK 3) similar, execute the following query:

```
SELECT keyphrase, score
FROM semanticsimilaritydetailstable(MyTable, Text, 1, Text, 3)
```

The result looks as follows:

Keyphrase	Score
sql	1
server	0.458963
visual	0.458963
studio	0.40654

3 Analysis

This chapter tries to give you some insight into the underlying mechanics of the semantic search feature.

3.1 Keyword occurrence

Question: Does the score depend on the number of occurrences in the document?

Example: A document with more occurrences of a keyword should score the keyword higher than a document with fewer occurrences of the keyword.

Test: Insert documents with different occurrence counts and check if the score changes.

Answer: Yes, the score depends on the keyword occurrences. If a keyword can be found multiple times in the text, the score will be bigger.

3.2 Document repository dependency

Question: Do the keyword scores depend on the other documents?

Example: If every document contains information about SQL then the key phrase SQL should have a lower score because it is less interesting as it can be found in every document.

Test: Test the search engine with various sets of documents.

Answer: The scores are always the same. They do not depend on the other documents, only on the semantic database (language specific).

3.3 Missing keywords

During my tests I found some weird behaviors. I inserted the following tuple into the database:

```
INSERT INTO MyTable VALUES (1, 'This file is about Netbeans.')
```

When executing the following query we only get one keyword: “file”. Where is the keyword “Netbeans”?

```
SELECT * FROM semantickeyphrasetable(MyTable, *)
       WHERE document_key = 1 ORDER BY score DESC
```

Using the following query we can query the stop words and check whether the word is defined as stop word:

```
select * from sys.fulltext_stopwords
```

The word “Netbeans” is not in the list and I could not find the answer why the word is missing in the index. If you experience unexpected results with your queries, you should check whether the desired keywords are correctly indexed because – as we have seen in this example – some keywords are simply missing.

3.4 Stemming performance

Question: How good is stemming really working? Are there differences in different languages (e.g. English and German)?

Test: We tested various words to see if they are stemmed or not.

English:

- Plural:
  - “children” (child) => “children”
  - “men” (man) => “men”
- Conjugated words:
  - “swimming” (to swim) => “swimming”
  - “drove” (to drive) => “drove”
- Compound words:
  - “toolbox” => “toolbox”
  - “airport” => “airport”
  - “homework” => “homework”

German:

- Plural:
  - “Pizzen” (Pizza) => “pizzen”
  - “Häuser” (Haus) => “häuser”
- Conjugated words
  - “schwam” (schwimmen) => {}
  - “tauchte” (tauchen) => “tauchte”
  - “fuhr” (fahren) => “fuhr”
- Compound words:
  - “Sonnenbrand” => “sonnenbrand”, “sonnen”, “brand”
  - “Briefträger” => “briefträger”, “briefen”, “träger”
  - “Abfahrtszeit” => “abfahrtszeit”, “abfahrt”

Result: In general, stemming is not really working. In English and German, nouns are not stemmed from the plural to the singular form – verbs also were not touched at all. Compound words are split only in German.

3.5 Stop word performance

Question: How good is the filtering of irrelevant words working?

Test: We tested some sentences in English and German.

English

- "They have affected world history on many occasions. "
  - **Added:** "occasions", "affected", "history", "world"
  - **Removed:** "they", "have", "on", "many"

German

- "Er schwimmt über den See. "
  - **Added:** "schwimmt", "see"
  - **Removed:** "er", "über", "den"
- "Sein Sohn war der österreichische Diplomat. "
  - **Added:** "österreichischer", "diplomat", "sohn"
  - **Removed:** "sein", "war", "der"

Result: It seems the stop word list is very robust and only relevant words are indexed. As already described in a previous section some words are removed despite they are relevant.

3.6 Calculation of document similarity scores

Sometimes it is useful to know how the similarity scores are calculated. First we will look at the **semanticsimilaritydetailstable** procedure which shows keyphrase scores for two documents. The following query shows the similarity score between two files:

```
SELECT keyphrase, score
FROM semanticsimilaritydetailstable(
      MyTable,
      Text, 1,
      Text, 2)
```

The result looks as follows:

Keyphrase	Score
server	0.6774681
sql	1
file	0.3112285

To examine the calculation algorithm we have to run the following query which yields all scores for all key phrases in the two previous documents.

```
SELECT ID, Name, keyphrase, score FROM semantickeyphrasetable(MyTable, *)
INNER JOIN MyTable ON ID = document_key
WHERE ID = 1 or ID = 2 ORDER BY ID, score DESC
```

The query result:

ID	Name	Keyphrase	Score
1	a	**sql**	1
1	a	**server**	0.6774681
1	a	visual	0.6774681
1	a	studio	0.6376049
1	a	**file**	0.5578786
2	b	**server**	1
2	b	**sql**	1
2	b	engines	0.6641803
2	b	**file**	0.5578786
2	b	search	0.5578786

As we can see the scores in the first query are calculated the following way: All keywords which occur in both documents are gathered and their scores multiplied. If we look at the keyword “file” we can see in the second table that it occurs in both documents with a score of 0.5578786 each. After multiplying 0.5578786 with 0.5578786 we get 0.3112285 which can be found in the first table.

The next question is how the scores in the **semanticsimilaritytable** are calculated. The following query shows an example query:

```
SELECT ID, Name, Score
FROM semanticsimilaritytable(MyTable, *, 1)
INNER JOIN MyTable ON ID = matched_document_key
WHERE ID = 2
ORDER BY score DESC
```

The result:

ID	Name	Score
2	b	0.6998509

During my tests I could not find out the exact algorithm, but the score depends on the number of matching keywords and their scores.

3.7 Thesaurus files

Question: For Full-Text search it is possible to define thesaurus files [10] which are used to define synonyms and word replacements. Do the thesaurus files also work with semantic search?

Test: First, we have to setup a thesaurus file. The thesaurus files are stored in the directory "MSSQL11.MSSQLSERVER\MSSQL\FTData" with the filename "ts<language>.xml". The file "tsglobal.xml" contains synonyms for all languages. To test the feature, simply uncomment the "thesaurus" XML tag in the file. To load the thesaurus run the following query on the correct database with the correct language code:

```
EXEC sys.sp_fulltext_load_thesaurus_file 1033;
```

Now we insert the following two texts into our test table.

Text
Evaluation of IE in corporate environments.
This file is about Internet Explorer.

Using the FREETEXT procedure we can search for a text in the indexed text column. The following query should use the thesaurus file and therefore should show both inserted texts as "IE" and "Internet Explorer" are defined as similar words:

```
SELECT * FROM [MyTable] WHERE FREETEXT([Text], 'Internet Explorer')
```

After verifying that the full-text search with word replacement is working, we will try if it is working in semantic search with the next SQL statement.

```
SELECT keyphrase, score
FROM semanticsimilaritydetailstable(
       MyTable,
       Text, 5,
       Text, 6)
```

Both documents should have the "Internet Explorer" as keywords but they have not. The previous SQL query shows an empty list of similar keywords.

Answer: It seems that thesaurus files are not working with semantic search.

3.8 Custom stop words

Question: Sometimes it is required to define own stop words. Do custom stop words work with semantic search?

Test: The following figure shows how to create a stop list which extends the SYSTEM (default) stop word list [11].

```
CREATE FULLTEXT STOPLIST [myStoplist] FROM SYSTEM STOPLIST;
ALTER FULLTEXT STOPLIST [myStoplist] ADD 'lazy' LANGUAGE 'English';
ALTER FULLTEXT STOPLIST [myStoplist] ADD 'jumps' LANGUAGE 'English';
ALTER FULLTEXT STOPLIST [myStoplist] ADD 'dog' LANGUAGE 'English';
GO
```

The created stop word list can be used when creating a new full-text index as shown below.

```
CREATE FULLTEXT INDEX ON MyTable (
       Text
       Language 1033
       Statistical_Semantics
)
KEY INDEX PK_MyTable
ON MyTableCatalog
WITH STOPLIST = [myStoplist]
```

Now we insert the following text into our test table and compare the indexed keywords without and with the custom stop list.

Text
The quick brown fox jumps over the lazy dog.

- Using default stop list: **lazy, jumps, dog, fox, quick, brown**
- Using custom stop list from above: **fox, quick, brown**

As we can see, the stop list is working.

Answer: Custom stop lists work without problems with semantic search.

4 Conclusion

4.1 Strengths

Installing and using Microsoft SQL Server 2012 semantic search is simple. There are only three types of queries possible and their usage is easy to comprehend. According to Microsoft [11] a full-text index scales over 100 million of documents and thus can be used with very large sets of documents. The current implementation of semantic search's main purpose is to find similar documents for a given document. This functionality works satisfying, however you have to define own stop words to filter out noise words. One advantage of the feature is that it works on regular text columns as well as documents stored in the file system.

4.2 Limitations

At the moment, Microsoft SQL Server 2012 supports only unigrams. This limits the usage to simple keyword comparison instead of some more advanced semantic intelligence. In addition the usage scenarios of semantic search are very limited as it is generally only possible to search for similar documents for a given document. At the moment it is not possible to search for sentences in these documents. The stemming implementation can cause problems as it often does not transform the words to its correct stem word.

4.3 Opinions

According to various comments found on the internet [13], the semantic search feature is too limited to be used in practical business use. Actually a lot of mechanisms do not work correctly (e.g. stemming) – others are simply not implemented (e.g. multi-word phrases). In my eyes, semantic search is only a small extension to full-text search. To really deliver great value we need the ability to search for whole sentences and the server should return texts with similar passages and similarity scores. However because the simple installation and usage it may be a nice addition for simple usage scenarios.

5 References

[1] [Online]. Available: http://www.freebase.com/. [Accessed 22 1 2013].

[2] [Online]. Available: http://www.google.com/. [Accessed 22 1 2013].

[3] [Online]. Available: http://www.wolframalpha.com/. [Accessed 22 1 2013].

[4] [Online]. Available: http://www.seco.tkk.fi/publications/2005/makela-semantic-search-2005.pdf. [Accessed 22 1 2013].

[5] [Online]. Available: http://yury.name/algoweb/08algoweb-hand.pdf. [Accessed 22 1 2013].

[6] [Online]. Available: http://en.wikipedia.org/wiki/Semantic_search. [Accessed 22 1 2013].

[7] J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries".

[8] Y. Chen, E. K. Garcia and R. M. Gupta, "Similarity-based Classification: Concepts and Algorithms".

[9] [Online]. Available: http://msdn.microsoft.com/en-us/library/gg509116.aspx. [Accessed 22 1 2013].

[10] [Online]. Available: http://msdn.microsoft.com/en-us/library/ms142491.aspx#how_queries_use_tf. [Accessed 22 1 2013].

[11] [Online]. Available: http://msdn.microsoft.com/en-us/library/cc280405.aspx. [Accessed 22 1 2013].

[12] R. Mistry and S. Misner, Introducing Microsoft®SQL Server2012, Microsoft Press.

[13] [Online]. Available: http://www.simple-talk.com/sql/database-administration/exploring-semantic-search-key-term-relevance/. [Accessed 22 1 2013].

5.1 Additional sources

[1] Ross Mistry and Stacia Misner, “Introducing Microsoft® SQL Server 2012”, Microsoft Press.

[2] Microsoft’s MSDN

6 Appendix

6.1 Figures