SEMILAR API 1.0

User guide

Authors: Rajendra Banjade, Dan Stefanescu, Nobal Niraula, Mihai Lintean, and Vasile Rus

Contact: Rajendra Banjade at rbanjade@memphis.edu

7/29/2013

***Important**: The SEMILAR library is free to use only for non-commercial, academic and research purposes. Please check the Licensing terms on the SEMILAR website at http://www.semanticsimilarity.org.*

Outline

Introduction

The goal of the SEMantic simILARity software toolkit (SEMILAR; pronounced the same way as the word 'similar') is to promote productive, rigorous, and fair research advancements in the area of semantic similarity. Semantic similarity is a widely adopted approach to language understanding in which the meaning of a text A is inferred based on how similar it is to another text B, called the benchmark text and whose meaning is known. For instance, to assess the correctness of a student response to a Physics question in an Intelligent Tutoring System, e.g. DeepTutor (www.deeptutor.org), the student response is compared to an ideal response, i.e. the benchmark response, provided by a Physics expert.

The SEMILAR software environment offers users, researchers, and developers, easy access to fully-implemented semantic similarity methods in one place through both a GUI-based interface and a library. Besides productivity advantages, SEMILAR provides a framework for the systematic comparison of various semantic similarity methods.

It should be noted that SEMILAR offers measures for computing the semantic similarity of texts at various levels of granularity: word-to-word, sentence-to-sentence, paragraph-to-paragraph, and document-to-document, or a combination of these such as sentence-to-document.

This document describes the SEMILAR library API. The GUI-based SEMILAR application is described in a separate document. This document presents concisely the various semantic similarity methods available in the SEMILAR library (Java) along with guidelines on how to use them.

Please visit http://www.semanticsimilarity.org/ for further details and recent updates. See the example codes and readme file for a quick start. And feel free to contact us for any issues, suggestions you may have. Please find the details about references, attributions, and citation information in the later sections.

Dr. Vasile Rus (vrus@memphis.edu), Director.

Rajendra Banjade (rbanjade@memphis.edu)

Dr. Mihai Lintean (mclinten@memphis.edu)

Nobal Niraula (nbnraula@memphis.edu)

Dr. Dan Stefanescu (dstfnscu@memphis.edu)

Quick Overview

The following list of Frequently Asked Questions (FAQ) works as a quick overview of the SEMILAR API. The questions are meant to save you time. In case you have a question not in this list, please contact us and we will be happy to answer it and then add it to this list.

What are the similarity methods available in SEMILAR?

- SEMILAR API offers a variety of similarity methods based on WordNet (Fellbaum, 1998), Latent Semantic Analysis (LSA; Landauer et al., 1998), Latent Dirichlet Allocation (LDA; Blei et al., 2003), BLEU (Papieni et al., 2002), Meteor (Denkowski and Lavie, 2011), Pointwise Mutual Information (PMI) (Church et al., 1990), methods that use syntactic dependencies, optimized lexico-syntactic matching methods based on Quadratic Assignment, methods that incorporate negation handling, etc. Some methods have their own variations which, coupled with parameter settings and user's selection of preprocessing steps, could result in a huge methods space.

What is the granularity of similarity methods?

- SEMILAR contains methods to measure the semantic similarity at word level, i.e. word-to-word measures, sentence level or sentence-to-sentence measures, paragraph-to-paragraph measures, and document-to-document measures. In addition, the methods can be applied to compute the semantic similarity of texts of different granularities such as word-to-sentence similarity, sentence-to-document similarity, etc. For instance, in summarization a paragraph-to-document similarity measure could be useful to compare the summary paragraph to the source document.
- Please note that in order to expand from word-to-word similarity measures to larger texts such as sentence-to-sentence some methods require a special, additional step whereas some other methods are directly applicable between texts of any granularity. For example, there are variations of LDA-based similarity methods that work at word-level and others that can be used to compute sentence-level similarity. Expanding the word-level LDA-based measures to sentence level require a special step. On the other hand, LSA can be directly used to compute the similarity of two words or two sentences by simply relying on the same cosine similarity operation between two LSA vectors (an LSA vector must be obtained for a sentence using vector algebra, i.e. simply adding up the individual words' LSA vectors).

What are the word-to-word similarity methods available in SEMILAR?

- Please go to word-to-word similarity section. Note that there are sentence level or larger text level similarity measures that are expanded from word-to-word similarity measures. Also, keep

in mind that some similarity methods are resource-demanding as they are backed by large data-driven models (e.g. LSA, LDA). In order to reduce the memory requirements, we recommend employing them in separate runs.

Where can I find the details about the methods/algorithms available in SEMILAR?

- This document only describes the API of the most important methods implemented in SEMILAR. We do not have (yet) a single document describing in detail all the methods and algorithms available in SEMILAR. We offer a comprehensive list of references to the original publications that introduced the various methods. Please find the reference section for more details.

Similarity and Relatedness are quite different things. Have you categorized them?

- Though similarity and relatedness are quite different concepts, we refer them as similarity in general. Some of the methods measure similarity more than others, but all of them can be considered measures of relatedness. We should refer to their descriptions in details to characterize them. Corpus based models usually measure the relatedness.

I am not just doing word-to-word or sentence-to-sentence similarity research, my research is on text classification, clustering, text mining, information retrieval (or something related), machine translation evaluation, etc. How can I best utilize this tool?

- Certainly there are many ways of using word-to-word, sentence-to-sentence similarity or relatedness measures in information retrieval, text mining, clustering, classification, machine translation evaluation, etc. You may consider using word-to-word and/or sentence-to-sentence level similarity to document level or use document level similarity methods (The currently available document level similarity method is LDA based method. Work is on progress; please keep on visiting SEMILAR website for the recent updates) available in SEMILAR API itself. The typical use of word-to-word similarity method for document similarity would be to pair up the words from two documents and normalize the score (such as dividing the score by document lengths).

What are the recent updates to the SEMILAR API?

- This document covers the methods available in the SEMILAR API as of June 2013. Please visit http://www.semanticsimilarity.org for more details and latest updates.

Which programming language is used to create SEMILAR API?

- Java (jdk 1.7). And we plan to continue the development in this language.

How large is the SEMILAR package?

- It is a large library and application because it relies on large models and packages. Most of the basic NLP tools on which we rely come with big models such as the Stanford NLP or the Open NLP parser models. Other resources are relatively large as well (a couple of hundred MBs) such

as the WordNet lexical database. In addition, our similarity methods also require pre-built models. For example, LSA spaces, LDA models, and Wikipedia PMI data are large components by themselves. If a user wants to utilize selective methods, there is no need to download everything. SEMILAR can be downloaded in separate zipped files for ease of customization and setup that fits various needs. The whole SEMILAR package, which includes everything, has about 1.1 Gb.

Which corpora or data sets are needed?

- We have generated LSA spaces and LDA models using the TASA corpus and the Wikipedia (early spring 2013 version) corpus, whereas for PMI calculations we have used only Wikipedia. These models can be downloaded from the SEMILAR website. The user may generate new models based on different corpora, with different preprocessing steps, or other settings. The SEMILAR API allows the user to specify new, non-default model names and paths. Please see the corpora details section for more details about the corpora we are using.

I want to generate and use LSA/LDA models using different corpora or requirements. Is it possible to generate and use my models in SEMILAR?

- Yes, you can develop LSA/LDA models using your own corpus. But you have to follow the format of the model files, and certain file naming requirements for your model name, etc. We also offer a facility to generate LSA/LDA models (in progress; please see the section Using LDA for more details on creating LDA models and the References section about the tools we are using). It is recommended to generate models using the SEMILAR API as the output follows the format used by the other SEMILAR components.

What preprocessing steps are usually executed and what are the tools you are using?

- Some similarity methods require certain kinds of preprocessing such as POS tagging, parsing, stemming/lemmatization, etc. Tokenization is needed by all methods. The SEMILAR preprocessor gives users the option of selecting either StanfordNLP or OpenNLP tools for tokenization, tagging, and parsing. For stemming, Porter's stemmer or WordNet based stemmer (the latter guarantees the stem is an actual word, i.e. dictionary entry) are available.

Can I skip preprocessing or do it myself?

- You may preprocess your texts without using the SEMILAR preprocessor but your responsibility would be to create certain objects and populate corresponding field values in these objects.
- You may not need to do any preprocessing or just do the basics to use the semantic similarity methods. Please check the preprocessing requirements for particular methods you may want to use.

How much time consuming methods are there in SEMILAR?

- Most of the methods are quite fast. Some optimization methods that also rely on syntactic or other types of information may be slower. If you want to run it in a typical workstation, then running different methods together can be quite heavy or may not work. Please consider running it in a more powerful machine where you can run multiple methods together.

How much memory is consumed?

- It depends on the particular method. Most of the implemented methods should work on regular desktop and laptop computers (having at least 6GB memory). Running many methods together can be problematic in the regular PC.

Which WordNet version is used?

- WordNet 3.0.

Can SEMILAR be used for languages other than English?

- Similarity measures that are available in SEMILAR have been developed for English and there are no models included for languages other than English. But it is possible to adapt the methods to other languages. For instance, you can generate LSA/LDA models using texts from a target, other-than-English language (remember that you can develop LSA/LDA models using interface/functions available in SEMILAR API itself) and then use the SEMILAR LSA and LDA similarity measures to compute the similarity of texts in the target language.

Where can I find references and citation information?

- Please find below the sections - SEMILAR Citation info and References for more details.

What are the licensing terms of using SEMILAR?

- The SEMILAR library and application are free to use for non-commercial, academic and research purposes. Please check the Licensing terms on the SEMILAR website at http://www.semanticsimilarity.org. Please note that we provide licensing information about third-party components that are being used in SEMILAR in the reference section in on the website at http://www.semanticsimilarity.org. Complying with the SEMILAR licensing terms implies complying with license agreements issued by the third parties for the corresponding components included in SEMILAR. Please read the license agreement first before downloading and installing SEMILAR.

Are there any examples for quick start using it?

- Yes. Example code is available in the SEMILAR package in the extracted SEMILAR folder. There are different methods, options, possible values of parameters, different models you can choose

from, etc. You have to go through this guide and latest information from the website to make best use of SEMILAR.

I want to run all methods at once, is it possible?

- Yes, it is. Some of the preprocessing tools and similarity methods have to load huge models in memory. You can probably run all other methods at once if your machine has at least 8 GB of memory.

What platforms (OS) does it support?

- SEMILAR API can be used in Linux and Windows. Jdk 1.7 or higher is required.

I have encountered some error or exception, how can I diagnose?

- We try to provide as much support as possible. Please provide enough details when you encounter any issues. It is possible that errors can be caused by missing required files, misplaced folders (especially after extracting the zipped files), misspelling, incorrect input format, corrupted downloaded file, etc.

What is the similarity score if the given word(s) is not in the vocabulary? For example: Your LSA model may not have some of the words I would like to see the similarity score for.

- We are scrutinizing all the methods to make sure that users will not get confused (for example: if a similarity score of zero is returned for a given word pair it may be because one or both of the words are not available in the model). Many of the similarity measures return a value of -9999 in cased when it was not possible to calculate the similarity for the given word or sentence pairs.

I have little background on one or more similarity methods. Rather than description of methods, can I find solid steps (or examples) to get the results?

- We recommend users understand the fundamentals of each method before using their implementations available in SEMILAR. To make it easy (at the extent this is possible) for users to start using the SEMILAR methods with minimal leading effort, we have different resources including this manual and the package including example codes.

I have few questions, issues about using it, can I get some help?

- Yes sure. Please feel free to write to Rajendra Banjade (rbanjade@memphis.edu) and Dr. Vasile Rus (vrus@memphis.edu).

Data Objects

The following are the data objects you should be familiar with. Please note that it doesn't provide trivial details.

Word:

Class	semilar.data.Word
Description	This object represents a word (or token).
Fields	Raw form – Without any preprocessing, as given by the user. Base form – stemmed/lemmatized form. POS – part of speech tag. isStopWord – is it a stop word? isPunctuation – is it a punctuation?
Enumeration	N/A
Methods	N/A

Sentence:

Class	semilar.data.Sentence
Description	This class represents a sentence.
Fields	Raw form – Without any preprocessing, as given by the user. Words – List containing the list of Words. Syntactic tree –Syntactic tree (string form). Dependencies – dependency information. Dependency tree – dependency tree (string form).
Enumeration	N/A
Methods	Getter/setters

Note: Details about document representation will be published soon.

Configurations

The required files and location where to put the downloadable resources are available at SEMILAR website**.** To avoid any potential problems, please create a specific folder for SEMILAR and organize all the data in their default locations as specified in the SEMILAR website so that you can save some time for your actual work. However, if you really need it, you can organize the resources differently and specify the locations using configuration manager described below.

If you want to put data files – Open NLP, StanfordNLP, WordNet, or LSA/LDA/PMI data files at some folders other than the default locations (as mentioned in the SEMILAR website), you can set the file/folder paths using static methods of configuration manager class. For example,

semilar.config.ConfigManager
ConfigManager.setSemilarHomeFolder(String path) – set the SEMILAR home folder (i.e. folder containing the SEMILAR API jar file. It is essentially the home folder of project that uses the SEMILAR API). ConfigManager.setSemilarDataRootFolder(String path) – set the SEMILAR data root folder. It includes LSA/LDA models, test data set etc. ConfigManager. setWordnetPath(String path) – set the Wordnet root folder (end with /). ConfigManager. setLsaRootFolder (String path) – folder where LSA models are kept. The LSA model files should be in the folder (in the LSA root folder) named as the model name. For example, TASA.

Preprocessing

During preprocessing, you have to provide a word, sentence, or larger text as input in the raw form. The preprocessor processes and creates an object representing the given input. For example, when the user provide sentence as input, then the preprocessor creates an object of class 'Sentence'.

If you want to use your own preprocessing, then your responsibility would be to populate the data object(s) described above. Please see the description of methods you want to use and their preprocessing requirements. Be aware that creating the same representation is quite tricky, so we recommend using the preprocessor of SEMILAR itself. Please see the example codes for usage details.

Word to word similarity/relatedness methods

All word-to-word similarity methods have implemented functions *computeWordSimilarity*(Word word1, Word word2) – requires POS tag AND/OR *computeWordSimilarityNoPos*(Word w1, Word w2) – doesn't require POS tag. If POS is not provided, it finds the maximum similarity score of all possible categories (whenever applicable) – verb, noun, adjective, and adverb. They return the similarity score in the range of 0 to 1. However, the scores distribution within this interval varies from method to method.

Please see the word to word similarity example codes available in the SEMILAR website for quick start using it and find the papers from the reference section for more details.

Sentence to Sentence similarity methods

Please note that there are basically two ways to calculate sentence to sentence similarity. One is to expand word-to-word similarity (i.e. use similarity of word in one sentence to a word in another sentence and by some means calculate sentence level similarity score) and another approach is to use the semantic representations of sentences to calculate the sentence similarity directly. All of the word to word similarity methods described above can be expanded for sentence level similarity.

However, it is not always possible to semantically represent the sentence directly using some resources such as in WordNet. There are cases when you can choose whether to start with word-to-word similarity and then expand it to sentence level, or use sentence level representation to measure sentence

similarity. For example, using LSA, one can use either the semantic representation of sentences (word vectors) to calculate sentence similarity, or expand word level similarity (using the semantic representations of individual words) to sentence level without explicitly representing the sentence.

For example, optimal matching solution for sentence to sentence similarity is based on word-to-word similarity measures. The optimal lexical matching is based on the optimal assignment problem, a fundamental combinatorial optimization problem which consists of finding a maximum weight matching in a weighted bipartite graph. Whereas, dependency based method requires word-to-word similarity as well as some grammatical relations.

The similarity methods return scores within the interval [0, 1]. However, their distributions vary from method to method.

Please see the sentence to sentence similarity examples available in the SEMILAR website.

Document to document similarity

We are adding similarity functions for bigger texts. As of July 2013, we an LDA similarity method is available. Using LDA for similarity is a two steps process: the first one is to infer the probability distributions of documents over topics based on some LDA model and the second one is to use those distributions to calculate the similarity of documents. Please see the example codes and inline comments for further details.

Corpus and Models

TASA corpus: LSA and LDA models are developed from the lemmatized TASA corpus.

The Touchstone Applied Science Associates (TASA) corpus contains 60,527 samples from 6,333 textbooks, works of literature, and popular works of fiction and nonfiction. It contains more than 17 million word tokens, and 154,941 word types. However, TASA is weighted toward language arts and social studies. “The TASA corpus contains approximately the quantity and quality of text that the average college-level student has experienced across his lifetime” (Jones et al., 2006, P. 510).

Category	Samples	Paragraphs
Language Arts	16,044	57,106
Health	1,359	3,396
Home Economics	283	403
Industrial Arts	142	462
Science	5,356	15,569
Social Studies	10,501	29,280
Business	1,0079	4,834
Miscellaneous	675	2,272
Unmarked	2,212	6,305
Total	37,651	119,627

Table: Break down of the TASA corpus.

Source: Comparing Semantic Space Models Using Child-direct Speech, by Brian Riordan, Page: 25

English Wikipedia articles (early spring 2013): LSA models and PMI values calculated from Wikipedia texts. Please contact us if you want to use the clean Wikipedia texts and PMI data. These are not available for the download from SEMILAR website as their size is quite large (8GB+).

Using LDA tool

We have provided an interface to LDA tool (Phan et al., 2008). A TASA LDA model is available in SEMILAR website for download and word-to-word similarity can be calculated based on that model. Sentence to sentence similarity expanding the word-to-word is also possible without inferring the document distribution over topics.

However, to measure the similarity of sentences or larger text by inferring the document distributions over topics, you have to first infer the documents distributions over topics based on the available LDA model (TASA LDA model by default) or develop the LDA models from your corpus and then infer distributions for your documents. So, it's a two steps process. Model development (or you use the LDA models available in SEMILAR website) and inferencing.

Below is some description of interface to the LDA tool. Please see the example codes and inline comments for better understanding on how to use it.

LDA Estimator:

If you just want to infer the probability distributions using the already developed LDA model (LDA model generated using TASA corpus is available in SEMILAR website), please skip this and go to the section: LDA Inferencing (below).

Class	semilar.tools.lda.LDAEstimator
Description	This is a wrapper around the JGibbLDA tool (Phan et al., 2008) for estimating document distribution over topics and topic distribution over the words in the vocabulary. First line of the input file should contain the number of documents in that file. Document means anything that is in the single line. So, if you want to create LDA model using documents containing multiple lines, make a single line for that document. Please find the example code and input/output files in the similar package.
Parameters	LDA data folder – folder where the input data file exists. The output will be generated in the same folder. Input file name – Name of the input data file. The first line should contain the number of documents in that file. Model name – name the LDA model. Note that the output files will be created with this name, and during inferencing (please see below), this name will be used while loading model for the inferencing. Number of topics, Alpha, Beta, Number of iterations, words per topic – please see LDA documentations.
Function	startEstimation() – starts estimation.
Dependencies	N/A
Preprocessing	Preprocess the input text as you like before providing it to the LDA for modeling.
Input data	File containing the documents. The first line should contain the number of documents in that file. Please find the sample input file for LDA estimator available the SEMILAR package.
Output	Model files in the lda data folder specified above. The output files are, <model-name>.info – information about the model such as number of topics etc. <model-name>.phi - topic distribution over words. <model-name>.theta – document distribution over topics. <model-name>.twords – topics <model-name>.wordmap – Vocabulary along with their frequencies.

LDA Inferencing:

Class	semilar.tools.lda.LDAInferencer
Description	This is a wrapper around the JGibbLDA tool (Phan et al., 2008) for inferencing document distribution over topics and topic distribution over the words in the vocabulary. Please find the example code and input/output files in the similar package.
Parameters	LDA data folder – folder where the input data file and model files generated during estimation exist (please see LDA estimation). The output will be generated in the same folder. Input file name – Name of the input data file. Model name – name the LDA model. Note that the model name should match with the name of the model you estimated.
Function	startInferencing () – starts inferencing.
Dependencies	N/A
Preprocessing	Preprocess the input text as you like.
Input data	File containing the documents. First line of the input file should contain the number of documents in that file. Document means anything that is in the single line. So, if you want to create LDA model using documents containing multiple lines, make a single line for that document. Model files – The output generated during estimation (please see above). The model name should match. **Note**: If you want to use the TASA model and do infer those probability distributions for your input data, then keep the TASA model files (please find the LDA model files and their default location from similar website) to the LDA directory you specified and give the model name "TASA".
Output	Model files in the lda data folder specified above. The output files are, <input file name>.<model-name>.info – information about the model such as number of topics etc. <input file name>.<model-name>.phi - Topic distributions over words. <input file name>.<model-name>.theta – Document distribution over topics <input file name>.<model-name>.twords – Topics <input file name>.<model-name>.wordmap – Words and their frequencies in the corpus.

SEMILAR Citation info

Rus, V., Lintean, M., Banjade, R., Niraula, N., and Stefanescu, D. (2013). SEMILAR: The Semantic Similarity Toolkit. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, August 4-9, 2013, Sofia, Bulgaria.

References

Blei, David M., Andrew Y. Ng, and Michael I. Jordan, (2003). "Latent dirichlet allocation." *the Journal of machine Learning research* 3: 993-1022.

Church, Kenneth Ward, and Patrick Hanks. (1990). "Word association norms, mutual information, and lexicography." *Computational linguistics* 16.1: 22-29.

Fellbaum, C. (1998). "WordNet: An Electronic Lexical Database". Cambridge, MA: MIT Press.

Landauer, Thomas K., Peter W. Foltz, and Darrell Laham. (1998). "An introduction to latent semantic analysis." *Discourse processes* 25.2-3: 259-284.

Denkowski, Michael and Alon Lavie (2011). "Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems", *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*.

Papineni, Kishore, et al. (2002). "BLEU: a method for automatic evaluation of machine translation." *Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics*, 2002.

Rus, Vasile, Lintean Mihai. (2012). "A Comparison of Greedy and Optimal Assessment of Natural Language Student Input Using Word-to-Word Similarity Metrics".

Rus, Vasile, Nobal Niraula, and Rajendra Banjade. (2013). "Similarity Measures Based on Latent Dirichlet Allocation." *Computational Linguistics and Intelligent Text Processing*. Springer Berlin Heidelberg, 459-470.

Phan, Xuan-Hieu, Le-Minh Nguyen, and Susumu Horiguchi. (2008). Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. *In Proc. of The 17th International World Wide Web Conference (WWW 2008)*, pp.91-100, Beijing, China.

Getting started with SEMILAR API 1.0

NOTE:

1. Please see the user agreement file first (it should be available in the SEMILAR download package or you may find it from the semanticsimilarity.org).
2. Please see the introductory sections of this user manual. You will find the overview of SEMILAR and is meant to save your time.
3. Examples help a lot to quick start using SEMILAR, please find them in the SEMILAR package or download from the website. Please see the details below on how to make them work.
4. If you want to use only certain methods, you may not need to download some of the data files. You can find more about this on SEMILAR website. Some files are very big and they are not available for download (e.g. cleaned Wikipedia text, PMI values). Please email us to get them.
5. Please visit http://semanticsimilarity.org for recent updates.
6. If you have any issues, questions, suggestions or need some help, please feel free to contact Rajendra Banjade (rbanjade@memphis.edu) and/or Dr. Vasile Rus (vrus@memphis.edu).

Prerequisites

1. Jdk 1.7 or higher.
2. OS: Windows/Linux.
3. Can be run on regular workstations. Running different methods together may resource demanding .

A: Steps for downloading SEMILAR package and Creating test project

1. Please create a Java project from Eclipse or Netbeans. Let's say the project name is SemilarDemo and the project home folder is SEMILAR home.
2. Download the SEMILAR main package and unpack it in the SEMILAR home folder. The SEMILAR project home folder will contain SEMILAR API jar file and some dependent files.
3. Download the example code from the SEMILAR website. Extract the zip file and add those example code files to your project. You may need to fix the package name in the example code files to match with the package name you imported them into.
4. Compile errors? Add the SEMILAR API into the Classpath (i.e. add the SEMILAR jar into your project). SEMILAR API should be in the SEMILAR home folder.

B: Downloading and setting up data files

1. Create a folder: let's say named Semilar-data. We will call it as SEMILAR data home folder (please note that SEMILAR home folder is the demo project home folder, as described above. But this is the SEMILAR data home folder).
2. Download LSA model files. Extract it and put the LSA-MODELS folder in the SEMILAR data home folder.
3. Similar to LSA, download the LDA model files. Extract it and put the LDA-MODELS folder in the data home folder.
4. Download and extract the word-to-word similarity test dataset (Word2Word-Similarity-test-data) and put that in the data folder.
5. Download and extract the LDA tool test dataset (LDA-tool-test-data) and put that in the data folder.

C: Running example codes

1. Please read the comments in the code.
2. Once you read the summary below, please see the code for the details.
3. *Word2WordSimilarityTest.java* file contains the demonstration code for word-to-word similarity. Comment/uncomment some of the word metrics, as running them altogether may be quite heavy (depending on the machine you are using). Please set the SEMILAR data home folder in the code. For example:
   ```
   ConfigManager.setSemilarDataRootFolder("C:/Users/<user name>/data/Semilar-data/");
   ```
4. *Sentence2SentenceSimilarityTest.java* file has an example code for sentence to sentence similarity. Comment/uncomment some of the methods and run the file. Running all of the methods together may not work (as they require a lot of memory) if you are using a regular machine. Similar to the word-to-word similarity test file, set the SEMILAR data home folder. Please note that for METEOR method, you have to provide the project home folder.
5. *LDABasedDocumentSimilarityTest.java* file shows how to measure the similarity of documents using LDA based method. Please note that document may be a single sentence or bigger text, but we refer them as document especially while working with LDA. Measuring similarity of documents using LDA is somewhat different from other methods, so we created a separate example code file.
6. *LDATest.java* file contains the example code showing how to use LDA tool to estimate the probability distributions and infer them for the new documents based on the already available LDA model. Please see the details about the LDA tool and reference about it in the SEMILAR API guide available in the website or available with the SEMILAR package.
7. Got errors? Usually the source of errors are missing files, extracting the files in the wrong place, providing the wrong data path etc. If you really got into trouble, please feel free to contact us.
8. Once you are able to run all or some of the methods, you may try changing parameters or try using different models etc.