Semantics Models (semmod) Documentation

Simon Dennis
School of Psychology
University of Adelaide

Benjamin Stone
School of Psychology
University of Adelaide

In both intelligence and command and control operations the ability to identify and process natural language is pivotal. The task is made difficult by the volume of such information available making automated methods important in narrowing the search for crucial information. Unlike existing search engine technologies that are successful on the world wide web, emphasis must be placed not only on the precision of retrieved results, but also on recall. There are a number of methods for extracting semantic information that have been introduced in recent years that have yet to be compared systematically in military-like contexts. In this package we implement some of the more prominent methods, in preparation for there use in a systematic comparison. The methods we intend to cover are:

1. Vector Space Model (Salton, Wong & Yang, 1975)
2. Latent Semantic Analysis (Martin & Berry, 2007)
3. the topics model (Griffiths & Steyvers, 2002)
4. Non-negative matrix factorization (Lee & Seung, 1999, Ge & Iwata, 2002)
5. Sparse Non-negative matrix factorization (Shashua & Hazan, 2005)
6. Independent Components Analysis (Isbell & Viola 1998)
7. Sparse ICA (Bronstein, Bronstein, Zibulevsky & Zeevi, 2005)
8. Syntagmatic Paradigmatic model (Dennis, 2005)
9. Constructed Semantics Model (Kwantes, 2005)

Apart from the syntagmatic paradigmatic model, these models all start with the same input representation, but produce decompositions with somewhat different properties. For instance, Griffiths and Steyvers (2002) showed that the representations produced by the topics model have neighborhood densities indicative of a small world process, whereas Latent Semantic Analysis does not. This observation is significant because free association norms, which presumably capture something of the structure of semantic organization in people, show similar neighborhood densities. In addition, methods like the topics model produce factors that are more interpretable than those in LSA. So, even if over all reliability is not improved these methods might be used to provide superior feedback to human operators.

The first step is to produce code capable of creating the latent representations employed by each of these models and to allow them to be queried in a number of ways.

Address correspondence to: Simon Dennis, School of Psychology, University of Adelaide, SA 5005, Australia. Telephone: +61 8 83034936. Facsimile: +61 8 83037177. Electronic Mail: simon.dennis@adelaide.edu.au

Table 1
Titles for Topics on Music and Baking

Label	Titles
M1	*Rock* and *Roll Music* in the 1960s
M2	Different *Drum Rolls*, a *Demonstration* of Techniques
M3	*Drum* and Bass *Composition*
M4	A Perspective of *Rock Music* in the 90s
M5	*Music* and *Composition* of Popular Bands
B1	How to Make *Bread* and *Rolls*, a *Demonstration*
B2	*Ingredients* for Crescent *Rolls*
B3	A *Recipe* for *Sourdough Bread*
B4	A Quick *Recipe* for Pizza *Dough* using Organic *Ingredients*

Programming Strategy

In order to make access to each of the methods as straightforward as possible we have chosen to implement the package in python. Python is a scripting language that has good support for the object oriented programming practices, well optimized and easy to use hash tables, as well as advanced text processing mechanisms. Several of the algorithms are, however, computationally intensive and so we have complemented the python modules with C extensions which encapsulate these operations.

All of the lexical semantics models operate by creating a latent structure, which we will term a space, that summarizes the information in a background corpus. The first step then is to provide this corpus. All of the modules assume that the corpus will be provided as an ASCII file containing a set of documents each separated by a blank line. In what follows we will use the example from Martin and Berry (2007). The corpus file is derived from the documents in Table 1.

Assuming that only the italicized words are to be considered, then the corpus file, which we call default.cor, would contain the text in Table 2:

Each model comes with two critical files - the command and the python module.

Command:. The command, just denoted by the name of the model (e.g. lsa) can be run from the command line and is able to create a space from a corpus file and to query the space once it has been created. To create an lsa

Table 2
default.cor file.

```
rock roll music

drum roll demonstration

drum composition

rock music

music composition

bread roll demonstration

ingredients roll

recipe dough bread

recipe dough ingredients
```

space for our example corpus, one would issue the command:

```
lsa -d 2
```

This will create a space file called MartinBerry.spc keeping two lsa dimensions (see Martin & Berry 2007, for an explanation of the LSA model and the meaning of dimensions. Note currently the command assumes local log weighting and global entropy weighting as outlined in Martin & Berry 2007). Now we can query this space using the following command:

```
lsa "music" "ingredients"
```

which will return the value -0.138, which is the cosine of the angle between the vectors representing "music" and "ingredients". As a check of surface validity we can query with

```
lsa "music" "roll"
```

which will return a value of 0.931, demonstrating that the model has learned that "music" and "drum" are more similar to each other than "music" and "ingredients".

The model is not constrained to single word inputs. So, one can also issue the command:

```
lsa "music" "roll rock"
```

or

```
lsa "music bread" "roll rock"
```

which return the values 0.989 and 0.781, respectively. In each case, lsa will choose the form of similarity and weighting calculations appropriate to compare the arguments.

lsa has a number of other useful flags. lsa –help provides the summary in Table D2. The -d and -i options control the calculation of the SVD and allow you to control the number of dimensions and number of iterations, respectively. The -f option allows you to force lsa to overwrite a space file that already exists. The -n option allows you to use a space name other than "default". The remaining options allow you to extract additional information about the space including vectors associated with the arguments (-v), the singular values (-s) and the time and date when the space was created (-t).

Module file:. The python module file (e.g. lsa.py) provides a module callable from python that provides the functionality associated with that model. For instance, to create a space from within python one would enter python and issue the following commands:

```
>> import lsa
>> space = lsaSpace("default.lsa")
>> space.Similairity("music" "drum bass")
```

Additional information on using python to create and interact with spaces can be found in the lsa.py file.

Testing

Testing code has been included in the main module files. To run the tests, change to the semmod directory and run the module (e.g.):

```
semmod/lsa.py
semmod/topics.py
```

Timing

Each model (except SP) was timed creating a space from a revised corpus derived from the King James Bible. A stoplist was used to remove function words. Also removed, were words that appeared less than twice in the corpus, and words that only appeared in one document In this revised corpus, the total number of terms was 6532, and the total number of documents was 1039.

The laptop that ran the timing trials had an Intel Centrino Duo 1.6 GHz CPU. The timing information can be reviewed below in Table 3.

Appendix A
Installing the code

Firstly, you should ensure that you have version 2.5 of python, version 1.0.1 of numpy and version 0.5.2 of scipy. The steps to install are as follows:

1. Unpack the tar file using tar -zxf semmod-1.0.tar.gz. Or unzip semmod-1.0.zip on windows.
2. In the semmod-1.0 directory, type setup.py install
3. Place commands from the semmod-1.0/bin directory in either a local bin directory and add to PATH, or by copying in a global bin directory.

Table 3

Timing of models on Bible.cor

Model	Timing (h:m:s)
Vectorspace	00:00:24
LSA	00:00:46
CSM	00:14:40
Topics	01:33:14
SICA (KMeans)	02:20:59
ICA	06:01:55
SPNMF	07:47:51
SICA (Instant Runoff)	08:50:48
SICA (Fuzzy CMeans)	greater than 19:00:00

Appendix B
The GNU General Public License

he licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU General Public License

Terms and Conditions For Copying, Distribution and Modification

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution

system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix C
Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> ¡one line to give the program's name and a brief idea of what it does.¿
> Copyright (C) ¡year¿ ¡name of author¿
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) ¡year¿ ¡name of author¿
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> ¡signature of Ty Coon¿, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix D
Help for Models

Table D1

vectorspace –help

```
Usage: vectorspace [options] [args]

Implements the Vector Space model. If a .vec space exists use -f to force the
creation of a new space. Similarities are cosines the vectors associated with
the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -n NAME, --name=NAME  Name of Space
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
```

Table D2

lsa –help

```
Usage: lsa [options] [args]

Implements Latent Semantic Analaysis. If a lsa space exists use -f to force
the creation of a new space. Similarities are cosines the vectors associated
with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -d DIMENSIONS, --dimensions=DIMENSIONS
                        Number of dimensions
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        number of iterations to run in SVD
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -S, --singular_values
                        Display the singular values
  -T, --time_date       Display creation time and date of current Space
```

Table D3

topics –help

```
Usage: topics [options] [args]

Implements the topics model by Griffiths and Steyvers (2002). If a topics
space exists use -f to force the creation of a new space. Similarities are 1.0
- the Jensen Shannon divergences of the probability distributions associated
with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -t TOPICS, --topics=TOPICS
                        Number of topics
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        number of iterations to run in SVD
  -B BURNIN, --burnin=BURNIN
                        number of iterations of burn in to run before starting
                        sampling
  -L LAG_BETWEEN_SAMPLES, --lag_between_samples=LAG_BETWEEN_SAMPLES
                        number of iterations between samples
  -a ALPHA, --alpha=ALPHA
                        Prior pseudocount for words
  -b BETA, --beta=BETA  Prior pseudocount for topics
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed for MT random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -p, --topic_probabilities
                        Display the topic probabilities
  -T, --time_date       Display creation time and date of current Space
```

Table D4

nmf –help

```
Usage: nmf [options] [args]

Implements Nonnegative Matrix Factorization. If a nmf space exists use -f to
force the creation of a new space. Similarities are cosines of the vectors
associated with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -d DIMENSIONS, --dimensions=DIMENSIONS
                        Number of dimensions
  -e EPSILON, --epsilon=EPSILON
                        Epsilon - the total sum squared errors under which to
                        stop optimizing.
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        maximum number of iterations
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
```

Table D5

spnmf –help

```
Usage: spnmf [options] [args]

Implements Nonnegative Matrix Factorization. If a spnmf space exists use -f to
force the creation of a new space. Similarities are cosines of the vectors
associated with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -d DIMENSIONS, --dimensions=DIMENSIONS
                        Number of dimensions
  -e EPSILON, --epsilon=EPSILON
                        Epsilon - the total sum squared errors under which to
                        stop optimizing.
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        maximum number of iterations
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
```

Table D6

ica –help

```
Usage: ica [options] [args]

Implements Independent Components Analysis. If a ica space exists use -f to
force the creation of a new space. Similarities are cosines of the vectors
associated with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -t TOPICS, --topics=TOPICS
                        Number of topics
  -n NAME, --name=NAME  Name of Space
  -u UPPER, --upper_threshold=UPPER
                        Upper threshold for ICA relevancy grouping
  -l LOWER, --lower_threshold=LOWER
                        Lower threshold for ICA relevancy grouping
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed for random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
```

Table D7

sica –help

```
Usage: sica [options] [args]

Implements Sparse Independent Components Analysis. If a sica space exists use
-f to force the creation of a new space. Similarities are cosines of the
vectors associated with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -d DIMENSIONS, --dimensions=DIMENSIONS
                        Number of dimensions
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        maximum number of iterations to run in K means and
                        fuzzy C means algorithms
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
  -K, --kmeans          Use K means clustering
  -F, --fuzzycmeans     Use fuzzy C means clustering
  -R, --instantrunoff   Use instant runoff clustering
```

Table D8

sp –help

```
Usage: sp [options] [args]

Implements the Syntagmatic Paradigmatic model. If a sp space exists use -f to
force the creation of a new space. Similarities are cosines of the vectors
associated with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -d DIMENSIONS, --dimensions=DIMENSIONS
                        Number of dimensions
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        maximum number of iterations to run in K means and
                        fuzzy C means algorithms
  -s RANDOMSEED, --random_seed=RANDOMSEED
                        Seed random number generator
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
  -K, --kmeans          Use K means clustering
  -F, --fuzzycmeans     Use fuzzy C means clustering
  -R, --instantrunoff   Use instant runoff clustering
```

Table D9

sica –help

```
Usage: csm [options] [args]

Implements the Constructed Semantics Model (Kwantes, 2005). If a .csm space
exists use -f to force the creation of a new space. Similarities are cosines
the vectors associated with the args.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current space
  -I, --information     Print information about the space.
  -n NAME, --name=NAME  Name of Space
  -v, --vector          Get vector associated with argument.
  -V, --verbose         Verbose.
  -T, --time_date       Display creation time and date of current Space
```