Lexical Semantics System 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 diffiucult by the volume of such information available making automated methods important in narrowing the search for crucial ifnormation. 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 prominant 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 Anaoysis (Martin & Berry, 2007)
3. the topics model (Griffiths & Steyvers, 2002)
4. Non-negative matrix factorization (Lee & Seung, 1999, Ge & Iwata, 2002)
5. Sparse ICA (Bronstein, Bronstein, Zibulevsky & Zeevi, 2005)

While these models all start with the same input representation, they 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.

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.

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*

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.crp, 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 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 menaing of

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 2
default.crp file.

```
rock roll music

drum roll demonstration

drum composition

rock music

music composition

bread roll demonstration

ingredients roll

recipe dough bread

recipe dough ingredients
```

dimensions. Note currently the command assumes local log weighting and global entropy weightng 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 3.

The -d and -i options control the calculation of the SVD and allow you to control the number of dimensions and number of iterations, respectivley. 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.spc")
>> space.cosTermDoc("music" "drum bass")
```

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

Table 3
lsa –help

```
usage: lsa [options]

options:
  -h, --help            show this help message and exit
  -f, --force           Force over-write of current Space
  -d DIMENSIONS, --dimensions=DIMENSIONS
                        Number of dimensions in Space
  -n NAME, --name=NAME  Name of Space
  -i ITERATIONS, --iterations=ITERATIONS
                        number of iterations to run in SVD
  -v, --vector          Get vector associated with argument.
  -s, --singular_values
                        Display Singular Values from current Space
  -t, --time_date       Display creation time and date of current Space
```