Online appendix

1. Text Annotation

Annotation involves the use of automatic tools for lemmatization and part-of-speech (POS) tagging of the corpus texts. Annotating syntactic constituents (parsing) is not common in LMDA.

Lemmatization, however, is a common practice for LMDA. It involves reducing inflectional word forms to their dictionary forms (lemmas), ensuring that different word forms are treated as the same lexical unit. For example, *books*, *booking, worked, working*, and *nicer, nicest* become *book*, *work*, and *nice* after lemmatization. Cases where a researcher might not wish to lemmatize include those in which variance in co-occurrence patterns across word forms might be meaningful.

In turn, POS taggers automatically assign labels to each word within a text, revealing their grammatical roles. For instance, in the sentence *The cat sat on the mat*, POS tagging identifies *the* as an article, *cat* and *mat* as nouns, *sat* as a verb, and *on* as a preposition. The value of POS tagging for LMDA lies in enabling researchers to select particular word classes to retain.

Several tools automate lemmatization and POS tagging, including:

- Tree-Tagger: Performs part-of-speech tagging and lemmatization for multiple languages.
- Stanford Parser: Offers comprehensive linguistic analysis, including lemmatization.
- Python Libraries: NLTK and SpaCy provide efficient lemmatization functions for text processing tasks.

2. Feature Counting

Feature—or lexical variable—counting involves tabulating the occurrence of the individual lexical items in each text. Corpus software like WordSmith Tools can conduct this step in the analysis, by producing a word list for each text. Additionally, this can be achieved through a programming script, like the Python code below, which reads all the texts from the *corpus* folder and outputs a text file containing the counts of each word in each text. If one is lemmatizing the corpus, at this stage, the corpus files should contain only the lemmas of interest (e.g. content words).

```
import os
from collections import Counter

def count_words_in_file(file_path):
    """Counts the words in a single text file."""
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read().lower()  # Convert text to lowercase for consistency
        words = text.split()
        word_counts = Counter(words)
        return word_counts

def process_corpus(corpus_folder, output_filename):
    """Processes all text files in a corpus folder and writes word counts to an output file."""
    with open(output_filename, 'w', encoding='utf-8') as output_file:
        for filename in os.listdir(corpus_folder):
            if filename.endswith(".txt"):
                file_path = os.path.join(corpus_folder, filename)
                word_counts = count_words_in_file(file_path)
                for word, count in word_counts.items():
                    output_file.write(f"{word} {filename[:-4]} {count}\n")

# Specify the paths to your corpus folder and desired output file
corpus_folder = "corpus"  # Replace with the actual path to your corpus
output_filename = "corpus_word_counts.txt"  # Replace with your preferred output filename

process_corpus(corpus_folder, output_filename)

print("Counts saved to output file:", output_filename)
```

The output file (corpus_word_counts.txt) will list each lemma, filename, and count on a separate line:

```
early t0001 1
bird t0001 1
get t0001 1
worm t0001 1
rose t0002 1
red t0002 1
violet t0002 1
```

Different output formats can be used. This “long format” is widely used in SAS. In this Element, we provide code snippets for SAS (www.sas.com/en_us/software/on-demand-for-academics.html), a powerful software package that can be used free of charge and R. However, the same functionalities can be obtained through other statistical software.

The following SAS code implements reading in the corpus_word_counts.txt file. Norming the counts to a fixed rate (e.g. per 1000 words) is generally recommended, but for reasons of space, this will not be demonstrated here.

```
DATA count;
  INFILE "/home/<path to your SAS directory>/corpus_word_counts.txt" ;
  length lemma $ 20 text $ 8 count 8 ;
  input lemma $ text $ count ;
RUN;
```

The following R code will achieve a similar outcome:

```
count <- read.table("<path to your file>/corpus_word_counts.txt", header = FALSE, col.names =
c("lemma", "text", "count"))
library(reshape2)
count_matrix <- dcast(count, text ~ lemma, value.var = "count", fill = 0)
```

3. Feature Selection

Features can be refined according to multiple criteria. Sometimes they are refined by word class (see section 3.2) or relationship to a node word examined through concordance analysis. They can also be refined by quantitative criteria such as frequency. The frequency counts can be filtered using dispersion or frequency markedness (or both), to narrow down the word pool and mitigate the influence of outliers that might distort the analysis. The choice of dispersion and/or frequency markedness depends on the specific research questions and corpus characteristics.

Dispersion:

- Dispersion measures how evenly a lemma is distributed across a corpus.
- Researchers often employ cut-off criteria to focus on lemmas with widespread presence, for example by specifying a criterial number of texts that a lemma must occur in to be retained (e.g. 10 percent of the texts, 5 texts, etc.).
- This eliminates lemmas skewed by idiosyncratic factors, such as those overly frequent in a single text but absent elsewhere.
- Removing such outliers ensures a word pool that accurately reflects the overall corpus distribution.

Frequency markedness (keyness):

- Frequency markedness highlights lemmas with unusual frequencies compared to a reference corpus.

- Keywords, a common operationalization of frequency markedness, identify words significantly more frequent in the target corpus than in a reference corpus.
- Software like WordSmith Tools automates keyword analysis, using either a subset of the target corpus or a separate benchmark corpus for comparison.
- Careful consideration of reference corpus selection is crucial for valid key word analysis.

4. Statistical Analysis

Factor analysis is the main statistical procedure in LMDA, as it uncovers the sets of correlated lexical items. Factor analysis can be performed using statistical packages like SPSS, SAS, and R. For detailed step-by-step guidance using these programs, refer to Friginal and Hardy (2019) and Egbert and Staples (2019).

- Factor analysis operates on a correlation matrix, which quantifies the relationships between word frequencies.
- Pearson (product moment) correlation coefficients are the default for interval data, where counts directly represent frequencies.
- For ordinal data (counts converted to an ordinal scale), polychoric correlation is often more suitable.
- For binary data (counts transformed into two-way categories), tetrachoric correlation is typically preferred.

In SAS, the following code will extract the factors for the dataset *count*, for variables v0001 through v1000, using standard Pearson correlations by default:

```
PROC FACTOR DATA= count
METHOD=principal
PLOTS=scree
VAR v0001 – v1000 ;
RUN;
```

Polychoric and tetrachoric correlation can be computed in SAS using the polychoric option in PROC CORR, as shown below. The procedure will automatically switch to tetrachoric correlation if the values are binary, otherwise it will calculate polychoric coefficients. In the code below, the correlation matrix is saved to a dataset called *binary*.

```
PROC CORR data = counts OUTPLC=binary POLYCHORIC;
VAR v001 – v1000 ;
RUN;
```

The following code will run factor analysis on the *binary* dataset. As this uses a correlation matrix for input, SAS requires that the number of observations (texts) be entered manually (in this example, 5000 texts).

```
PROC FACTOR DATA= binary (TYPE=corr)
METHOD=principal
PLOTS=scree
NOBS=5000;
VAR v001 – v1000 ;
RUN;
```

In R, similar functionalities can be achieved (without tetrachoric correlations, though) with the following code (adapted from Egbert & Staples, 2019, p. 138):

```
# Read in the word counts
count <- read.table("<your path>/corpus_word_counts.txt", header = FALSE, col.names =
c("lemma", "text", "count"))

# Convert the data into a matrix format
count_matrix <- xtabs(count ~ lemma + text, data = count)

# Install and load the 'psych' package if not already installed
if (!require(psych)) {
   install.packages('psych')
   library(psych)
}

# Perform an initial factor analysis
fa_results <- fa(count_matrix, nfactors = 20, rotate = "none")

# Generate a scree plot
scree(count_matrix)
```

The next step involves determining the number of factors to extract from the data. While several methods exist, the most commonly used in MDA is plotting the eigenvalues (measures of variance) for each factor on a graph (a scree plot, which the previous SAS command will generate; see case study #1, in section 4, for an example) and visually determining a break in the line (generally an “elbow” where the line momentarily flattens out), signaling that further factors offer diminishing returns on the variation captured. Striking this balance between capturing as large a proportion of the variation as possible and maintaining a parsimonious model is essential for a successful factorial extraction.

Before extracting the determined number of factors, it is recommended to drop variables with low commonalities, that is, variables which share little variance with other variables. Low

commonalities suggest that these variables do not correlate well with the overall factor structure, making them less useful for identifying underlying dimensions. A commonality cutoff often applied in LMDA is between 0.15 and 0.2.

The final factorial extraction is usually subjected to rotation, aimed at achieving a simpler and more interpretable factor structure. The goal of rotation is to redistribute the loadings within factors in a way that clarifies which variables have strong associations with each factor, thereby making the results easier to interpret and understand. The following code runs a rotated factor analysis extracting four factors using PROMAX rotation (although VARIMAX can also be used):

```
PROC FACTOR DATA= binary (TYPE=corr)
METHOD=principal
NFACTORS = 4
ROTATE = PROMAX
NOBS=5000;
VAR v001 – v1000 ;
RUN;
```

In R, the following code will run and rotate the factor analysis:

```
# Perform a factor analysis with four factors and promax rotation
results <- fa(count_matrix, fm = "pa", nfactors = 4, rotate = 'promax')
print(results, cut = 0)
```

Once factor extraction has been completed, factor scores can be calculated as a way to represent the incidence of each dimension on each text. A minimum loading value, such as 0.3, needs to be established, in order to exclude variables with weak associations to the factor. The analyst must inspect the factor pattern table and list all the variables in each factor whose loadings meet or exceed the cutoff value. Since a variable may have loadings above the cutoff threshold for multiple factors, during scoring a variable is exclusively assigned to the factor where it has its highest loading; generally, these secondary loadings are reported in parentheses. However, for the purpose of interpreting the factors, a variable can be considered relevant to all factors where it has successfully loaded.

Factor Score Calculation Process:

1. List the variables (lemmas) belonging to each pole of the factor, whose loading is higher than the cutoff value.
2. Optionally, standardize these counts for better comparability (see below).
3. Subtract the sum of negative pole counts from the sum of positive pole counts.

Interpreting the Factor Score:

- Positive score: Text aligns more strongly with the positive pole's linguistic features.
- Negative score: Text exhibits a greater affinity for the negative pole's characteristics.
- Score magnitude: Reflects the degree of alignment with a particular pole.

Z-scores, a common standardization method, can be computed prior to factor scoring to control for variable magnitude, ensuring that high-frequency variables do not disproportionately impact factor scores compared to low-frequency ones. The resulting z-scores express each variable in terms of how many standard deviations it deviates from the average for that variable. This puts all variables on a comparable scale, regardless of their normed frequency. Standardization is not applicable to ordinal or binary data, because it relies on the concept of a mean and standard deviation, which are not meaningful or appropriate for these types of data.

Z-Score Standardization process:

1. Calculate the mean and standard deviation for each variable's count across all texts.
2. Subtract the mean from each variable's count in a specific text.
3. Divide this difference by the standard deviation.

Standardization can be computed in SAS using the code below:

```
PROC STANDARD DATA=<dataset name> MEAN=0 STD=1 OUT=standardized;
   VAR v0001 – v1000;
RUN;
```

In R, this can be accomplished using the following code:

```
variables_to_standardize <- data[, paste0("v", sprintf("%04d", 1:1000))]
standardized_variables <- scale(variables_to_standardize)
z_data <- data
z_data[, paste0("v", sprintf("%04d", 1:1000))] <- standardized_variables
```

Factor scores can be calculated in SAS using the code below; this assumes that the values have been standardized, and that the variables loading on the positive pole of factor 1 are v0001, v0034, v0090 and v0709, and on the negative pole are v0202, v0766, and v0800:

```
DATA scores;
   SET standardized;
   fac1 = (v0001 + v0034 + v0090 + v0709) – (v0202 + v0766 + v0800) ;
RUN;
```

The following will calculate the same factor scores in R:

```
z_data$fac1 <- (z_data$v0001 + z_data$v0034 + z_data$v0090 + z_data$v0709) -
(z_data$v0202 + z_data$v0766 + z_data$v0800)
```

If the dataset contains metadata, i.e. information about register, time period, or source, for instance, ANOVAs (analyses of variance) can be computed to reveal relationships between the factors and these variables. An ANOVA typically includes two indices. The F-test determines

whether there are significant differences between the means of groups. The R^2 or coefficient of determination measures the proportion of variance in the data that can be explained by the metadata variables, quantifying the strength of these relationships.

The following SAS code will compute an ANOVA for factor 1 having *year* as the classification (metadata) variable:

```
PROC GLM DATA=scores;
        CLASS year ;
        MODEL fac1 = year ;
        MEANS year ;
RUN;
```

Similarly, in R:

```
model <- lm(fac1 ~ year, data = z_data)
anova_results <- anova(model)
print(anova_results)
```