1 The WORDFISH Algorithm

WORDFISH is a word scaling algorithm to estimate political positions from text documents on a single dimension (Slapin & Proksch 2008). It is implemented as an R code and available at www.wordfish.org.

1.1 The Algorithm

WORDFISH analyzes word frequencies of text documents and assumes the frequencies are generated by a Poisson process. This particular distribution is chosen because of its estimation simplicity: it only has one parameter, $\lambda$, which is both the mean and the variance. This assumption means that the number of times an actor $i$ mentions word $j$ is drawn from a Poisson distribution. The functional form of the model is as follows:

$$ y_{ij} \sim \text{Poisson}(\lambda_{ij}) $$

$$ \lambda_{ij} = \exp(\alpha_i + \psi_j + \beta_j \ast \omega_i) $$

where $y_{ij}$ is the count of word $j$ in actor $i$’s document (e.g. manifesto, speech, etc.), $\alpha$ is a set of actor fixed effects, $\psi$ is a set of word fixed effects, $\beta$ is an estimate of a word specific weight capturing the importance of word $j$ in discriminating between positions, and $\omega$ is the estimate of actor $i$’s position. Word fixed effects capture the fact that some words are used much more often than other words by all actors. The actor fixed effects control for the possibility that some actors write or talk more. The parameters of interest are the $\omega$’s, the position of the actors, and the $\beta$’s because they allow us to analyze which words differentiate between party positions.

1.2 Time-Series Estimation

The same model can be used to compare documents over time under the assumption that the word usage remains constant. Thus, a manifesto from party A at time $t + 1$ is simply treated as a new document in the analysis and it is assumed to be unrelated to party A’s manifesto at time $t$. The parametric model is the same:

$$ y_{ijt} \sim \text{Poisson}(\lambda_{ijt}) $$

$$ \lambda_{ijt} = \exp(\alpha_{it} + \psi_j + \beta_j \ast \omega_{it}) $$

where $y_{ijt}$ is the count of word $j$ in party $i$’s document at time $t$ (e.g. election manifesto), $\alpha$ is a set of party-election year fixed effects, $\psi$ is a set of word fixed
effects, $\beta$ is an estimate of a weight for word $j$, and $\omega$ is the estimate of party $i$’s position in election year $t$. Therefore it is indexing one specific election manifesto.

1.3 Estimation

WORDFISH uses an expectation maximization (EM) algorithm to retrieve maximum likelihood estimates for all parameters. The EM algorithm is an iterative procedure to compute maximum likelihood estimates for latent variables. The E step involves calculating the expectation of the latent variable as if it were observed. The M step then maximizes the log-likelihood conditional on the expectation. The implementation of this algorithm entails several steps:

**Step 1: Calculate starting values.**

Starting values are obtained for word fixed effects ($\psi$) by calculating the logged mean count of each word. For the party fixed effects ($\alpha$) the logged ratio of the mean word count of each document relative to the first document in our dataset is used. The starting values are set relative to the first document because this party-fixed effect is set to zero during the estimation in order to identify the model. To obtain starting values for word weights ($\beta$) and party positions ($\omega$) from the word frequencies, starting values for the word and party fixed effects are subtracted from the logged word frequencies. Then the left and right-singular vectors from a singular value decomposition of this matrix are used as starting values for $\omega$ and $\beta$.

**Step 2: Estimate party parameters.**

Party (document) parameters ($\omega$ and $\alpha$) are estimated conditional on the expectation for the word parameters. In the first iteration, the expectation of those word parameters equal their starting values calculated in step 1. The following log-likelihood is maximized for each document/party $i$:

$$\sum_{j=1}^{m} \left( -\lambda_{ij} + \ln(\lambda_{ij}) \times y_{ij} \right),$$

where

$$\lambda_{ij} = \exp(\alpha_i + \psi_j^{\text{start}} + \beta_j^{\text{start}} \times \omega_i).$$

We use $\omega_i^{\text{start}}$ and $\alpha_i^{\text{start}}$ as starting values in the maximization stage. To identify the model, in addition to setting $\alpha_1$ to zero, the mean of all party/document positions is set to zero and the standard deviation of those to one.
Step 3: Estimate word parameters.

In the next step, word parameters ($\psi$ and $\beta$) are estimated conditional on the expectation for the party parameters, which are obtained in step 2. For each unique word $j$, the following log-likelihood is maximized:

$$\sum_{i=1}^{n} (-\lambda_{ij} + \ln(\lambda_{ij}) * y_{ij}) ,$$

where

$$\lambda_{ij} = \exp(\alpha_i^{step2} + \psi_j + \beta_j * \omega_i^{step2}).$$

Included in this log-likelihood is a relatively diffuse word-specific prior in order to prevent words from carrying infinite weight. The prior belief is that $\beta$'s are distributed normally with mean of zero and standard deviation $\sigma$. This reduces the weight given to words that are mentioned very infrequently (e.g. in only one document) which might otherwise discriminate perfectly.\(^1\)

Step 4: Calculate log-likelihood.

The log-likelihood of the model is the sum of the individual word log-likelihoods from step 3, which are themselves calculated conditional upon the party log-likelihoods from step 2:

$$\sum_{j}^{m} \sum_{it=1}^{n} (-\lambda_{ijt} + \ln(\lambda_{ijt}) * y_{ijt}) .$$

Step 5: Repeat steps 2-4 until convergence.

Using the new expectations for the word parameters, party parameters are re-estimated (step 2). Then, using those expectations, word parameters are re-estimated (step 3). This process is repeated until an acceptable level of convergence, measured as the sum of the differences of individual log-likelihoods from step 4 (relative to the total log-likelihood) between the current and the previous iteration, is reached.

\(^1\)This means that in the algorithm, the actual log-likelihood for each word is as follows:

$$\sum_{i=1}^{n} (-\lambda_{ij} + \ln(\lambda_{ij}) * y_{ij}) - (\beta_j^2 / 2 * \sigma^2).$$


2 How to run WORDFISH in \textit{R} (Version 1.3)

The WORDFISH algorithm is implemented as a function in \textit{R}. The estimation involves the following steps: (1) document processing, (2) creation of a word count dataset, (3) setting options, (4) estimation until convergence, (5) diagnostics and possible re-estimation, and (6) generating uncertainty estimates for the parameters of interest.

2.1 Document Processing

Document processing is essential and possibly the most arduous task in the estimation process. First, researchers should pre-define the sets of texts to be analyzed. Second, these texts need to be processed and all unnecessary information must be removed. Third, the spelling needs to be checked in all texts.

1. Choosing documents & policy dimension

   The selection of texts will depend on what kind of policy dimension should be analyzed. For instance, if a researcher is interested in comparing foreign policy statements of parties in country \textit{X}, then only such texts should be included in the analysis. On the other hand, if the research question is to determine a general ideological position using all aspects of policy (e.g. left-right), then the analysis should potentially be conducted using all parts of an election manifesto, assuming that such documents are encyclopedic statements of policy positions.

2. Removal of unnecessary information

   Some texts include text data that researchers may prefer to remove prior to the estimation. Examples include the listing of speakers’ or parties’ names, self-reference of party names, headers and footers, enumeration, bullets, section headings, etc. This can either be done manually or with the help of pattern-matching using customized PERL or PYTHON scripts.

3. Spell check

   Finally, researchers should also ensure that the spelling of words is consistent across documents.

1. Choosing documents & policy dimension

   The selection of texts will depend on what kind of policy dimension should be analyzed. For instance, if a researcher is interested in comparing foreign policy statements of parties in country \textit{X}, then only such texts should be included in the analysis. On the other hand, if the research question is to determine a general ideological position using all aspects of policy (e.g. left-right), then the analysis should potentially be conducted using all parts of an election manifesto, assuming that such documents are encyclopedic statements of policy positions.

2. Removal of unnecessary information

   Some texts include text data that researchers may prefer to remove prior to the estimation. Examples include the listing of speakers’ or parties’ names, self-reference of party names, headers and footers, enumeration, bullets, section headings, etc. This can either be done manually or with the help of pattern-matching using customized PERL or PYTHON scripts.

3. Spell check

   Finally, researchers should also ensure that the spelling of words is consistent across documents.
2.2 Creating a term-document matrix

After document processing, a word count dataset will be generated. This can be done using any available word count programs. In R, the text mining package TM offers an easy way to generate a term-document matrix (Feinerer, Hornik & Meyer 2008). Alternatively, we can recommend Will Lowe’s programs, including YOSHIKODER (Lowe 2007b) and JFREQ (Lowe 2007a).

1. Stemming words

A stemmer algorithm removes morphological and inflexional endings from words and returns the stemmed words. The advantage is that essentially similar words will be captured as one. Moreover, the term-document matrix will have fewer unique words if words are stemmed, thus making the estimation more efficient. For instance, the stemmer would reduce the words “fishing”, “fisher”, and “fished” to the root word “fish”. The is no clear rule whether to use a stemmer and the decision will depend on the data to be analyzed. The advantage is that essentially similar words will be captured as one. Moreover, the word count matrix will have fewer unique words if documents are stemmed, thus making the estimation more efficient. A potential disadvantage is that certain compound words might be reduced to a stem thus meaning that information is lost. There is a trade-off and researchers should possibly consider both routes in the estimation. Both the TM package and JFREQ allow the application of stemming algorithms.

2. Other options

Unless there are strong reasons not to do so, we recommend using the lowercase option, which transforms all words into lowercase. In addition, we also suggest using the number removal option. Additionally, very common words (stopwords) can be removed unless there are theoretical reasons not to do so.

3. The term-document matrix

Please refer to the manual of the TM package or JFREQ for instructions on how to create the term-document matrix. A typical code in R for the TM package would include the following steps:

```r
# DEFINE DIRECTORY THAT CONTAINS UTF-8 TEXT FILES
directory <- "/*/Volumes/manifestos/"

# LOAD TEXT DOC COLLECTION (Here:German)
textcorpus <- Corpus(DirSource(directory), readerControl = list
```
# EXTRACT DOCUMENT NAMES FOR TRACKING PURPOSES

docnames <- list.files(directory)
for (i in 1:length(textcorpus)){
    Author(textcorpus[[i]])<-docnames[i]
}

# GENERATE TERM-DOCUMENT MATRIX

text.corpus.format<-textcorpus
# MAKE EVERYTHING LOWERCASE

text.corpus.format<-tmMap(text.corpus.format,tmTolower)
# REMOVE NUMBERS

text.corpus.format<-tmMap(text.corpus.format,removeNumbers)
# CREATE TERM-DOC MATRIX

wordfreqmatrix <-TermDocMatrix(text.corpus.format)
# CONVERT & TRANSPOSE WORD COUNT MATRIX FOR USE WITH WORDFISH

wcdata<-as.matrix(wordfreqmatrix)
rownames(wcdata)<-lapply(text.corpus.format,Author)
wdata<-t(wcdata)

2.3 Choosing the word sample

While WORDFISH can, in theory, be run on the entire word count data matrix, we recommend considering using only a subsample of words unless word usage has remained relatively constant. Analogous to roll call analysis, where near unanimous votes are usually not used for the analysis, words that are used only in one document will produce a word weight that carries - in theory - infinite weight. To cope with such a situation, WORDFISH includes a diffuse prior on the distribution of $\beta$. Nevertheless, we recommend that words used very infrequently should be removed.

2.4 Running WORDFISH

There are three steps involved in running WORDFISH: loading the data, setting WORDFISH options, and running the code until convergence has been achieved.

1. Loading the data

If you use the TM package to create the term-document matrix, please follow the sample steps outlined above and proceed to step 2. If you have created the word count matrix outside of R, you should load the word count data. The matrix should look as follows:
Load the data:

```r
wcdata<-read.csv("wordcount.csv")
```

2. Setting the options

WORDFISH allows two identification strategies. The first one sets the mean of the positions ($\omega$) to zero and the standard deviation to one. This is the default identification strategy. If you run this version, it also necessary to indicate two documents, the first of which will have a more negative $\omega$ than the second. This requirement ensures global identification of the model. It is recommended that you choose documents that you think are likely to be very different in word usage. In other words, when estimating an ideological dimension, choose texts you believe are likely represent the opposite ends of the political spectrum.

A WORDFISH estimation can then be run simply with the following command:

```r
results<-wordfish(input=wcdata,dir=c(1,5))
```

In this example, `wordfreq` is the input term-document matrix and `dir` indicates which two documents (i.e. columns) are used for global identification purposes. The full list of options include:
input
The term-document matrix. Words in rows, texts in columns.

wordsincol
If TRUE, removes first column of input matrix prior to estimation. Use this option, if the term-document matrix does not include words as row names. Default is FALSE.

fixtwo
If TRUE, identifies the model by fixing two document positions at set values; these must be given by fixdoc. If FALSE identifies the model by setting the mean of positions to 0 and the standard deviation to 1. Default is FALSE.

dir
Required for default identification (i.e. if fixtwo=FALSE). Two documents are selected, the first of which is constrained to have a more negative value than the second. Needs to be a vector with two columns. For example, dir=c(1,5) would constrain \( \omega_1 \) to be smaller than \( \omega_5 \). Try to choose documents whose documents you think are likely to be very dissimilar (e.g. for Germany, you could choose a document from the FDP and a document from the Greens). Must be provided by user, no default values.

fixdoc
Required if fixtwo=TRUE. Vector of two documents and their fixed positions (four elements). This vector must be provided by the user, there are no default values. The first two vector elements indicate the documents in the matrix, the last two the fixed values. We recommend choosing values between 0 and 1. E.g. If the first and fifth document in the matrix are to be constrained to have omega values of +0.5 and +1, then the option should look like this: fixdoc=c(1,5,0.5,1). The two values cannot be identical.

tol
Tolerance criteria for convergence, default is 1e-7.

sigma
Variance parameter to constrain beta, default is 3.

boots
If TRUE, runs parametric bootstrap (increased computation time!). Default is FALSE.

nsim
Number of bootstrap trials if bootstrap is turned on, default is 500.

writeout
If TRUE, writes three output files to working directory.

output
Name of output file if writeout=TRUE. Default is the prefix ‘wordfish_output’.

3. Convergence
As a convergence criterion, Wordfish calculates the sum of the differences in word log-likelihoods between the current and previous iteration relative to the total log-likelihood of the current iteration. The code stops once this difference is smaller than
a tolerance value. The default convergence criterion tol is set to 1e-7. The starting value calculation is optimized for the default identification strategy (\( \omega \) mean is 0, and the SD is 1). If you run the alternative identification strategy, the number of iterations until convergence is likely to increase.

2.5 WORDFISH output

The estimation output can easily be called from the list object results for plotting purposes or further analysis. The following output is available:

```
results$documents    Document estimates, contains \( \omega \) and \( \alpha \)
results$words        Word estimates, contains \( \beta \) and \( \psi \)
results$difflik      Differences in relative LL per iteration (convergence criterion)
results$diffomega    Average difference in positions per iteration
results$maxlik       Total LL per iteration
results$estimation   Summary of the estimation
results$ci.documents Confidence interval for positions (only if boots=T)
results$ci.words     Confidence interval for words (only if boots=T)
```

If writeout=TRUE, WORDFISH will also write three output files to the working directory: results_estimation.csv, results_parties.csv, and results_words.csv.

- **results_parties.csv**
  This file will contain the document parameters \( \omega \) (positions) in the first column and \( \alpha \) (fixed effects) in the second column.

- **results_words.csv**
  This file will contain the word parameters \( \beta \) (positions) in the first column and \( \psi \) (fixed effects) in the second column.

- **results_estimation.csv**
  The same as results$estimation. This file includes the number of unique words used, the number of documents used, the number of iterations until convergence, the final total log-likelihood, the convergence criterion, and the average difference in party positions between the previous to last and last iteration.

2.6 Diagnostics

A good start for diagnostics is the analysis of word discrimination parameters. Weights with large values mean that these words are estimated to be on the extremes of the dimension.
2.7 Uncertainty Estimates

WORDFISH produces uncertainty estimates using a parametric bootstrap. If the bootstraps option is turned on, `boots=TRUE`, WORDFISH will run 500 bootstraps. You may increase or decrease the number of bootstrap trials by setting the `nsim` option. The bootstrap procedure works as follows. The ML estimates for $\alpha$, $\omega$, $\beta$, $\psi$ are used to calculate $\lambda_{ij}$ for each pair of unique word and document (each cell in the word count matrix). WORDFISH then takes $\text{nsim}$ random draws from a Poisson distribution with the parameter $\lambda_{ij}$ for each cell, creating $\text{nsim}$ new datasets. The algorithm is run on each of these new term-document matrices and the 0.025 and the 0.975 quantiles of the estimated $\omega$ parameters constitute an approximate 95% confidence interval (it is possible to calculate confidence intervals for the remaining parameters as well). Note that this procedure can be very time consuming. If you have a very large dataset, it may take several days to estimate confidence intervals on a standard desktop computer.

References


3 Frequently Asked Questions

3.1 Languages

QUESTION: Does WORDFISH work only in English?

No, you can use WORDFISH for any language as long as you can identify words. Typically, word counters identify words by white space separation. This is impossible, for instance, in Japanese, but there are work-arounds (so-called tokenizers that identify the word bounds). WORDFISH has been successfully applied to estimate speeches in the European Parliament using three different languages (English, French, German) and the estimated positions correlate highly (Proksch & Slapin Forthcoming). We therefore have good reason to believe that the technique is robust to language choice.

QUESTION: I need to estimate positions using documents written in different languages. Is this possible?

No. WORDFISH compares relative word usage, thus it is not possible to compare different languages. You either need to translate the documents into one language (careful: translator effects!), or resort to a different type of content analysis.

QUESTION: I want to compare and analyze policy positions in country A and the country B. When I run the estimation separately for the two countries, the parties line up according to what I would have expected. However, when I run a pooled analysis, all parties from country A are to the left and all parties from country B are to the right. How can this be?

It could be that you have words in the analysis that are used only in the political debate in country A, and not in country B (or vice versa). You might want to include only words that are used in both countries. You may also wish to consider whether the political debate in both countries is truly comparable.

3.2 Document processing and word count data

QUESTION: I only have very few documents in the analysis, or the documents I have are very short. Can I use WORDFISH?

If you are only estimating few positions, there might be problems with regard to the stability of the word parameters, which are estimated using only very few
data points. Similarly, if the documents are very short (e.g. a few sentences),
the position estimates will be based on too few words. We recommend that you
either get more data or resort to alternative ways of measuring the positions.

**QUESTION:** I stemmed the words before creating the word count dataset.
After looking at the word parameter output, I am afraid I can no
longer tell the meaning of the stemmed words. What should I do?

There are advantages and disadvantages when stemming words. Clearly, the
ability to perform ex-post diagnostics decreases, as it becomes difficult to iden-
tify the full word meanings. On the other hand, similar words are treated as
one (e.g. “peace” and “peaceful”) and this should improve the estimation. It
is a trade-off and researchers may wish to consider both routes.

**QUESTION:** I am skeptical that single words can actually discriminate
between parties. How would the difference between ‘we want lower
taxes’ (party A) and ‘we want higher taxes’ (party B) be captured,
if the word ‘taxes’ has the exact same frequency in party A and
B’s document? Isn’t it possible to compare word pairs (e.g. “lower
taxes” and “higher taxes”)?

There are two aspects to this question. The first is whether unigrams - single
words - are able to capture differences. If there are two parties and one mentions
the phrase “we want lower taxes” frequently and the other party mentions “we
want higher taxes” then the words “lower” and “higher” will discriminate and
place the parties apart. If you like to use bigrams - word pairs - you simply
need to count word pairs instead of single words. WORDFISH could then scale
those bigram frequencies just as it can unigram frequencies. As a drawback,
the data matrix will be larger and computation time will significantly increase
as a result.

### 3.3 Diagnostics

**QUESTION:** I ran WORDFISH and the words with the largest weights
seem trivial and not politically meaningful. Should I worry about
the validity of the estimated dimension?

Not necessarily. There are potentially thousands of unique words in the anal-
ysis. One word will thus not have a dominant influence on the estimated
positions, even if its discrimination parameter $\beta$ is large (remember that those
are constrained by the option sigma in the WORDFISH function). A large
discrimination parameter simply means that the word is at the extreme of the estimated dimension. If there are many trivial words with large weights, it would be worth checking the spelling and the word usage in the documents. Note that just looking at extreme $\beta$ values might miss important words that are used by parties on the “left” and on the “right” equally.

### 3.4 Dimensionality

**QUESTION:** I want multidimensional positions. Can WORDFISH do this?

Yes and no, depending on your conception of the policy space. WORDFISH does not estimate multiple dimensions, only a single dimension, but it does allow the estimation of different dimensions if you use different text sources. For instance, if your interest is in estimating positions of presidential candidates on foreign policy and economic policy, then you could estimate separate positions using foreign policy speeches only on the one hand and economic policy speeches on the other hand. Similarly, if you are analyzing party manifestos, you might want to use only sections of the manifesto pertaining to a particular policy area. If you do this, we recommend including your definition of the dimension in the analysis. Different researchers might come to different conclusions about what a particular dimension should include.

**QUESTION:** I want positions on a particular pre-defined policy area. Why shouldn’t I use WORDSCORES instead of WORDFISH?

You very well could. The choice of technique really depends on your particular research question and different content analysis techniques have different advantages. WORDSCORES (Laver, Benoit & Garry 2003) is a technique to estimate positions on an *ex ante* defined dimension, which is anchored by reference texts, whereas WORDFISH scales the word counts and retrieves a dimension that explains the most variance in word usage. These are conceptually two different approaches, even though both use word counts. Thus, if you have exogenously given reference texts that define the space you want to estimate, we recommend using WORDSCORES rather than WORDFISH. On the other hand, if you are interested in explaining the most salient political speech dimension, you might want to use WORDFISH.
3.5 Uncertainty Estimates

QUESTION: I ran WORDFISH, got estimates, but when I run the bootstrap the position estimate is outside the 95%-confidence interval. How can this happen?

You either have very few documents, making the word weight estimation unstable, or you have a substantial number of words in your documents that are used very infrequently. We recommend not using words found only in one document.