Chapter 3

Dimensionality Reduction

3.1 Introduction

In the previous chapters, semantic similarity calculations have been carried out using the words' original feature space, which usually contains a large number of highly correlated features. The goal of a dimensionality reduction – also called factorization – is to find a smaller number of uncorrelated or lowly correlated dimensions (factors). There are two reasons for applying such a transformation to the data:

- When the feature space is large, similarity calculations often become computationally expensive or even impossible. A dimensionality reduction reduces the feature space to a much smaller number of dimensions, so that computations become tractable again.
- A dimensionality reduction is able to discover latent structure present in the data. This way, a dimensionality reduction is able to generalize over individual data samples. By classifying the data according to the latent structure and not according to the individual features, a dimensionality reduction is able to overcome data sparseness and noise.

One of the most famous dimensionality reduction methods for text processing is latent semantic analysis (LSA). LSA allegedly finds 'latent semantic dimensions', according to which nouns and documents can be represented more efficiently. In

the subsequent section, we will first have a look at LSA and its underlying singular value decomposition. Next, we will examine non-negative matrix factorization, a dimensionality reduction algorithm that overcomes some of the problems linked to LSA.

3.2 Latent semantic analysis

3.2.1 Introduction

Latent semantic analysis (Landauer and Dumais, 1997; Landauer, Foltz, and Laham, 1998) models the meaning of words and documents by projecting them into a vector space of reduced dimensionality; the reduced vector space is built up by applying singular value decomposition (SVD) – a well known linear algebraic method – to a simple term-by-document frequency matrix $\mathbf{A}$. The resulting lower dimensional matrix $\hat{\mathbf{A}}$ is the best possible fit in a least squares sense (minimization of the Frobenius norm; equation 3.1).

$$\arg\min_{\hat{\mathbf{A}}} \| \mathbf{A} - \hat{\mathbf{A}} \|_F \tag{3.1}$$

By enforcing a lower number of dimensions, the algorithm is forced to make generalizations over the simple frequency data. Co-occurring terms are mapped to the same dimensions; terms that do not co-occur are mapped to different dimensions.

In the next section, we have a closer look at the principles and mathematics behind SVD. Next, some example SVD's are provided in order to exemplify their generalization capacity. We conclude with a discussion of the drawbacks linked to LSA.

3.2.2 Singular value decomposition

While rooted in linear algebra, singular value decomposition has proven to be a useful tool in statistical applications. It is closely akin to statistical methods such as principal components analysis, and has been used as a versatile dimensionality reduction technique in different scientific fields, such as image recognition, signal processing (Deprettere, 1988), and information retrieval. SVD stems from a well known theorem in linear algebra: a rectangular matrix can be decomposed into

three other matrices of specific forms, so that the product of these three matrices is equal to the original matrix:[1]

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times z}\ \Sigma_{z\times z}\ (\mathbf{V}_{n\times z})^T \tag{3.2}$$

where $z = min(m, n)$. A graphical representation of SVD (with $z = n$) is given in figure 3.1.

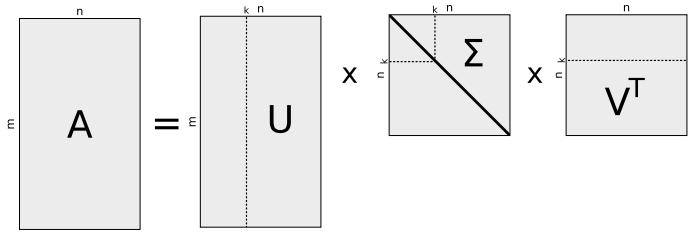

Figure 3.1: Graphical representation of SVD

Matrix $\mathbf{A}$ is the original matrix of size $m \times n$. Matrix $\mathbf{U}$ is an $m \times z$ matrix that contains newly derived vectors called left-singular vectors. Matrix $\mathbf{V}^T$ denotes the transpose of matrix $\mathbf{V}$, an $n \times z$ matrix of derived vectors called right-singular vectors. The third matrix Σ is a $z \times z$ square diagonal matrix (i.e. a square matrix with non-zero entries only along the diagonal); Σ contains derived constants called singular values. A key property of the derived vectors is that all dimensions are orthogonal (i.e. linearly independent) to each other, so that each dimension is uncorrelated to the others.

The singular value decomposition can be interpreted as a method that rotates the axes of the n-dimensional space in such a way that the largest variation is captured by the leading dimensions. The diagonal matrix Σ contains the singular values sorted in descending order. Each singular value represents the amount of variance that is captured by a particular dimension. The left-singular and right-singular vector linked to the highest singular value represent the most important dimension in the data (i.e. the dimension that explains the most variance of the matrix); the singular vectors linked to the second highest value represent the second most important dimension (orthogonal to the first one), and so on. Typically, one uses only the first $k \ll z$ dimensions, stripping off the remaining singular values and

[1] The singular value decomposition that is presented here is called the 'thin' or 'reduced' SVD. In applications such as LSA, it is unusual to compute the full SVD; the reduced version is faster to compute and more economical in storage, and it provides sufficient information for statistical applications.

singular vectors. If one or more of the least significant singular values are omitted, then the reconstructed matrix will be the best possible least-squares approximation of the original matrix in the lower dimensional space. Intuitively, SVD is able to transform the original matrix – with an abundance of overlapping dimensions – into a new, many times smaller matrix that is able to describe the data in terms of its principal components. Due to this dimension reduction, a more succinct and more general representation of the data is obtained. Redundancy is filtered out, and data sparseness is reduced.

The calculation of SVD involves iteratively solving a number of eigenvalue problems. A thorough understanding of the algorithm's computational details requires a firm background in linear algebra, and explaining all the mathematical nuts and bolts is well beyond the scope of this thesis. Suffice it to say that there are a number of programs available that can handle the kind of large-scale singular value decompositions necessary for linguistic data sets. In this research, SVDPACK (Berry, 1992) has been used. SVDPACK is a program that is able to handle sparse matrices quickly and efficiently (depending on the number of singular values one wants to retain).

3.2.3 Examples

Consider two documents, one about *Belgium* (B) and one about the *Netherlands* (NL).

- Belgium is a kingdom in the middle of Europe, and **Brussels** is its capital. **Brussels** has a Dutch-speaking and a French-speaking university, but the largest student city is **Leuven**. **Leuven** has 31,000 students.
- The Netherlands is a country in Western Europe, located next to the North Sea. The Netherlands's capital is **Amsterdam**. **Amsterdam** has two universities. **Groningen** is another important student city. In **Groningen**, there are 37,000 students.

As we have seen in the previous chapter, these documents can easily be transformed into a term-document matrix, in which each document is represented by a column vector. Each element in the column vector corresponds to the frequency of a particular term (in this case cities) in the document. Similarly, each element on the row vector indicates how often a term appears in a particular document. The resulting matrix, together with its singular value decomposition, is given in figure 3.2.

$$\mathbf{A}\begin{bmatrix} & B & NL \\ Groningen & 0 & 2 \\ Leuven & 2 & 0 \\ Amsterdam & 0 & 2 \\ Brussel & 2 & 0 \end{bmatrix} = \mathbf{U}\begin{bmatrix} 0.00 & 0.71 \\ -0.71 & 0.00 \\ 0.00 & 0.71 \\ -0.71 & 0.00 \end{bmatrix} \Sigma \begin{bmatrix} 2.83 & 0 \\ 0 & 2.83 \end{bmatrix} \mathbf{V}^T \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Figure 3.2: Singular value decomposition of a term-document matrix

The original matrix **A** is decomposed into three other matrices **U**, Σ and $\mathbf{V}^T$. The singular values in Σ show that two equally important dimensions are found; furthermore, the left- and right-singular vectors show that the frequencies are evenly divided among terms as well as among documents.

Figure 3.3 shows what happens when we add another document about Belgium, with a slightly different frequency distribution of terms: the Belgian dimension becomes the most important (i.e. captures the most variation, 2.92), while the Dutch dimension remains the same (2.83). The third dimension (0.68) captures the remaining variation (the fact that the third document only talks about Brussels).

If we now truncate the SVD by keeping only the two most important dimensions, and then reconstruct our original matrix, we get matrix $\hat{\mathbf{A}}$, which is the best possible reconstruction from only two dimensions. Note that matrix $\hat{\mathbf{A}}$ resembles matrix **A**, except for the numbers of the third document: instead of assigning all frequency mass to the term *Brussel*, the mass is almost evenly divided among the Belgian terms *Brussel* and *Leuven*. When keeping only two dimensions, the SVD 'guesses' the best possible distribution. This is an example of how the technique is used to obtain a more succinct model that is able to generalize among the data.

Below, we describe a more elaborate example, illustrating once again the generalization capacity of a singular value decomposition. Figure 3.4 represents another term-by-document matrix, containing Dutch nouns that are related to two distinct semantic topics. The nouns *tulp* 'tulip', *tuin* 'garden', and *park* 'park' are related to the topic of gardening. The nouns *ei* 'egg', *kaas* 'cheese', and *boter* 'butter' all relate to the topic of food. The noun *bloem* is an ambiguous word in

$$A \begin{bmatrix} & B & NL & B \\ Groningen & 0 & 2 & 0 \\ Leuven & 2 & 0 & 0 \\ Amsterdam & 0 & 2 & 0 \\ Brussel & 2 & 0 & 1 \end{bmatrix} =$$

$$U \begin{bmatrix} 0.00 & -0.71 & \cancel{0.00} \\ -0.66 & 0.00 & \cancel{0.75} \\ 0.00 & -0.71 & \cancel{0.00} \\ -0.75 & 0.00 & \cancel{-0.66} \end{bmatrix} \Sigma \begin{bmatrix} 2.92 & 0.00 & \cancel{0.00} \\ 0.00 & 2.83 & \cancel{0.00} \\ \cancel{0.00} & \cancel{0.00} & \cancel{0.68} \end{bmatrix}$$

$$V^T \begin{bmatrix} -0.97 & 0.00 & 0.26 \\ 0.00 & -1.00 & 0.00 \\ \cancel{-0.26} & \cancel{0.00} & \cancel{-0.97} \end{bmatrix} \cong \hat{A} \begin{bmatrix} 0.00 & 2.00 & 0.00 \\ 1.87 & 0.00 & 0.50 \\ 0.00 & 2.00 & 0.00 \\ 2.12 & 0.00 & 0.56 \end{bmatrix}$$

Figure 3.3: Truncated singular value decomposition

Dutch, meaning 'flower' (related to the gardening topic) as well as 'flour' (related to the food topic). Figure 3.4 represents the distribution of the seven nouns across five different documents. The complete SVD (matrices $\mathbf{U}$, Σ and $\mathbf{V}^T$) is given in figures 3.5 to 3.7.

$$\mathbf{A} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{tulp} & 1 & 0 & 1 & 0 & 0 \\ \text{tuin} & 1 & 1 & 0 & 0 & 0 \\ \text{park} & 0 & 1 & 0 & 0 & 0 \\ \text{ei} & 0 & 0 & 0 & 1 & 1 \\ \text{kaas} & 0 & 0 & 0 & 1 & 0 \\ \text{boter} & 0 & 0 & 0 & 1 & 1 \\ \text{bloem} & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.4: Term-by-document matrix $\mathbf{A}$

We can now easily project the terms and documents of the original matrix into a space of reduced dimensionality; in the following example, we will retain two dimensions. Matrix $\mathbf{B}$ gives the terms after a reduction to two dimensions, scaled

$$\mathbf{U} = \begin{bmatrix} & \text{dim1} & \text{dim2} & \text{dim3} & \text{dim4} & \text{dim5} \\ \text{tulp} & -0.21 & 0.52 & -0.48 & -0.58 & 0.35 \\ \text{tuin} & -0.22 & 0.60 & 0.46 & 0 & -0.40 \\ \text{park} & -0.05 & 0.22 & 0.65 & 0 & 0.55 \\ \text{ei} & -0.56 & -0.30 & 0.06 & 0 & 0.20 \\ \text{kaas} & -0.26 & -0.22 & 0.17 & -0.58 & -0.55 \\ \text{boter} & -0.56 & -0.30 & 0.06 & 0 & 0.20 \\ \text{bloem} & -0.47 & 0.30 & -0.31 & 0.58 & -0.20 \end{bmatrix}$$

Figure 3.5: The left-singular matrix **U**

$$\Sigma = \begin{bmatrix} 2.30 & 0 & 0 & 0 & 0 \\ 0 & 1.93 & 0 & 0 & 0 \\ 0 & 0 & 1.30 & 0 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 0.52 \end{bmatrix}$$

Figure 3.6: The square diagonal matrix Σ

$$\mathbf{V}^T = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{dim1} & -0.39 & -0.12 & -0.09 & -0.60 & -0.69 \\ \text{dim2} & 0.74 & 0.43 & 0.27 & -0.43 & -0.16 \\ \text{dim3} & -0.26 & 0.85 & -0.37 & 0.22 & -0.15 \\ \text{dim4} & 0 & 0 & -0.58 & -0.58 & 0.58 \\ \text{dim5} & -0.49 & 0.28 & 0.67 & -0.28 & 0.39 \end{bmatrix}$$

Figure 3.7: The right-singular matrix $\mathbf{V^T}$

with the singular values. The matrix is obtained by multiplying a slice of matrix **U** ($\mathbf{U}_{7\times2}$) with a slice of matrix Σ ($\Sigma_{2\times2}$). Matrix **B** is given in figure 3.8. The term vectors – normalized to vector length – are represented graphically in figure 3.9.

In figure 3.9, we can clearly distinguish the two different topics: the 'garden' topic (with *tulp*, *tuin* and *park*) in the second quadrant, and the 'food' topic (with *ei*, *kaas* and *boter*) in the third quadrant. Note that the terms *tulp* and *park* do not appear together in the same document in the original matrix; in the reduced two-dimensional SVD space, however, they are clearly closely related. This is again an

$$\mathbf{B} = \begin{bmatrix} & \text{dim1} & \text{dim2} \\ \text{tulp} & -0.48 & 1.01 \\ \text{tuin} & -0.51 & 1.16 \\ \text{park} & -0.12 & 0.43 \\ \text{ei} & -1.28 & -0.58 \\ \text{kaas} & -0.60 & -0.43 \\ \text{boter} & -1.28 & -0.58 \\ \text{bloem} & -1.08 & 0.58 \end{bmatrix}$$

Figure 3.8: Matrix **B**, the multiplication of $\mathbf{U}_{7\times2}$ and $\Sigma_{2\times2}$

example of the generalization capability of the SVD. Also note that the ambiguous word *bloem*, related to both the 'garden' topic and the 'food' topic, ends up in between them.

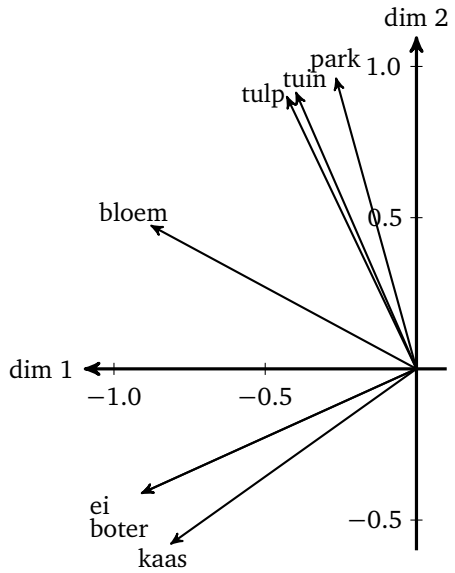

Figure 3.9: A graphical representation of the term vectors in the reduced dimensional space

Similarly, we obtain matrix **C** – the projection of the documents into the two-dimensional reduced vector space – by multiplying $\Sigma_{2\times2}$ with $\mathbf{V}^T_{2\times5}$. Matrix **C** is given in figure 3.10. The document vectors – again normalized to vector length – are represented graphically in figure 3.11.

$$\mathbf{C} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{dim1} & -0.89 & -0.27 & -0.21 & -1.37 & -1.58 \\ \text{dim2} & 1.42 & 0.82 & 0.52 & -0.82 & -0.30 \end{bmatrix}$$

Figure 3.10: Matrix **C**, the multiplication of $\Sigma_{2\times2}$ and $\mathbf{V}^T_{2\times5}$

Again, we see the same topic division among the document vectors. Documents d_1, d_2 and d_3 are grouped together in the second quadrant, and documents d_4 and d_5 appear together in the third quadrant. Note again that documents d_2 and d_3 appear closely together, although they do not share any terms in the original term-document matrix.

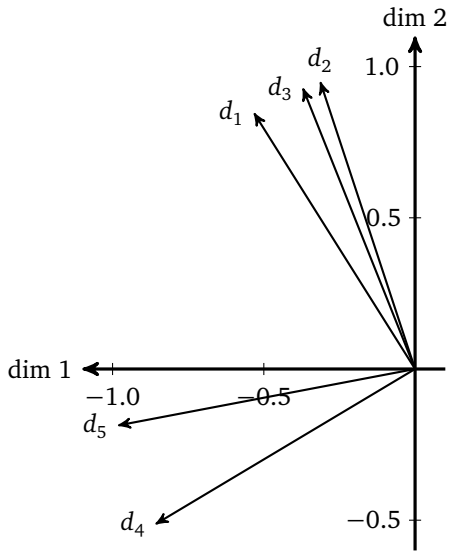

Figure 3.11: A graphical representation of the document vectors in the reduced dimensional space

3.2.4 Drawbacks

LSA suffers from a number of drawbacks, that have been regularly noted in the literature. (Manning and Schütze, 2000, p. 565)

The first major drawback is that a singular value decomposition (or, more correctly, its probabilistic interpretation) assumes normally distributed data. As Manning and Schütze (2000) note, a normal distribution is inappropriate for frequency count data, such as textual co-occurrence data. There are other distributions – such as a Poisson distribution – that are better suited for modeling count data. As a consequence of the normality assumption, the reconstruction $\mathbf{A}'$ of the original matrix $\mathbf{A}$ may contain negative numbers, which clearly is a bad approximation for frequency counts.

A second drawback – related to the first one – is the presence of negative values in the derived dimensions themselves. The derived dimensions are said to represent actual 'latent semantic' dimensions. A particular term or document can have a positive or negative value on those dimensions. It is not clear what negative values on a semantic scale should designate. A particular term or document either is related (positive value) or is not related (zero value) to a particular topic; it seems counterintuitive to say that a particular word is negatively related to a particular topic. This intuition is confirmed by experiments. In the following section, we will present an algorithm that only allows non-negative data in its dimensionality reduction. By enforcing this constraint, the algorithm is able to find much more distinct and clear-cut semantic dimensions.

3.3 Non-negative matrix factorization

3.3.1 Introduction

In this section, we describe a dimensionality reduction technique called non-negative matrix factorization (NMF) that does not suffer from the drawbacks of LSA and its underlying singular value decomposition. Non-negative matrix factorization is a dimensionality reduction technique that has become popular in fields such as image recognition, speech recognition and machine learning. Its key idea is to impose a non-negativity constraint on the factorization. This constraint brings about a parts-based representation, because only additive and no subtractive combinations are allowed. In many cases, this constraint proves beneficial for the inductive capabilities of the dimensionality reduction: the algorithm is able to extract more clear and distinct characteristics from the data.

The difference between the parts-based induction of NMF and the holistic induction of non-constrained methods such as PCA (and the related singular value decomposition) can be illustrated with an example from facial image recognition (Lee and Seung, 1999). A famous method in facial image recognition uses so-called 'eigenfaces' (Turk and Pentland, 1991). These are a small number of prototypical faces represented by the eigenvectors that are found by applying PCA to a database of facial images. Eigenfaces may contain positive as well as negative values. A key characteristic is that they are 'holistic': an eigenface contains all kinds of facial traits, and thus represents a prototypical face. By taking a linear combination of various 'eigenfaces', a particular instance of a face may be reconstructed.

The representation that is found by NMF looks quite different: instead of finding holistic, prototypical faces, the algorithm induces particular facial traits (different kinds of eyes, noses, mouths, ...). By enforcing a non-negative constraint, the algorithm is able to build up a parts-based representation of facial images. A particular instance of a face may then be reconstructed by taking a linear combination of the different parts. The very same characteristic will also prove to be beneficial for building up semantic representations from text.

3.3.2 Theory

Non-negative matrix factorization (NMF) (Lee and Seung, 2000) is the name for a group of algorithms in which a matrix **V** is factorized into two other matrices, **W** and **H**.

$$\mathbf{V}_{n\times m} \approx \mathbf{W}_{n\times r}\mathbf{H}_{r\times m} \tag{3.3}$$

Figure 3.12 gives a graphical representation of non-negative matrix factorization.

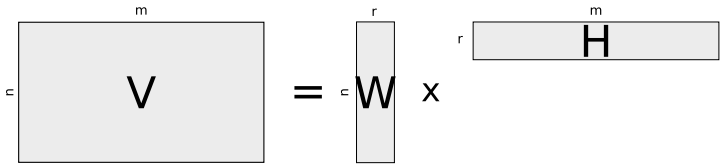

Figure 3.12: A graphical representation of non-negative matrix factorization

Typically r is much smaller than n, m so that both instances and features are expressed in terms of a few components. As mentioned above, non-negative matrix

factorization enforces the constraint that all three matrices must be non-negative, so all elements must be greater than or equal to zero.

There are two objective functions that may be used in order to quantify the quality of the approximation of the original matrix. One objective function minimizes the sum of squares (equation 3.4).

$$\min \ \| \mathbf{V} - \mathbf{WH} \|_F = \ \min \sum_i \sum_j (\mathbf{V}_{ij} - (\mathbf{WH})_{ij})^2 \tag{3.4}$$

The other one minimizes the Kullback-Leibler divergence (equation 3.5).

$$\min \ D_{\mathrm{KL}}(\mathbf{V} \parallel \mathbf{WH}) = \min \sum_i \sum_j \left(\mathbf{V}_{ij} \log \frac{\mathbf{V}_{ij}}{(\mathbf{WH})_{ij}} - \mathbf{V}_{ij} + (\mathbf{WH})_{ij} \right) \tag{3.5}$$

Practically, the factorization can be efficiently carried out through the iterative application of multiplicative update rules. The set of update rules that minimize the Euclidean distance are given in 3.6 and 3.7.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{(\mathbf{W}^T \mathbf{V})_{a\mu}}{(\mathbf{W}^T \mathbf{WH})_{a\mu}} \tag{3.6}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{(\mathbf{VH}^T)_{ia}}{(\mathbf{WHH}^T)_{ia}} \tag{3.7}$$

The set of update rules that minimize the Kullback-Leibler divergence are given in 3.8 and 3.9.

$$\mathbf{H}_{a\mu} \leftarrow \mathbf{H}_{a\mu} \frac{\sum_i \mathbf{W}_{ia} \frac{\mathbf{V}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_k \mathbf{W}_{ka}} \tag{3.8}$$

$$\mathbf{W}_{ia} \leftarrow \mathbf{W}_{ia} \frac{\sum_\mu \mathbf{H}_{a\mu} \frac{\mathbf{V}_{i\mu}}{(\mathbf{WH})_{i\mu}}}{\sum_\nu \mathbf{H}_{a\nu}} \tag{3.9}$$

Matrices $\mathbf{W}$ and $\mathbf{H}$ are randomly initialized, and the update rules are iteratively applied – alternating between them. In each iteration, the matrices $\mathbf{W}$ and $\mathbf{H}$ are suitably normalized, so that the rows of the matrices sum to 1. The algorithm stops after a fixed number of iterations, or according to some stopping criterion (the change of the objective function drops below a certain threshold). The update rules are guaranteed to converge to a local optimum. In practice, it is usually sufficient to run the NMF algorithm repeatedly in order to find the global optimum.

3.3.3 Example

In the following example, we take matrix **V** in figure 3.13 (reproduced from matrix **A** used in the SVD example on page 38), and factorize it to two dimensions using non-negative matrix factorization. As objective function, we take the Kullback-Leibler divergence (which implies the use of the update rules in 3.8 and 3.9). The globally optimal matrices **W** and **H** are represented in figures 3.14 and 3.15.[2]

$$\mathbf{V} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{tulp} & 1 & 0 & 1 & 0 & 0 \\ \text{tuin} & 1 & 1 & 0 & 0 & 0 \\ \text{park} & 0 & 1 & 0 & 0 & 0 \\ \text{ei} & 0 & 0 & 0 & 1 & 1 \\ \text{kaas} & 0 & 0 & 0 & 1 & 0 \\ \text{boter} & 0 & 0 & 0 & 1 & 1 \\ \text{bloem} & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 3.13: Term-by-document matrix **V**

Matrices **W** and **H** can be interpreted as conditional probabilities. Matrix **W** represents the probability of a word given a particular topical, 'semantic' dimension.

$$\mathbf{W} = \begin{bmatrix} & \text{dim1} & \text{dim2} \\ \text{tulp} & 0 & \frac{1}{3} \\ \text{tuin} & 0 & \frac{1}{3} \\ \text{park} & 0 & \frac{1}{6} \\ \text{ei} & \frac{1}{3} & 0 \\ \text{kaas} & \frac{1}{6} & 0 \\ \text{boter} & \frac{1}{3} & 0 \\ \text{bloem} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

Figure 3.14: Matrix **W**, containing the original nouns and a reduced number of dimensions

Matrix **H** gives the probability of a dimension given a document (in the example, each document contains one particular topic).

[2]Note once again that the update rules are guaranteed to converge to a local optimum; it might take a number of tries to find the globally optimal solution.

$$\mathbf{H} = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{dim1} & 0 & 0 & 0 & 1 & 1 \\ \text{dim2} & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Figure 3.15: Matrix **H**, containing the original documents and a reduced number of dimensions

By multiplying matrices **W** and **H**, we get matrix $\mathbf{V}'$, containing the probabilities of a word given a document.

$$\mathbf{V}' = \begin{bmatrix} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{tulp} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \text{tuin} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \text{park} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & 0 \\ \text{ei} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \text{kaas} & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{6} \\ \text{boter} & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \text{bloem} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

Figure 3.16: The reconstructed matrix $\mathbf{V}'$, the multiplication of **W** and **H**

Note that the values of (tulp, d_2), (tuin, d_3), and (park, $d_{1,3}$) – that have zeros in the original co-occurrence matrix – have received appropriate probability values in the reconstructed matrix $\mathbf{V}'$. The factorization model has made correct inferences about words related to the gardening topic appearing in sentences related to the gardening topic. Likewise, (kaas, d_5) has received an appropriate probability value. The ambiguous word *bloem* – related to the two topics – has received appropriate probability values across the whole document range.