Chapter 4
Text Preprocessing

Abstract This chapter starts the process of preparing text data for analysis. This chapter introduces the choices that can be made to cleanse text data, including tokenizing, standardizing and cleaning, removing stop words, and stemming. The chapter also covers advanced topics in text preprocessing, such as n-grams, part-of-speech tagging, and custom dictionaries. The text preprocessing decisions influence the text document representation created for analysis.

Keywords Text preprocessing · Text parsing · n-grams · POS tagging · Stemming · Lemmatization · Natural language processing · Tokens · Stop words

4.1 Introduction

By the end of the planning stage, the data should be collected, and the goal of the analysis should be well defined. After completing the planning stage, the next step is to prepare the data for analysis. Each record of the collected text data should have a unique identifier that can be used to refer to that instance. In text analytics, these instances are known as documents. A document is typically made up of many characters. The many documents make up a document collection or corpus. Characters are combined to form words or terms in a given language. These words are the focus of our analysis, although groupings of terms can also be the chosen unit of analysis, as described in this chapter. The collection of terms is sometimes called the vocabulary or dictionary. Figure 4.1 illustrates the components of our text data.

Let's consider an example of a document collection in which ten people were told to envision and describe their dog. The dog could be wearing dog clothes. Some of these people describe their own dogs, while others describe a fictional dog. The document collection is shown in Fig. 4.2.

M. Anandarajan et al., *Practical Text Analytics*, Advances in Analytics and Data Science 2, https://doi.org/10.1007/978-3-319-95663-3_4

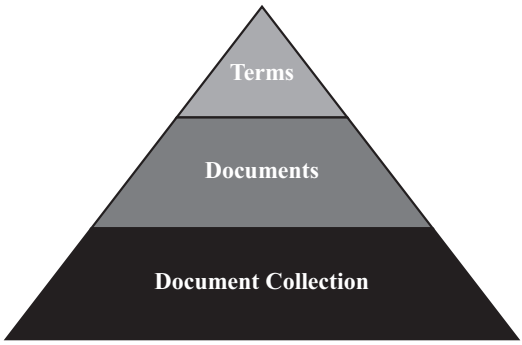


Fig. 4.1 Hierarchy of terms and documents

4.2 The Preprocessing Process

Much of the text preparation and preprocessing methods have their roots in natural language processing. Text preprocessing takes an input of raw text and returns cleansed tokens. Tokens are single words or groups of words that are tallied by their frequency and serve as the features of the analysis.

The preprocessing process includes (1) unitization and tokenization, (2) standardization and cleansing or text data cleansing, (3) stop word removal, and (4) stemming or lemmatization. The stages along the pipeline standardize the data, thereby reducing the number of dimensions in the text dataset. There is a balance between retained information and reduced complexity in the choices made during the process. This process is depicted in Fig. 4.3.

Each step removes unnecessary information from the original text. Proper preprocessing of text data sets up the analysis for success. In text analysis, far more time is spent in preparing and preprocessing the text data than in the analysis itself (Dumais et al. 1998). Diligent and detailed work in cleansing and preprocessing makes the analysis process smoother.

4.3 Unitize and Tokenize

The first step involves the choice of the unit of text to analyze and the separation of the text based on the unit of analysis. This unit could be a word; however, in other cases, it may be a grouping of words or a phrase. Single words are the simplest choice and make a good starting point. It is difficult for a computer to know where to split the text. Fortunately, most text mining software contains functions to split text, because computers do not naturally sense when punctuation designates the end of a word or sentence. For example, apostrophes could indicate the end of a token, or not, depending on the use (Weiss et al. 2010).

In our example, tokenization is done under the assumption of bag-of-words (BOW), meaning that the grammar and ordering of the text in a document is not

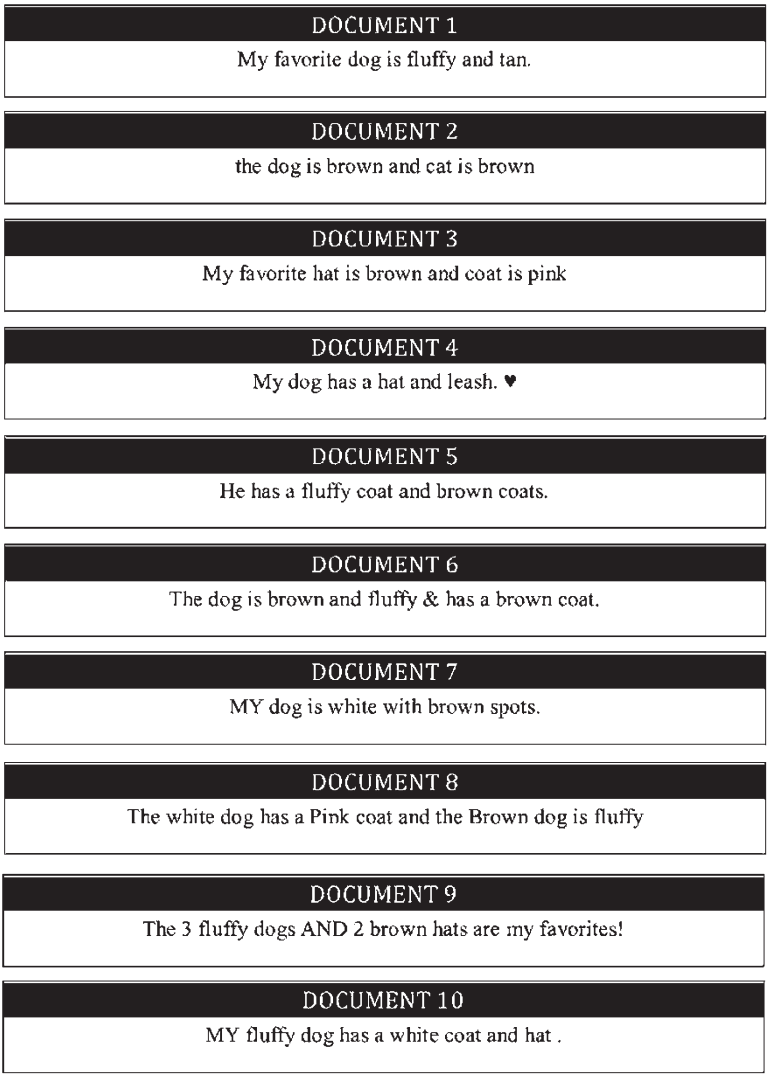


Fig. 4.2 Example document collection

considered in building the quantitative representation of the qualitative text data. First, we analyze the text in the document collection at the word or term level. For instance:

Document 1: *My Favorite Dog Is Fluffy and Tan* There are seven words in this document. Next, through tokenization, we separate the text into a more usable form, known as tokens.

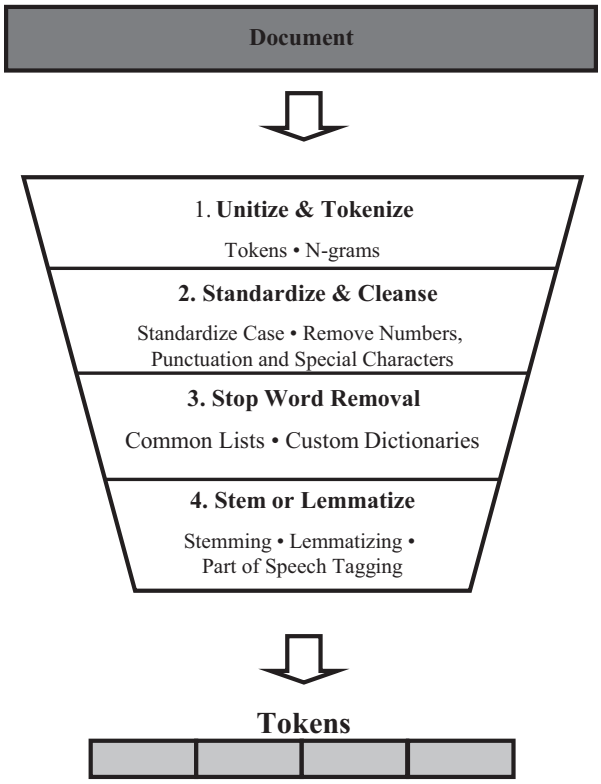


Fig. 4.3 The text data pre-processing process

Based on Document 1, we have eight tokens. Each of the seven words is a token, in addition to the period at the end of the sentence. The tokenized documents are shown in Fig. 4.4. Each bracket in the documents represents a token.

4.3.1 N-Grams

N-grams are an alternative to single words in the tokenization process. N-grams are tokens; they are consecutive word sequences with length *n*. For instance, bigrams are tokens composed of two side-by-side words; a single word is known as a unigram. N-grams retain information about the co-occurrence of words, because they group adjacent words into the same token.

Visualize the process as a picture frame that contains *n* words. Initially, the frame rests over the first *n* words. This counts as a token. The frame then moves over one word resulting in the exclusion of the first word. This is the second token. This process repeats for the length of the text (Struhl 2015). We will demonstrate this process with *n* equal to two, known as a bigram. Again, we use Document 1.

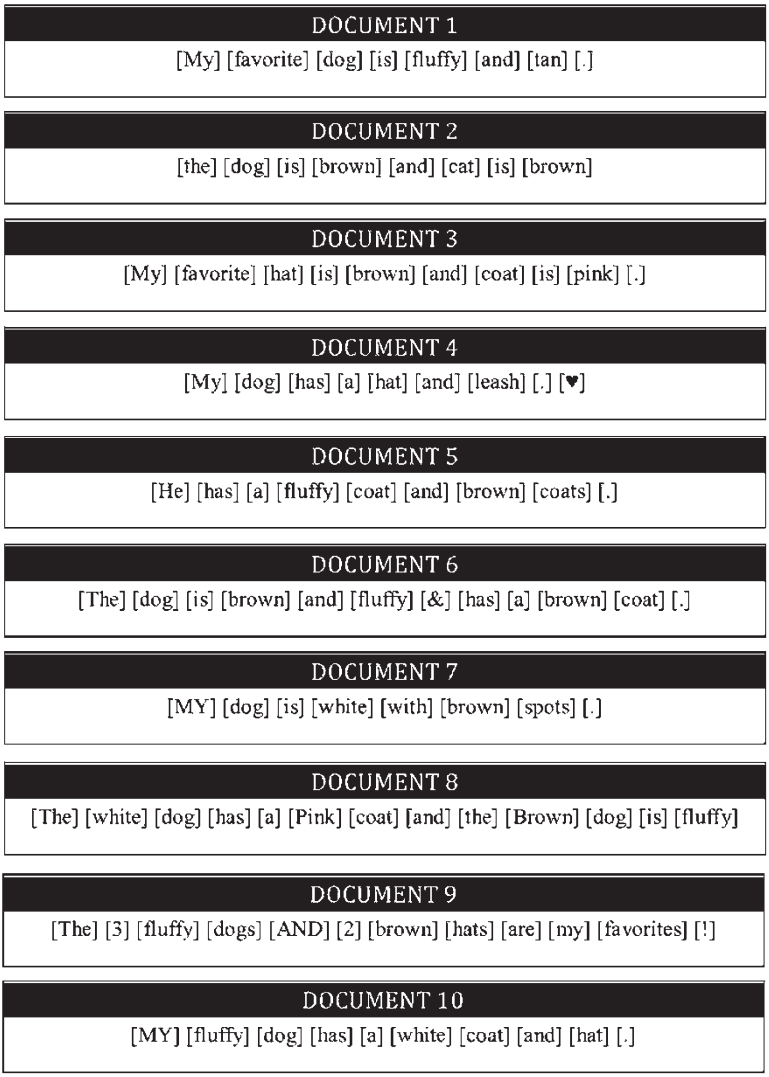


Fig. 4.4 Tokenized example documents

The first token is:
My favorite *dog is fluffy and tan.*
The second token is:
My *favorite dog* *is fluffy and tan.*
The final token is:
My favorite dog is fluffy and tan.
The bigram representation of Document 1 is:

My favorite
favorite dog
dog is
is fluffy
fluffy and
and tan
tan

This procedure can be used to create n-grams for higher values of n.

4.4 Standardization and Cleaning

First, we want to standardize and clean the tokens. These transformations level the playing field by making the terms in each of the documents comparable. For instance, we do not want *character*, *character*, and *Character* to be considered separate items just because one has a comma, and another has an upper case *C*. This standardization and cleaning prevents this possibility from occurring.

Our first step is to convert the terms in the text to lower case. In this step, any capital letters are converted to lower case. Without this conversion, the first token in Document 1, *My*, and the first token in Document 7, *MY*, would erroneously be considered two different terms.

Following this conversion, we want to remove numbers, punctuation, and special characters. In Document 9, we will remove the numbers 3 and 2. We will also remove any punctuation at the ends of sentences, such as periods and exclamation points. We remove periods from Documents 1, 3, 4, 5, 6, 7, and 10. We also remove an exclamation point from Document 9 and an ampersand from Document 6. In this document collection, we have one special character, a ♥ in Document 4, which is removed. In real-world text data, there may be additional characters and tokens to clean. For instance, in some text documents, there may be extra spaces, such as white space. These are also eliminated at this stage. The results of cleansing and standardizing our text data appear in Fig. 4.5.

4.5 Stop Word Removal

Next, we want to drop frequently used filler words, or stop words, which add no value to the analysis. According to the Oxford English Dictionary, *and*, *the*, *be*, *to*, and *of* are the most common words in the English language.[1] In the case of text analysis, we remove common terms because, although common terms such as

[1] https://en.oxforddictionaries.com/explore/what-can-corpus-tell-us-about-language

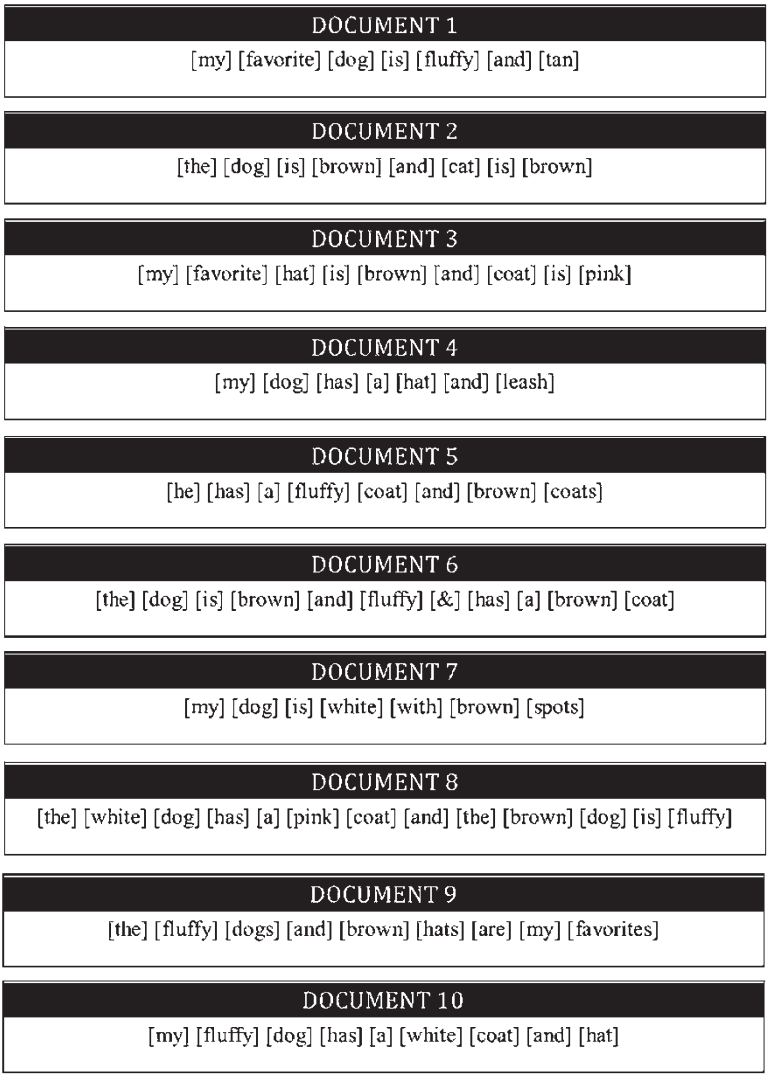


Fig. 4.5 Cleansed and standardized document collection

these serve a grammatical purpose, they provide little information in terms of content (Salton 1989; Wilbur and Sirotkin 1992).

A collection of stop words is known as a stop list. Alternatively, this collection of words can be known as a dictionary. There are several stop lists such as those presented in Chaps. 13, 14, 15 and 16 that are utilized by software programs for text analytics. These lists include many different languages and methods.[2] The

[2] Stop lists in more than 40 languages can be found at http://www.ranks.nl/stopwords

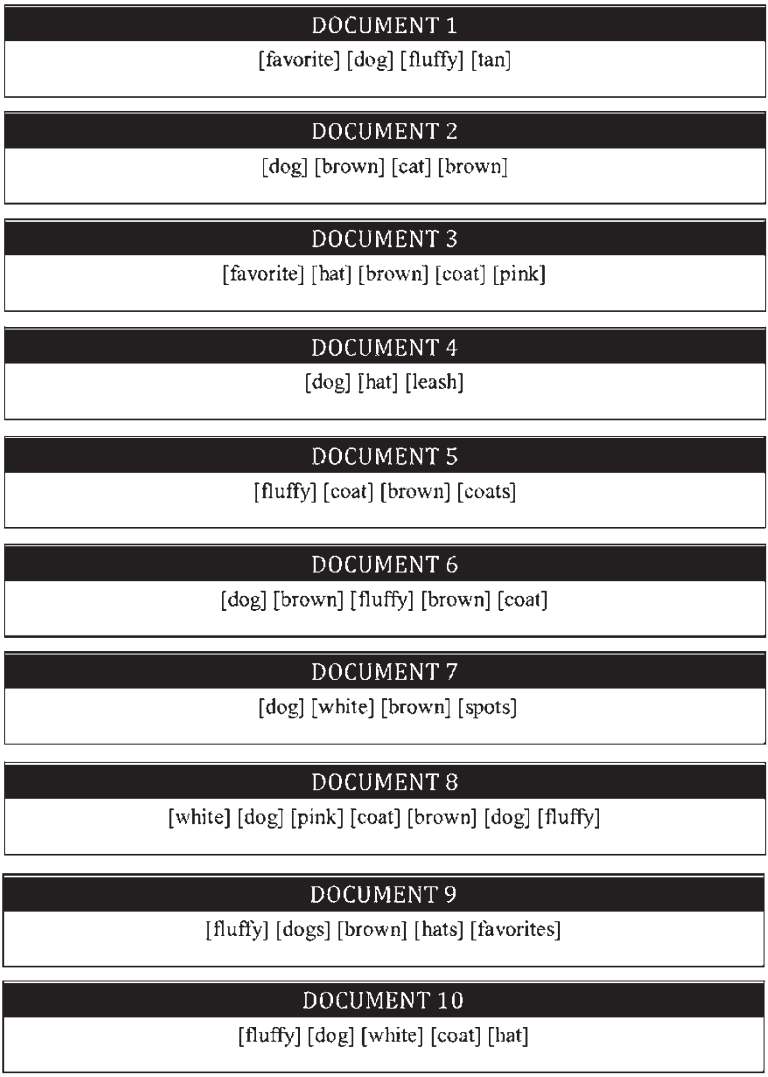


Fig. 4.6 Documents after stop word removal

SMART information retrieval system, introduced by Salton (1971), has a popular stop word list containing 571 words.[3]

In our example, we will use the SMART dictionary (Salton 1971) to identify and remove stop words from our documents. The following stop words were removed from the documents: *a*, *and*, *has*, *he*, *is*, *my*, *the*, and *was*. The resulting document collection after stop word removal is displayed in Fig. 4.6. An alternative to using existing stop word dictionaries is to create a custom dictionary.

[3] The stop word list based on the SMART information retrieval system can be found at http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

4.5.1 Custom Stop Word Dictionaries

Standard stop word dictionaries eliminate many of the most common words in a given language, but sometimes we also want to drop domain- or project-specific words or tokens. In this case we can build a custom dictionary to complete this task. Often, projects will have topic-specific words that occur frequently but add little value. In the documents in our example, we may consider adding the term *dog* to a custom stop list. Given that the ten respondents were told to describe their dog, this word may not be informative in the analysis. Since one respondent described both a cat and a dog, in this case, we chose not to remove the term. If this were not the case, the word *dog* would be included in a custom stop word dictionary and removed.

To identify these words, we can also look at term frequencies. Words that have high frequencies across all documents in our collection but do not provide informational content are good candidates for removal. After making these choices, we can select a subset of the data with the term in it and read it. While reading, we want to ensure that the term does not provide information that is relevant to the analysis question. In the creation of a custom stop word dictionary, the process should be repeated, creating subsets for reading several times to check for multiple uses of the term (Inmon 2017).

Custom dictionaries can be used in many text analysis projects and not exclusively to filter stop words. Custom dictionaries are created in information retrieval to find keywords in context (KWIC). Instead of finding tokens for removal, these keywords are used as search terms. Additional uses of common and custom lexicons will be presented in Chap. 10 for sentiment analysis.

4.6 Stemming and Lemmatization

4.6.1 Syntax and Semantics

Prior to introducing the final preprocessing step, it is necessary to consider two important concepts: syntax and semantics. Syntax concerns sentence structure, including grammar and parts of speech. Parts of speech are grammatical categories or word classes, such as noun, verb, and adjective (Manning and Schütze 1999). Semantics, on the other hand, refers to meaning. Part-of-speech tagging is beneficial in text analysis because in identifying the part of speech of a token, the most likely meaning can also be identified.

Two important semantic concepts related to part-of-speech tagging are synonymy and polysemy. Synonymy refers to two different words having the same meaning. If a document were added that said, “My cap is brown,” the terms *cap* and *hat* would demonstrate synonymy. Since the synonyms cannot be recognized automatically, each word is a separate token.

Polysemy refers to a single word having multiple meanings. In the case of the word *coat* in our example, in one context, it means a garment that can be worn, but

in another context, it can refer to a dog's hair or fur. In our example, it is possible that in Document 10, which reads, "My fluffy dog has a white coat and hat," the dog is wearing a white coat, but it is also possible that the dog has white fur. As a noun, the word *coat* can mean "fur covering the body of an animal" (Wordnet 3.1).

4.6.2 Stemming

The final stage is to perform either stemming or lemmatization on the documents. Stemming and lemmatization involve breaking words down to their root word. Stemming involves the removal of a word's suffix to reduce the size of the vocabulary (Porter 1980). Lemmatization is similar to stemming, except it incorporates information about the term's part of speech (Yatsko 2011). Both methods combine words that contain the same root into a single token to reduce the number of unique tokens within the analysis set. Words with a common root often share a similar meaning. These words are then grouped into one token (Manning et al. 2008). There are exceptions to the roots of words sharing the same meaning, but the added reduction in complexity is often worth the price of incorrectly categorizing a few words.

As an example, let's use the root *train*. *Train* has several forms, including:

- *Train*
- *Trains*
- *Trained*
- *Training*
- *Trainer*

In stemming, these words return the root *train*. Common stemmers, such as Porter's (1980), Lovins' (1968), and Paice's (1990, 1994), use a series of rules to remove word endings. These algorithms aim to return the base word. The number of characters removed changes depending on the stemming algorithm. A stemmer that removes more from a word will result in less variation among tokens and more word forms grouped within the same token. Depending on the project, this could mean better results or increased errors (Manning et al. 2008).

As an example of stemming in our document collection, we can take a closer look at Document 9, shown in Fig. 4.7, before and after stemming.

The term *fluffy* describes the dog's fur or fluff. Using Porter's stemming algorithm (Porter 1980), the term *fluffy* is broken down to the root word *fluffi*. Other terms with this root will also be replaced by the root. Some terms that would also be truncated to the root *fluffi* are *fluffier*, *fluffiest*, *fluffiness*, and *fluffily*.

As the figure shows, in the document after stemming, the terms *dogs* and *hats* are converted to their singular form, *dog* and *hat*, respectively. The term *favorites* is not only broken down to its singular form but also further reduced to the root, *favorit*. Some terms that would also be stemmed to the root *favorit* are *favorite*, *favorites*, and *favorited*. The full, stemmed document collection appears in Fig. 4.8.

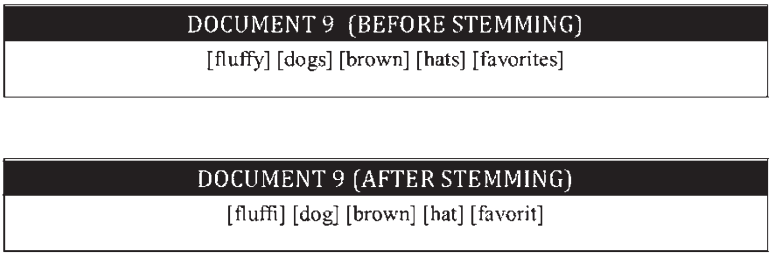


Fig. 4.7 Document 9 tokenized text before and after stemming

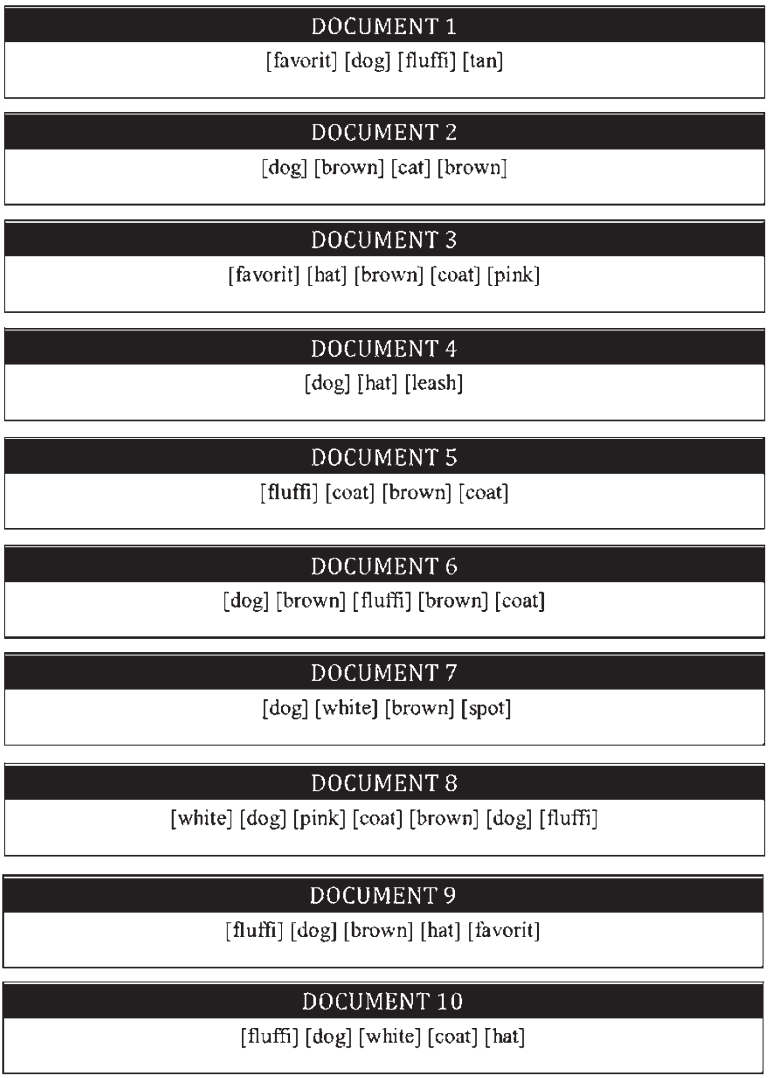


Fig. 4.8 Stemmed example document collection

4.6.3 Lemmatization

One difficulty encountered with stemming (and text analytics in general) is that a single word could have multiple meanings depending on the word's context or part of speech. Lemmatization deals with this problem by including the part of speech in the rules grouping word roots. This inclusion allows for separate rules for words with multiple meanings depending on the part of speech (Manning et al. 2008). This method helps improve the algorithm by correctly grouping tokens at the cost of added complexity.

As an example of lemmatization in our document collection, we can again look at Document 9 in Fig. 4.9. The figure depicts the document at the end of Step 3 in green and the document after stemming and after lemmatization in orange. As shown, stemming and lemmatization produce the same tokens for the terms *dog*, *brown*, and *hat* but vary with respect to *fluffy* and *favorite*.

Returning to the terms that would be truncated to the root *fluffi* using stemming, we can consider how lemmatization would impact them. These terms and their parts of speech are displayed in Table 4.1. As shown, all adjectives are lemmatized to *fluffy*, while the noun and adverb, *fluffiness* and *fluffily*, remain unchanged.

The same procedure can be done for the related terms that reduce to the root *favorit* in stemming. Table 4.2 displays the words, parts of speech, and

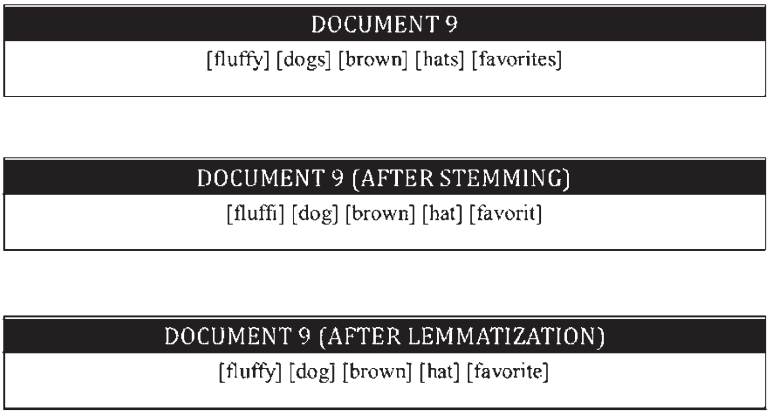


Fig. 4.9 Document 9 before and after stemming and lemmatization

Table 4.1 Document 9 related words, POS, and lemmatization for the word fluffy

Word	Part of speech	Lemmatization
Fluffy	Adjective	Fluffy
Fluffier	Adjective	Fluffy
Fluffiest	Adjective	Fluffy
Fluffiness	Noun	Fluffiness
Fluffily	Adverb	Fluffily

Table 4.2 Document 9 related words, POS, and lemmatization for the word favorite

Word	Part of speech	Lemmatization
Favorite	Noun/adjective	Favorite
Favorites	Noun	Favorite
Favorited	Verb	Favorited

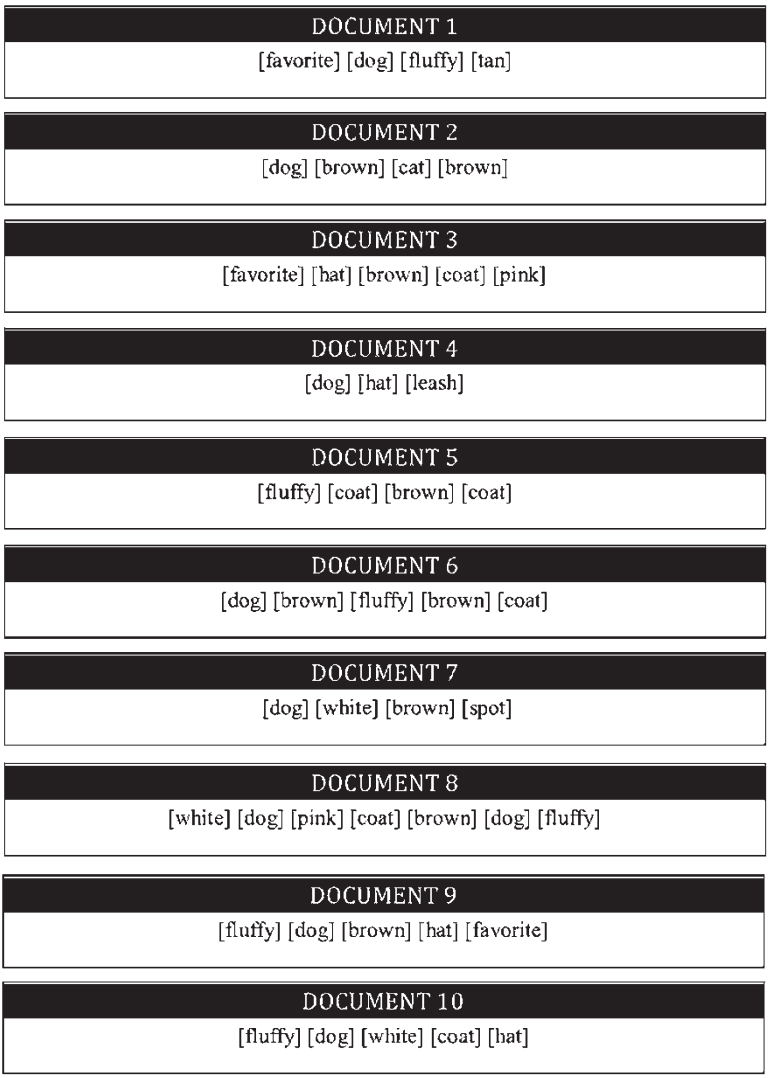


Fig. 4.10 Lemmatized example document collection

lemmatization of these terms. As shown, *favorite* and *favorites*, which are primarily nouns, are lemmatized to *favorite*, while the verb, *favorited*, remains unchanged.

The full, lemmatized document collection is displayed in Fig. 4.10.

The choice between stemming and lemmatization is up to the analyst and will depend on the application and text data.

4.6.4 Part-of-Speech (POS) Tagging

Part-of-speech tagging involves labeling tokens or words by their part of speech (Manning and Schütze 1999). Two of the most popular tag sets in English are the Brown Corpus (Kučera and Francis 1967) and the Lancaster-Oslo-Bergen (LOB) Corpus (Johansson et al. 1978). A newer tag set, the Penn Treebank, was developed in 1989 and has over 7 million words tagged by their parts of speech (Taylor et al. 2003).

Part-of-speech tagging can be completed using one of the software programs described in Chap. 1, including many of those presented in Chaps. 13, 14, 15 and 16. In these programs, the documents are entered as inputs. The program processes and outputs the word and annotates the parts of speech (Bird et al. 2009). The method used to identify the part of speech may be rule-based, Markov model-based, or maximum entropy-based (Indurkhya and Damerau 2010). Additionally, machine learning techniques, such as those introduced in Chap. 9, can be used to automatically identify the parts of speech of words in the document collection. These methods can prevent errors caused by stemming or lemmatizing words to the same root that actually have different meanings depending on the part of speech, demonstrating polysemy. The accuracy of the analysis can be improved by grouping more words, such as synonyms, appropriately.

Key Takeaways

- The text preprocessing process involves unitization and tokenization, standardization and cleaning, stop word removal, and lemmatization or stemming.
- A custom stop word dictionary can be created to eliminate noise in the text.
- Part-of-speech tagging involves labeling tokens or words by their part of speech and can be used to prevent stemming and lemmatization-related errors.
- N-grams are consecutive token sequences with length n that preserve token co-occurrence.

References

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. Beijing: O'Reilly Media, Inc.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998, November). Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management* (pp. 148–155). ACM.

Indurkhya, N., & Damerau, F. J. (Eds.). (2010). *Handbook of natural language processing* (Vol. 2). Boca Raton: CRC Press.

Inmon, B. (2017). *Turning text into gold: Taxonomies and textual analytics*. Bradley Beach: Technics Publications.

Johansson, S., Leech, G. N., & Goodluck, H. (1978). *The Lancaster-Oslo/Bergen Corpus of British English*. Oslo: Department of English: Oslo University Press.

Kučera, H., & Francis, W. N. N. (1967). *Computational analysis of present-day American English*. Providence: Brown University Press.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics, 11*(1–2), 22–31.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge: MIT Press.

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511809071.

Paice, C. D. (1990). Another stemmer. *ACM SIGIR Forum, 24*(3), 56–61.

Paice, C. D. (1994, August). An evaluation method for stemming algorithms. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 42–50). Springer-Verlag New York, Inc.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130–137.

Salton, G. (1971). *The SMART retrieval system: Experiments in automatic document processing*. Englewood Cliffs: Prentice-Hall.

Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of*. Reading: Addison-Wesley.

Struhl, S. (2015). *Practical text analytics: Interpreting text and unstructured data for business intelligence*. London: Kogan Page Publishers.

Taylor, A., Marcus, M., & Santorini, B. (2003). The penn treebank: An overview. In *Treebanks* (pp. 5–22). Dordrecht: Springer.

Weiss, S. M., Indurkhya, N., Zhang, T., & Damerau, F. (2010). *Text mining: predictive methods for analyzing unstructured information*. Springer Science & Business Media.

Wilbur, W. J., & Sirotkin, K. (1992). The automatic identification of stop words. *Journal of Information Science, 18*(1), 45–55.

Yatsko, V. A. (2011). Methods and algorithms for automatic text analysis. *Automatic Documentation and Mathematical Linguistics, 45*(5), 224–231.

Further Reading

For a more comprehensive treatment of natural language processing, see Indurkhya and Damerau (2010), Jurafsky and Martin (2014), or Manning and Schütze (1999).

Chapter 5
Term-Document Representation

Abstract This chapter details the process of converting documents into an analysis-ready term-document representation. Preprocessed text documents are first transformed into an inverted index for demonstrative purposes. Then, the inverted index is manipulated into a term-document or document-term matrix. The chapter concludes with descriptions of different weighting schemas for analysis-ready term-document representation.

Keywords Inverted index · Term-document matrix · Document-term matrix · Term frequency · Document frequency · Term frequency-inverse document frequency · Inverse document frequency · Weighting · Term weighting · Document weighting · Log frequency

5.1 Introduction

Following the text preparation and preprocessing, outlined in Chap. 4, the next step is to transform the text documents into a compatible format for text analysis. At this stage, we need to convert the text data into frequencies that can be used in analytic calculations. To build the term-document representation, we borrow some concepts from matrix algebra. In this chapter, before transforming the text data into a term-document matrix representing the frequency of each word in each document, we create an inverted index. Finally, we present several weighting measures that can be used to transform the matrix representation.

5.2 The Inverted Index

The first step toward building a representation of the terms and documents in our document collection is to create an inverted index. An inverted index contains a dictionary of the unique terms or n-grams in the preprocessed tokenized text. The index also contains postings where the documents in which each of the dictionary terms occurs are listed (Manning et al. 2008).

M. Anandarajan et al., *Practical Text Analytics*, Advances in Analytics and Data Science 2, https://doi.org/10.1007/978-3-319-95663-3_5

Table 5.1 Unprocessed and preprocessed text

Documents		
Number	Text	Preprocessed text
1	My favorite dog is fluffy and tan	[favorite] [dog] [fluffy] [tan]
2	The dog is brown and cat is brown	[dog] [brown] [cat] [brown]
3	My favorite hat is brown and coat is pink	[favorite] [hat] [brown] [coat] [pink]
4	My dog has a hat and leash	[dog] [hat] [leash]
5	He has a fluffy coat and brown coats	[fluffy] [coat] [brown] [coat]
6	The dog is brown and fluffy and has a brown coat	[dog] [brown] [fluffy] [brown] [coat]
7	My dog is white with brown spots	[dog] [white] [brown] [spot]
8	The white dog has a pink coat and the brown dog is fluffy	[white] [dog] [pink] [coat] [brown] [dog] [fluffy]
9	The three fluffy dogs and two brown hats are my favorites	[fluffy] [dog] [brown] [hat] [favorite]
10	My fluffy dog has a white coat and hat	[fluffy] [dog] [white] [coat] [hat]

Table 5.2 Inverted index for dcument collection

Dictionary	Postings							
brown	2	3	5	6	7	8	9	
cat	2							
coat	3	5	6	8	10			
dog	1	2	4	6	7	8	9	10
favorite	1	3	9					
fluffy	1	5	6	8	9	10		
hat	3	4	9	10				
leash	4							
pink	3	8						
spot	7							
tan	1							
white	7	8	10					

As described in Chap. 4, we preprocess the text to transform it into a format to create a representation of the term-document information. Table 5.1 displays the sample document collection containing ten documents in which dog owners talk about their dogs. In the text column, the raw document text is displayed. The preprocessed text appears in the third column of the table.

From the preprocessed text, we create an inverted index, illustrated in Table 5.2. The table contains the unique preprocessed terms in the document collection on the left, under Dictionary, and the document numbers in which the terms appear are on the right, under Postings. The terms listed in the dictionary can be considered the terms in the vocabulary. The inverted index creates the foundation for our term frequency representation.

Based on the inverted index, we represent the document collection as a listing of each term-posting pair. This listing includes frequency information or the number of times the term appears in a document. For example, as illustrated in

Table 5.3 Document frequency of the term *brown*

Document	Preprocessed tokenized text	Frequency
1	Favorite dog fluffy tan	0
2	Dog *brown* cat *brown*	2
3	Favorite hat *brown* coat pink	1
4	Dog hat leash	0
5	Fluffy coat *brown* coat	1
6	Dog *brown* fluffy *brown* coat	2
7	Dog white *brown* spot	1
8	White dog pink coat *brown* dog fluffy	1
9	Fluffy dog *brown* hat favorite	1
10	Fluffy dog white coat hat	0

Table 5.4 Term-postings frequency table for the term *brown*

Term	Document	Frequency
brown	1	0
brown	2	2
brown	3	1
brown	4	0
brown	5	1
brown	6	2
brown	7	1
brown	8	1
brown	9	1
brown	10	0

Table 5.3, we create a table by counting the number of times the word *brown* appears in each of the documents.

The term *brown* appears in Documents 2 and 6 twice and appears once in Documents 3, 5, 7, 8, and 9. For the term *brown*, the term-posting pairs and frequency information are displayed in Table 5.4. Now, we can represent our documents containing the term *brown* by their document and frequency.

The remaining 11 terms can be represented in the same way as *brown*. By rearranging the inverted index in Table 5.2 to include the frequency information for the term *brown* from Table 5.3, we have computed the frequency values that will make up the first row of our term-document matrix. Our next step will be to transform this list of frequencies for term-document pairs into a matrix representation for all of the terms and documents in our document collection. We begin by introducing the term-document matrix representation.

5.3 The Term-Document Matrix

Text analysis is made possible using some concepts from matrix algebra. A matrix is a two-dimensional array with m rows and n columns. Matrix A is depicted below. Each entry in the matrix is indexed as a_{ij}, where i represents the row number and j indexes the column number of the entry. There are n columns and m rows in matrix A. a_{11}, for instance, is located in the first row and first column of matrix A.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

We model textual information from our document collection in two dimensions, terms and documents. Following text parsing and inverted indexing, we can model the individual terms or tokens in each of the documents in our document collection.

In text mining, we use a specific type of matrix to represent the frequencies of terms in documents. A term-document matrix (TDM) or document-term matrix (DTM) is created to represent a collection of documents for text analysis. In a TDM, the rows correspond to terms, and the columns correspond to documents. Alternatively, in a DTM, the rows correspond to documents, and the columns correspond to terms. An illustration of the setup of the two matrices appears in Fig. 5.1. In the examples of the DTM and TDM layouts in the figure, there are three terms and three documents. The only difference between the two is the placement of the

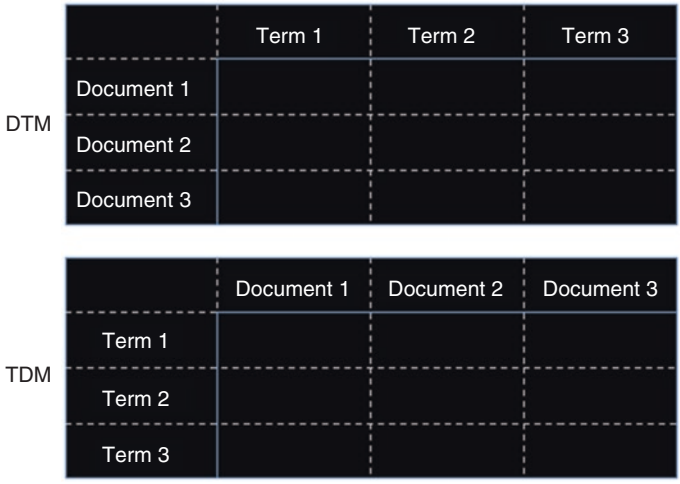


Fig. 5.1 Basic document-term and term-document matrix layouts

Table 5.5 Term-document matrix example

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
brown	0	2	1	0	1	2	1	1	1	0
cat	0	1	0	0	0	0	0	0	0	0
coat	0	0	1	0	2	1	0	1	0	1
dog	1	1	0	1	0	1	1	2	1	1
favorite	1	0	1	0	0	0	0	0	1	0
fluffy	1	0	0	0	1	1	0	1	1	1
hat	0	0	1	1	0	0	0	0	1	1
leash	0	0	0	1	0	0	0	0	0	0
pink	0	0	1	0	0	0	0	1	0	0
spot	0	0	0	0	0	0	1	0	0	0
tan	1	0	0	0	0	0	0	0	0	0
white	0	0	0	0	0	0	1	1	0	1

terms and documents, and either can be created and used in text mining analysis. In the example in this chapter, we will build a TDM.

As described, a TDM created to represent a collection of n documents has m rows and n columns, where m represents the total number of terms and n represents the total number of documents. Each entry a_{ij} contains the frequency with which term i occurs in document j. Typically, the number of terms in the TDM will be greater than the number of documents. The unique terms in the preprocessed text column of Table 5.1 are used to create the rows of our TDM. Each document in the table becomes a column in our TDM.

The term *brown* and the other terms in our vocabulary become the rows of our matrix, and the documents will be the columns. The frequency values will include the frequency for each of the term-document pairs. Any term that does not occur in a document will have a value of 0. We represent the document collection introduced in Table 5.1 as a TDM in Table 5.5. In this case, we have a 12-term by 10-document matrix. Note that the row for the term *brown* is the frequency column from Table 5.4. All of the rows in the TDM in Table 5.5 are computed in the same fashion.

Given that a matrix is a collection of points, we can represent the information visually to examine our TDM. In Fig. 5.2, a heat map is used to represent the frequency information in the TDM. The darker colors indicate lower frequency, and the lighter colors indicate higher frequency values in the TDM. The heat map shows that the words *dog* and *brown* are commonly used in our document collection, while *leash* and *spot* are rarely used.

5.4 Term-Document Matrix Frequency Weighting

When creating the TDM in Table 5.5, we used the term frequency values of each term in each document as the values in the matrix. The term frequency of term i in document j is sometimes denoted as $tf_{i,j}$. In using the term frequency in our TDM, the higher the frequency of a given term in a document, the more important

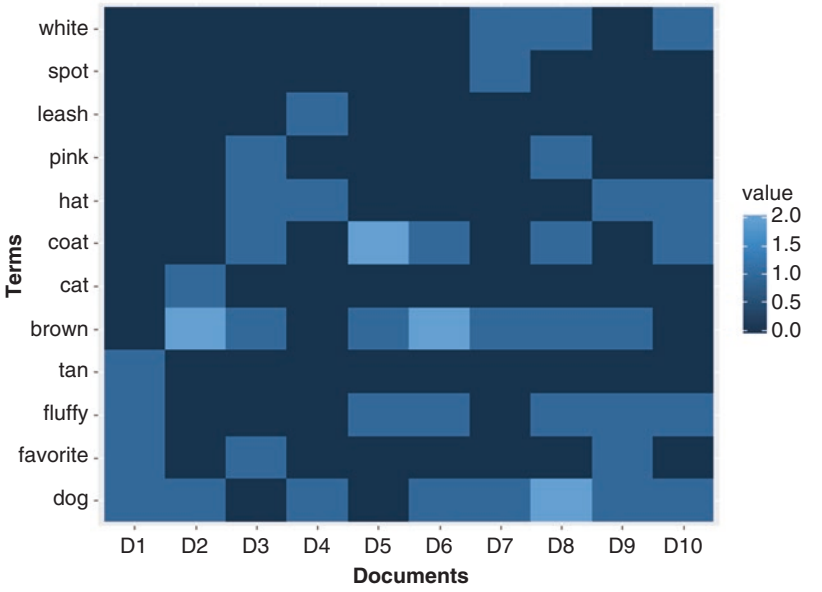


Fig. 5.2 Heat map visualizing the term-document matrix

that term is to the content of that document. For instance, in looking at the TDM in Table 5.5, the term *brown* appears twice in Document 2 and only once in Document 3. This example illuminates a major issue with using term frequency to measure importance. The term *brown* is more important in Document 2 than in Document 3, but is it twice as important? To reduce the impact of a high degree of variability in term frequencies, we can use alternative weighting approaches.

We will explore local weighting, global weighting, and combinatorial weighting approaches. Local weighting measures apply weighting to capture the importance of a term within a specific document in the larger collection of documents. This weighting tells us how much a term contributes to each of the documents. Global weighting is the overall importance of the term in the full collection of documents (Berry et al. 1999). Words that appear frequently, and in many documents, will have a low global weight. Combinatorial weighting combines local and global weighting.

5.4.1 Local Weighting

When applying local weighting, the result will be a matrix with the same dimensions as the original, unweighted TDM. In our case, the result of local weighting will be a 12-word by 10-document-weighted TDM. The local weighting alternatives that we consider are logarithmic (log) frequency and binary/Boolean frequency.

Table 5.6 Log frequency matrix

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
brown	0.0	1.1	0.7	0.0	0.7	1.1	0.7	0.7	0.7	0.0
cat	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
coat	0.0	0.0	0.7	0.0	1.1	0.7	0.0	0.7	0.0	0.7
dog	0.7	0.7	0.0	0.7	0.0	0.7	0.7	1.1	0.7	0.7
favorite	0.7	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.7	0.0
fluffy	0.7	0.0	0.0	0.0	0.7	0.7	0.0	0.7	0.7	0.7
hat	0.0	0.0	0.7	0.7	0.0	0.0	0.0	0.0	0.7	0.7
leash	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0	0.0
pink	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.7	0.0	0.0
spot	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0
tan	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
white	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.7	0.0	0.7

5.4.1.1 Logarithmic (Log) Frequency

Log frequency is a weighting method that reduces the effect of large differences in frequencies (Dumais 1991). The base of the logarithm can vary. Below, the natural logarithm, denoted ln, is used. Table 5.6 illustrates the log frequency-weighted TDM. As the table shows, the weight of the terms appearing twice in a document has a value of 1.1, and terms appearing once in a document have a value of 0.7. This method reduces the difference between the two weights from 1 to 0.4. The log frequency of term i in document j, $lf_{i,j}$, is calculated as

$$lf_{i,j} = \begin{cases} \ln\left(tf_{i,j} + 1\right), & \text{if } tf_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases}.$$

5.4.1.2 Binary/Boolean Frequency

Binary frequency captures whether a word appears in a document, without consideration of how many times it appears. In the binary frequency-weighted TDM in Table 5.7, there is no difference between a term occurring once or twice in a document. This approach is equivalent to recording if a term appears in a document. A binary frequency matrix can be used to perform further weighting on the TDM. The binary frequency of term i in document j, $n_{i,j}$, is calculated as

$$n_{i,j} = \begin{cases} 1, & \text{if } tf_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases}.$$

Table 5.7 Binary frequency matrix

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
brown	0	1	1	0	1	1	1	1	1	0
cat	0	1	0	0	0	0	0	0	0	0
coat	0	0	1	0	1	1	0	1	0	1
dog	1	1	0	1	0	1	1	1	1	1
favorite	1	0	1	0	0	0	0	0	1	0
fluffy	1	0	0	0	1	1	0	1	1	1
hat	0	0	1	1	0	0	0	0	1	1
leash	0	0	0	1	0	0	0	0	0	0
pink	0	0	1	0	0	0	0	1	0	0
spot	0	0	0	0	0	0	1	0	0	0
tan	1	0	0	0	0	0	0	0	0	0
white	0	0	0	0	0	0	1	1	0	1

5.4.2 Global Weighting

Global weighting indicates the importance of a term in the whole document collection, rather than in individual documents. When applying global weighting, the result of the calculations will be a vector of values that is the length of the total number of terms, which in our case is 12. The global weighting alternatives that we consider are document frequency, global frequency, and inverse document frequency.

5.4.2.1 Document Frequency (*df*)

Document frequency can be derived using the binary frequency-weighted TDM by summing the rows of the binary frequency-weighted TDM. Document frequency, df_i, is calculated as

$$df_i = \sum_{j=1}^{D} n_{i,j},$$

where $n_{i,j}$ is the binary frequency-weighted matrix and D is the total number of documents.

As an example, we calculate the document frequency of the word *brown* by adding the values in the row for *brown* in the binary frequency-weighted TDM.

$$df_{brown} = \sum_{j=1}^{10} n_{brown,j} = 0+1+1+0+1+1+1+1+1+0 = 7.$$

We find that the document frequency of the term *brown* is 7, meaning that the term *brown* appears in seven out of the ten documents in the collection. The

document frequency values for each of the 12 terms can be calculated the same way and are plotted in Fig. 5.3.

5.4.2.2 Global Frequency (*gf*)

Global frequency measures the frequency of terms across all documents and is calculated as

$$gf_i = \sum_{j=1}^{D} tf_{i,j},$$

where $tf_{i,j}$ is the frequency of term i in document j and D is the number of documents.

As an example, we compute the global frequency of the word *brown*. Using the formula, we calculate

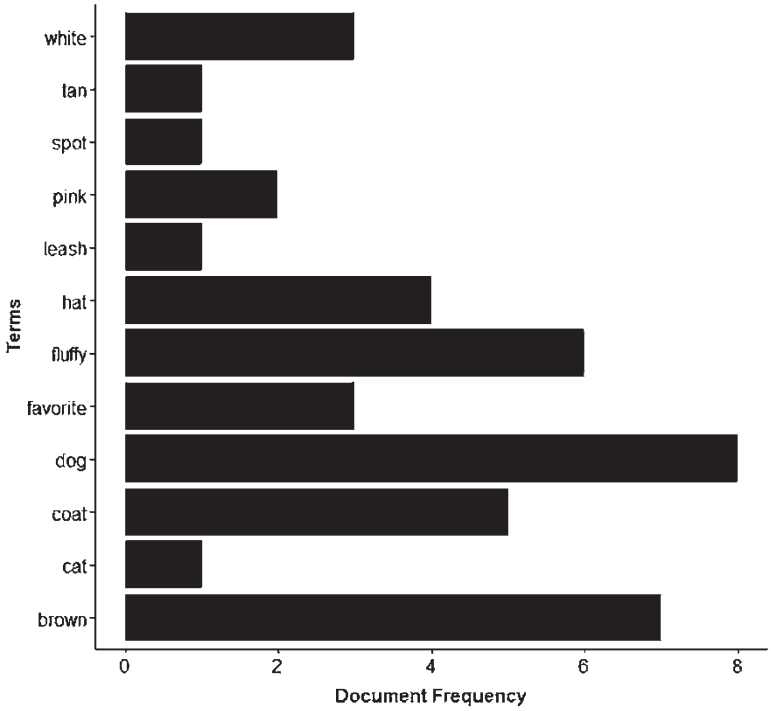


Fig. 5.3 Document frequency weighting

$$gf_{brown} = \sum_{j=1}^{10} tf_{brown,j} = 0+2+1+0+1+2+1+1+1+0 = 9.$$

The global frequencies of the other terms are computed in the same way and are displayed in Fig. 5.4.

5.4.2.3 Inverse Document Frequency (*idf*)

In inverse document frequency, rare terms have higher weights, and frequent terms have lower weights (Dumais 1991). Inverse document frequency, idf_i, is calculated as

$$idf_i = \log_2\left(\frac{n}{df_i}\right)+1,$$

where n is the total number of documents in the collection.

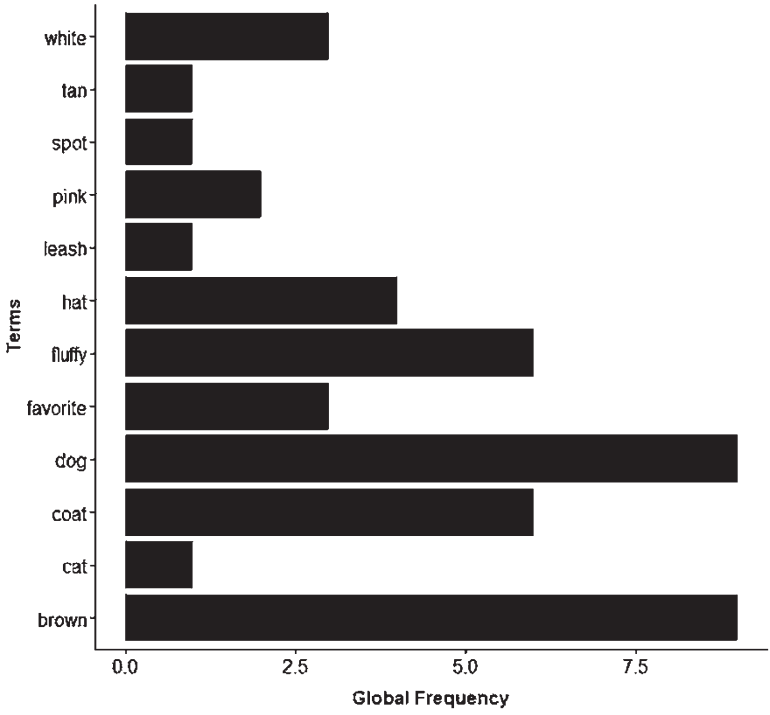


Fig. 5.4 Global frequency weighting

To find the inverse document frequency of the word *brown*, we would calculate it as follows

$$idf_{brown} = \log_2\left(\frac{10}{df_{brown}}\right) + 1 = \log_2\left(\frac{10}{7}\right) + 1 = 1.51.$$

The *idf* for each of the words can be computed in the same way. Figure 5.5 depicts the results of calculating the inverse document frequency for each of the terms.

5.4.3 Combinatorial Weighting: Local and Global Weighting

Combinatorial weighting combines local and global frequency weighting to consider the importance of each of the terms in the documents individually and in the document collection. In some cases, combinatorial weighting can also include normalization based on the total number of terms in each document. Here, we will focus on one of the most common combinatorial weighting measures, term frequency-inverse document frequency.

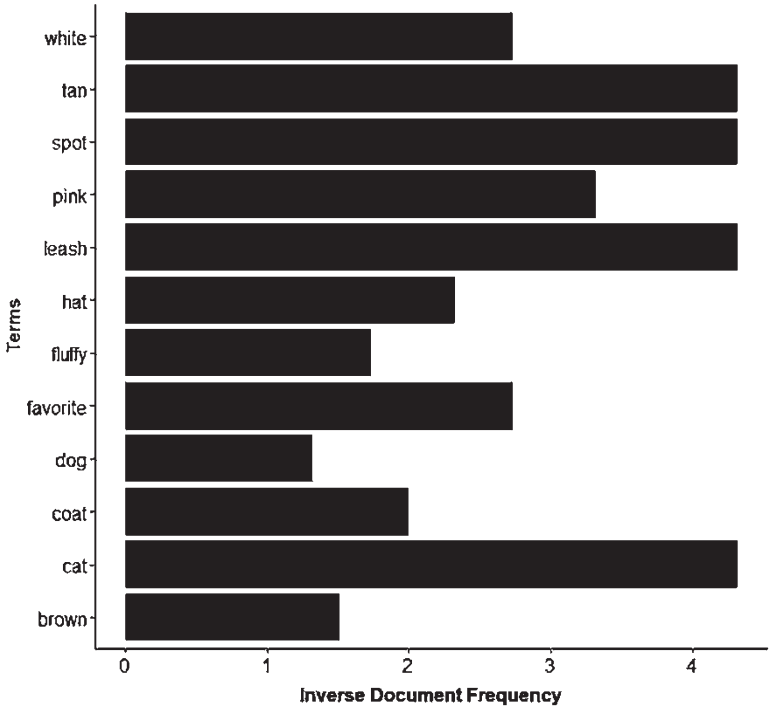


Fig. 5.5 Inverse document frequency weighting

5.4.3.1 Term Frequency-Inverse Document Frequency (*tfidf*)

Term frequency-inverse document frequency weighting combines term frequency and inverse document frequency by multiplying the local term frequency weight by the global inverse document frequency weight. *tfidf* is calculated as

$$tfidf_{i,j} = tf_{i,j} * idf_i,$$

where $tf_{i,j}$ is term frequency and idf_i is inverse document frequency.

tfidf is high when a term occurs many times in a few documents and is low when a term occurs in all, most or many documents. Intuitively, if a word appears frequently in a document or the collection of documents, it would make sense to consider the term to be important. However, the more frequently a term appears across documents, the less it actually helps with understanding the textual content.

The use of this method "balances the importance of a term to a document by its frequency in that document, evidenced by its frequency in that document, against a term's overall discriminative ability, based on its distribution across the collection as a whole" (Jessup and Martin 2001, p. 5). The *tfidf*-weighted TDM matrix appears in Table 5.8. As the table shows, terms such as *spots*, *cat*, and *tan*, which appear infrequently across the collection of documents but appear frequently in a particular document, have a high *tfidf* weight value in the documents in which they appear. The word *dog*, which appears frequently in the collection of documents, has a low weighting in the documents in which it occurs because of its high global frequency. The *tfidf*-weighted TDM, which is found by multiplying the term frequency-weighted TDM and the inverse document frequency vector, is shown in Table 5.8.

Table 5.8 *tfidf*-weighted TDM

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
brown	0.00	3.03	1.51	0.00	1.51	3.03	1.51	1.51	1.51	0.00
cat	0.00	4.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
coat	0.00	0.00	2.00	0.00	4.00	2.00	0.00	2.00	0.00	2.00
dog	1.32	1.32	0.00	1.32	0.00	1.32	1.32	2.64	1.32	1.32
favorite	2.74	0.00	2.74	0.00	0.00	0.00	0.00	0.00	2.74	0.00
fluffy	1.74	0.00	0.00	0.00	1.74	1.74	0.00	1.74	1.74	1.74
hat	0.00	0.00	2.32	2.32	0.00	0.00	0.00	0.00	2.32	2.32
leash	0.00	0.00	0.00	4.32	0.00	0.00	0.00	0.00	0.00	0.00
pink	0.00	0.00	3.32	0.00	0.00	0.00	0.00	3.32	0.00	0.00
spot	0.00	0.00	0.00	0.00	0.00	0.00	4.32	0.00	0.00	0.00
tan	4.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
white	0.00	0.00	0.00	0.00	0.00	0.00	2.74	2.74	0.00	2.74

5.5 Decision-Making

Having reviewed the frequency-weighting options, it is probably clear that each weighting schema has its own strengths and weaknesses. The choice of weighting method will depend on both the data and the intended modeling and analysis method. For instance, the first analysis method that we will explore, latent semantic analysis (LSA), covered in Chap. 6, is well suited to *tfidf* weighting. On the other hand, some topic models, including latent Dirichlet allocation (LDA), which is covered in Chap. 8, require the unweighted TDM as an input for the analysis.

In addition to the modeling and analysis considerations that influence the choice of weighting, there are some inherent weaknesses in raw frequency data that encourage the use of weighted TDMs. First, longer documents will have higher term counts than shorter documents. Additionally, high-frequency terms may be less important than lower-frequency terms. Indeed, the idea of a stop word list is based on the notion that the most common terms in a language will be the lowest in content. Terms such as *the* and *an* may be high in frequency in a document collection but will certainly be low in content.

Key Takeaways

- An inverted index represents term-posting frequency information for a document collection.
- A term-document or document-term matrix representation transforms preprocessed text data into a matrix representation that can be used in analysis.
- Local, global, and combinatorial weighting can be applied to the term-document or document-term matrix.

References

Berry, M. W., Drmac, Z., & Jessup, E. R. (1999). Matrices, vector spaces, and information retrieval. *SIAM Review, 41*(2), 335–362.

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers, 23*(2), 229–236.

Jessup, E. R., & Martin, J. H. (2001). Taking a new look at the latent semantic analysis approach to information retrieval. *Computational Information Retrieval, 2001*, 121–144.

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511809071.

Further Reading

For more about the term-document representation of text data, see Berry et al. (1999) and Manning et al. (2008).

Chapter 6
Semantic Space Representation and Latent Semantic Analysis

Abstract In this chapter, we introduce latent semantic analysis (LSA), which uses singular value decomposition (SVD) to reduce the dimensionality of the document-term representation. This method reduces the large matrix to an approximation that is made up of fewer latent dimensions that can be interpreted by the analyst. Two important concepts in LSA, cosine similarity and queries, are explained. Finally, we discuss decision-making in LSA.

Keywords Latent semantic analysis (LSA) · Singular value decomposition (SVD) · Latent semantic indexing (LSI) · Cosine similarity · Queries

6.1 Introduction

In Chapter 5, we built a term-document matrix (TDM) based on the text in our document collection. This matrix-based representation allows us to consider documents as existing in term space and terms as existing in document space. In this chapter, we present the latent semantic analysis (LSA) of the TDM. LSA is a fully automatic semantic space modeling approach in which terms are points in high-dimensional space and the spatial closeness between those points represents their semantic association (Landauer and Dumais 1997). Semantic representation tries to reveal meaning that can be hidden in the documents. Semantic knowledge extends beyond meaning to consider relations among terms and the hidden meaning and concepts present in the documents.

The examples in this chapter use the *tfidf*-weighted TDM created in Chap. 5, which includes 12 terms and 10 documents in which dog owners describe their dogs. *Tfidf* weighting is a popular weighting method used in LSA, because it combines a local and global weighting function to dampen the impact of high-frequency terms and give more weight to less frequently occurring documents that occur in fewer documents. The *tfidf*-weighted matrix used in this chapter is presented in Table 5.8. We use a weighted TDM because it produces improved results over models built with no weighting (Dumais 1991). We begin by plotting relationships based on the simple, unweighted TDM to conceptualize the term and document spaces.

M. Anandarajan et al., *Practical Text Analytics*, Advances in Analytics and Data Science 2, https://doi.org/10.1007/978-3-319-95663-3_6

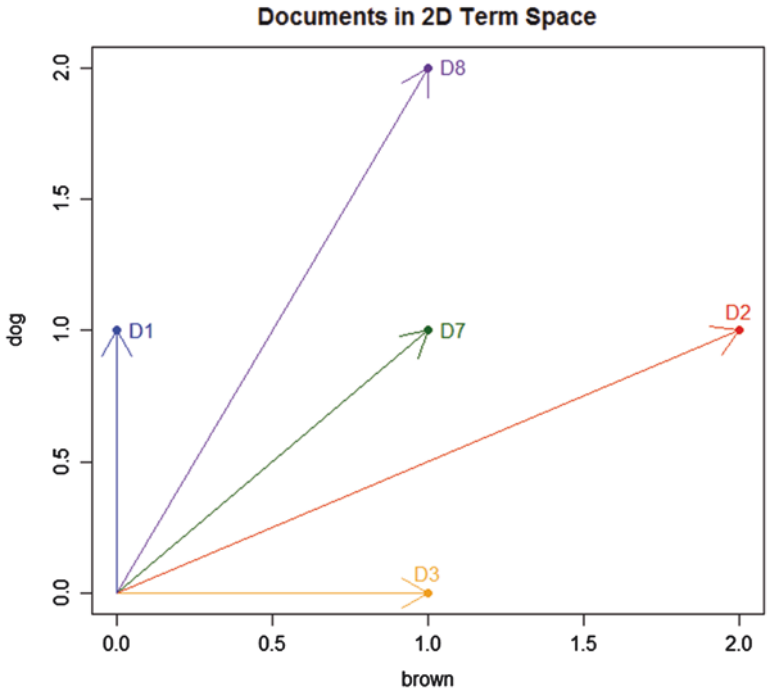


Fig. 6.1 Two-dimensional representation of the first five documents in term space for the terms *brown* and *dog*

To begin, we visualize a vector space representation. As shown in Fig. 6.1, we plot the raw frequency information for the words *brown* and *dog* using Documents 1, 2, 3, 7, and 8. Plotting documents in term space can help us understand the distance between documents and provides a geometric representation of our TDM. This figure depicts these five documents as vectors in two-dimensional term space. As shown, the documents are the points in this space. For instance, Document 1 is a point located at (0,1) in the *brown* and *dog* space, because *brown* does not occur in the document and *dog* occurs once. Document 2 is a point located at (2,1) in the *brown* and *dog* space because *brown* occurs twice and *dog* occurs once in the document. The angles formed by vectors in spatial representations are the basis for an important measure of association, cosine similarity, which will be covered in Sect. 6.2.

For more than two terms, we can visualize documents in term space in higher dimensions, as shown in Fig. 6.2. The three-dimensional term space in that figure includes the terms *brown*, *coat*, and *favorite*. In this figure, we visualize the frequencies of each of the three terms in each of our documents in our document collection. The same plots can be created to represent terms in document space. Due to the number of dimensions in the TDM, we are limited in how we can visualize these associations. However, the use of the semantic space representation allows us to model these associations in much higher dimensions than our graph allows.

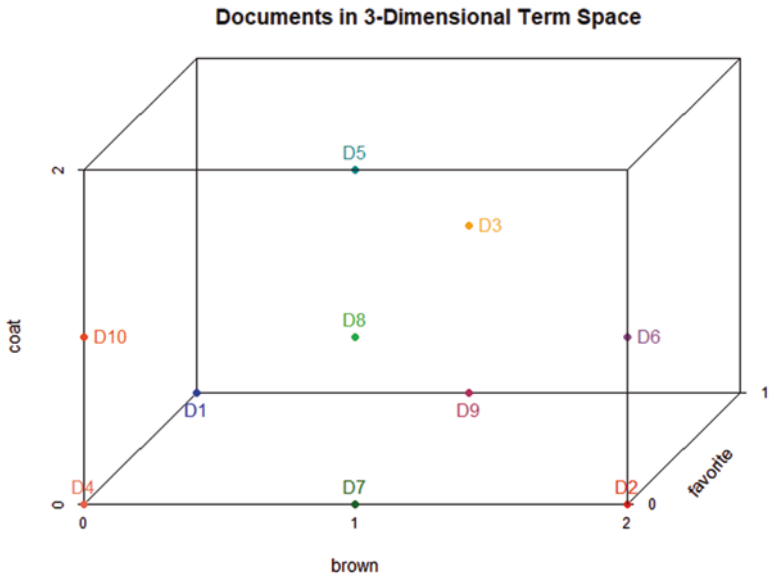


Fig. 6.2 Three-dimensional representation of the ten documents in term space for the terms *brown*, *coat* and *favorite*

6.2 Latent Semantic Analysis (LSA)

Dumais et al. (1988) and Deerwester et al. (1990) first introduced latent semantic analysis (LSA) as latent semantic indexing (LSI), due to its objective of indexing text. LSA extends the concept to all analytic applications beyond indexing. LSA creates a vector space representation of our original matrix using singular value decomposition (SVD). Specifically, LSA is an application of SVD to identify latent meaning in the documents through dimension reduction. The original TDM is assumed to be too big and sparse to be useful and/or meaningful. In real-world text applications, the TDM representation of a document collection can be very large and difficult to interpret or understand. LSA not only reduces the dimensionality but also identifies latent dimensions based on singular values. In order to understand how LSA works, we first need to familiarize ourselves with SVD.

6.2.1 Singular Value Decomposition (SVD)

LSA relies on SVD to identify latent information in the TDM. SVD splits a matrix, in our case the TDM, into three smaller matrices that, when multiplied, are equivalent to the original matrix. After this decomposition, we reduce the size of our three component matrices further by choosing to keep a smaller number of dimensions.

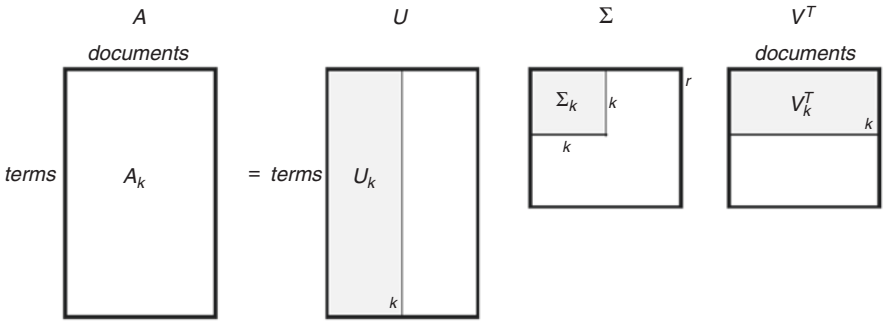


Fig. 6.3 SVD process in LSA, based on Martin and Berry (2007)

SVD is used in semantic space modeling to create smaller approximations of large document-term matrices. The truncated matrix created through SVD has four important purposes: latent meaning, noise reduction, high-order co-occurrence, and sparsity reduction (Turney and Pantel 2010, pp. 159–160).

In SVD, we calculate $A = U\Sigma V^T$, where A is the term-document matrix, U is the left singular vector of words, Σ is a matrix with weight values on the diagonal, and V is the right singular vector of documents. r is the rank of the matrix A. If we reduce r to a smaller number, k, we create an approximation of the original matrix. r can then be reduced to k, where A_k is then a lower dimensional, rank k approximation of the original matrix A. In addition, the dimensions of the three component matrices are adjusted from r to k. SVD can be applied to any rectangular matrix to decompose a larger matrix into the product of three smaller matrices. Figure 6.3 depicts the SVD of the A matrix.

6.2.2 LSA Example

When performing SVD, the TDM is known as the A matrix, which is the matrix of which we want to create a three-component representation. The rank of our A matrix, r, is 10, which is the minimum of the number of rows and number of columns in our A matrix. More formally, rank is

$$r = \min \begin{Bmatrix} m \\ n \end{Bmatrix},$$

where m is the total number of terms and n is the total number of documents.

If the number of documents in the TDM is larger than the number of terms in our TDM, the rank will be equal to the number of terms. On the other hand, if the num-

ber of terms is larger, the rank will be equal to the number of documents. In our case, we have the latter situation, because our terms outnumber our documents.

A =										
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
Brown	0.0	3.0	1.5	0.0	1.5	3.0	1.5	1.5	1.5	0.0
Cat	0.0	4.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Coat	0.0	0.0	2.0	0.0	4.0	2.0	0.0	2.0	0.0	2.0
Dog	1.3	1.3	0.0	1.3	0.0	1.3	1.3	2.6	1.3	1.3
Favorite	2.7	0.0	2.7	0.0	0.0	0.0	0.0	0.0	2.7	0.0
Fluffy	1.7	0.0	0.0	0.0	1.7	1.7	0.0	1.7	1.7	1.7
Hat	0.0	0.0	2.3	2.3	0.0	0.0	0.0	0.0	2.3	2.3
Leash	0.0	0.0	0.0	4.3	0.0	0.0	0.0	0.0	0.0	0.0
Pink	0.0	0.0	3.3	0.0	0.0	0.0	0.0	3.3	0.0	0.0
Spot	0.0	0.0	0.0	0.0	0.0	0.0	4.3	0.0	0.0	0.0
Tan	4.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
White	0.0	0.0	0.0	0.0	0.0	0.0	2.7	2.7	0.00	2.7

The *U* matrix is our term matrix and the left singular vector. The *U* matrix has 12 rows, a row for each of the terms in the TDM. The number of columns in the *U* matrix is ten, because the number of columns is equal to the rank of the *A* matrix.

U =									
−0.4	−0.3	−0.4	0.1	0.2	−0.1	0.3	0.2	−0.3	0.4
−0.1	−0.2	−0.5	0.3	0.3	−0.1	−0.2	−0.2	0.5	−0.3
−0.4	−0.1	0.0	−0.6	0.0	0.3	0.2	0.2	0.4	−0.1
−0.4	−0.1	0.1	0.3	0.0	0.2	−0.3	−0.1	−0.4	−0.1
−0.3	0.6	−0.1	0.1	−0.1	−0.3	0.3	0.0	0.0	−0.4
−0.4	0.1	0.0	−0.1	−0.2	0.4	0.0	−0.2	−0.3	−0.4
−0.3	0.3	0.3	0.1	0.4	−0.1	0.3	−0.4	0.2	0.4
−0.1	0.2	0.3	0.3	0.6	0.3	−0.1	0.5	0.0	−0.2
−0.3	0.1	0.1	−0.3	0.0	−0.6	−0.5	0.3	0.0	0.0
−0.1	−0.4	0.3	0.4	−0.3	−0.2	0.5	0.3	0.2	−0.2
−0.1	0.4	−0.2	0.3	−0.5	0.2	−0.2	0.3	0.3	0.4
−0.3	−0.3	0.4	0.1	−0.3	0.0	−0.2	−0.3	0.2	0.1

The Σ matrix contains the singular values on the diagonal, and the rest of the matrix is zeroes. The Σ matrix is a symmetric, or square matrix, with the number of rows and columns equal to the rank of our *A* matrix. For this reason, our Σ matrix has ten rows and ten columns. The singular values on the diagonal of the matrix are in decreasing order. The largest singular value is in the first column, and the smallest singular value is in the last column. This will be the case in any SVD application.

Σ =									
9.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	6.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	5.3	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	5.1	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	4.2	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	3.5	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.3	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7

The V^T matrix, in which T represents the transpose, is the document matrix and the right singular vector. The number of rows in the V^T matrix is equal to the rank of our A matrix and is ten. The number of columns in our V^T matrix is equal to the number of documents, ten. When the number of terms is larger than the number of documents in a TDM, V^T will have the same number of rows and columns, because the rank is equal to the number of documents. On the other hand, if the number of documents is larger than the number of terms in a TDM, the U matrix will have the same number of rows and columns, because the rank value, r, will be the number of terms.

V^T =									
−0.2	−0.2	−0.4	−0.1	−0.3	−0.3	−0.2	−0.5	−0.3	−0.3
0.6	−0.3	0.3	0.2	−0.1	−0.1	−0.5	−0.2	0.3	0.0
−0.2	−0.7	0.0	0.4	−0.1	−0.2	0.3	0.2	−0.1	0.3
0.4	0.4	−0.3	0.3	−0.4	−0.1	0.5	−0.2	0.2	−0.1
−0.5	0.4	0.2	0.7	0.0	0.0	−0.3	−0.1	0.1	0.0
0.2	−0.1	−0.7	0.3	0.4	0.3	−0.2	−0.1	−0.1	0.3
−0.2	−0.2	0.2	−0.1	0.3	0.2	0.4	−0.7	0.4	0.0
0.2	−0.2	0.2	0.3	0.3	0.2	0.2	0.1	−0.4	−0.7
0.2	0.3	0.3	0.0	0.3	−0.5	0.1	−0.3	−0.5	0.4
0.1	0.0	0.2	0.0	−0.5	0.6	0.0	−0.3	−0.4	0.3

Now that we have used SVD to create the three-component approximation of the original TDM, we can create a lower-rank approximation of A to reduce the size. The Σ matrix has ten singular values along the diagonal, equal to the rank of our A matrix. We want to reduce the number of singular values, thereby reducing the size of each of our three component matrices, because each of them has at least one dimension that depends on the rank of A. We choose a number, k, and reduce the size of each of our component matrices' dimensions from r to k.

We choose to retain $k = 3$ singular vectors or three latent dimensions. The reduced U, Σ, and V^T matrices are shown below. Setting $k = 3$, the U matrix has 12 rows and 3 columns, the Σ matrix has 3 rows and 3 columns, and the V^T matrix has 3 rows and 10 columns.

$U =$		
−0.4	−0.3	−0.4
−0.1	−0.2	−0.5
−0.4	−0.1	0.0
−0.4	−0.1	0.1
−0.3	0.6	−0.1
−0.4	0.1	0.0
−0.3	0.3	0.3
−0.1	0.2	0.3
−0.3	0.1	0.1
−0.1	−0.4	0.3
−0.1	0.4	−0.2
−0.3	−0.3	0.4

$\Sigma =$		
9.9	0.0	0.0
0.0	6.0	0.0
0.0	0.0	5.4

$V^T =$									
−0.2	−0.2	−0.4	−0.1	−0.3	−0.3	−0.2	−0.5	−0.3	−0.3
0.6	−0.3	0.3	0.2	−0.1	−0.1	−0.5	−0.2	0.3	0.0
−0.2	−0.7	0.0	0.4	−0.1	−0.2	0.3	0.2	−0.1	0.3

After choosing k, we multiply the above component matrices to find A_k, our reduced rank approximation of the original A matrix. Our A_k matrix, A_3, appears in Table 6.1 in what is referred to as the LSA space.

Table 6.1 The LSA space

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
brown	0.6	3.0	1.2	−0.7	1.8	2.3	1.1	2.1	1.0	0.7
cat	0.1	2.6	0.0	−1.4	0.8	1.2	0.0	0.3	0.1	−0.6
coat	0.7	1.1	1.5	0.5	1.4	1.5	1.3	2.3	1.2	1.5
dog	0.6	0.8	1.3	0.6	1.1	1.2	1.2	2.0	1.0	1.4
favorite	2.6	0.1	2.0	0.8	0.6	0.6	−1.2	0.4	2.0	0.5
fluffy	1.3	0.7	1.6	0.6	1.0	1.1	0.4	1.5	1.4	1.1
hat	1.4	−1.1	1.7	1.5	0.4	0.2	0.2	1.2	1.4	1.4
leash	0.3	−1.4	0.6	1.1	−0.2	−0.4	0.2	0.4	0.4	0.8
pink	0.7	0.2	1.2	0.7	0.8	0.8	0.8	1.5	1.0	1.2
spot	−1.3	0.0	−0.2	0.2	0.3	0.3	1.7	1.2	−0.5	0.9
tan	1.8	0.1	1.1	0.3	0.2	0.2	−1.3	−0.2	1.2	−0.1
white	−1.0	−0.2	0.5	0.9	0.8	0.7	2.4	2.3	0.1	1.9

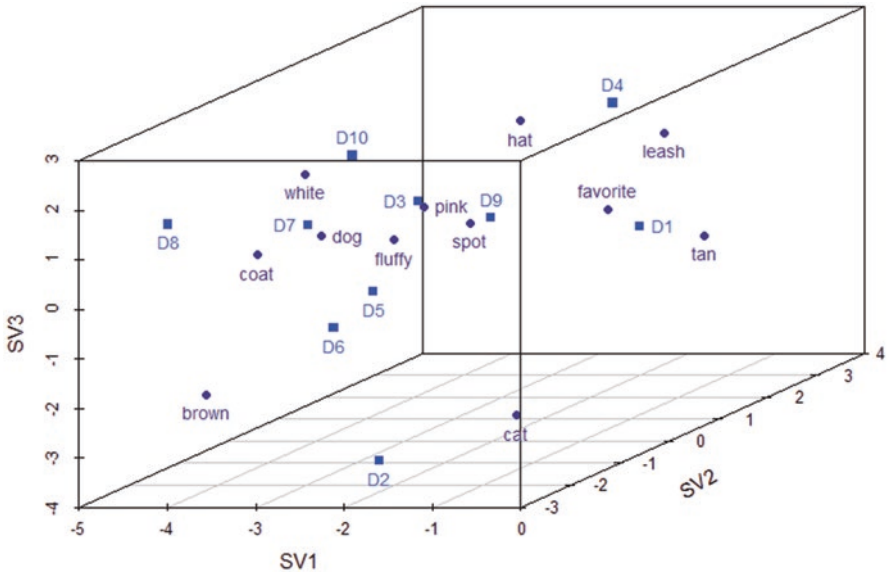


Fig. 6.4 Terms and documents in three-dimensional LSA vector space

Figure 6.4 shows the documents and terms in three-dimensional LSA vector space. The documents are depicted as light blue squares, and the terms are darker blue dots. We created this representation by multiplying our reduced U and V^T matrices by the Σ matrix. This visualization has a clear advantage over the visualization in Fig. 6.2 in that we can view both documents and terms concurrently across the three dimensions. Each of the documents and terms in the LSA vector space can be thought of as vectors emanating from the origin.

For concreteness, Fig. 6.5 presents a two-dimensional version of Fig. 2.4 with only the first two dimensions. The dashed lines drawn from the origin to *leash* and *cat* can be drawn for each document and term, because they are vectors. Using these concepts, we can map the associations between terms, documents, and terms and documents. This depiction gives rise to a geometric measure of closeness known as cosine similarity.

6.3 Cosine Similarity

The primary method of measuring the association between terms and documents in LSA is cosine similarity. Cosine similarity is a means of measuring the semantic similarity of words, regardless of their actual co-occurrence in text documents (Landauer and Dumais 1997). By applying LSA, we model the terms and documents in our TDM in vector space. Doing so gives us the ability to model many right triangles emanating from the origin to the documents and terms. While the

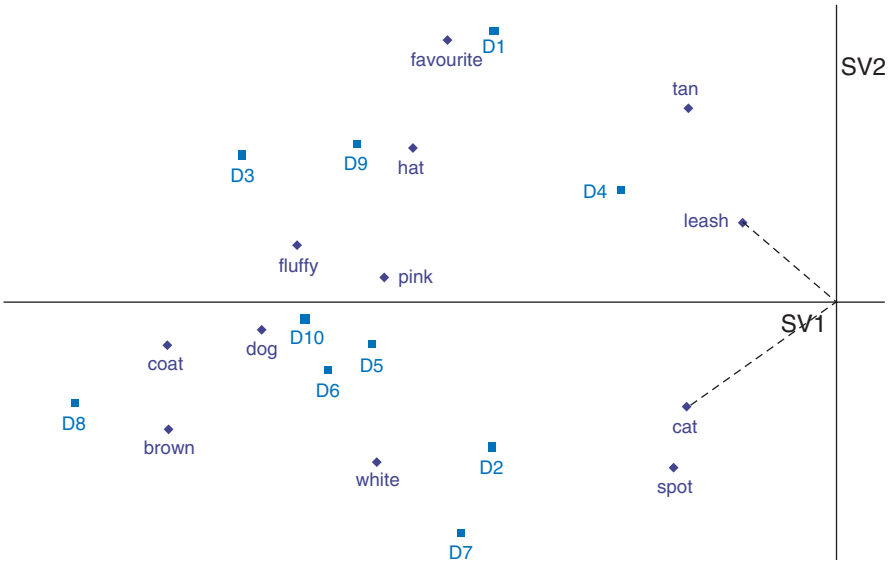


Fig. 6.5 Terms and documents of a two-dimensional LSA solution across the first two dimensions

calculation is made in multidimensional space, we consider cosine similarity in two dimensions for concreteness.

From trigonometry, we know that the cosine is the angle between two vectors. In Fig. 6.5, an angle is formed by the two term vectors meeting at the origin. The rows of the U matrix are converted to column vectors for the calculation. Cosine similarity can be applied to terms, documents, or both. It can also be applied to queries, or pseudo-documents, which will be covered in Sect. 6.5. Cosine similarity scores range between −1 and 1 but are typically greater than 0. The cosine similarity for two terms, t_1 and t_2, is calculated as.

$$Cosine\left(t_1,t_2\right)=\frac{t_1^{\,T}t_2}{t_1t_2},$$

where $t_1^{\,T}$ is the transpose of the t_1 vector and ||| represents the vector norm. In the case of documents, $Cosine(d_1, d_2)$, d_1 and d_2 replace t_1 and t_2, representing Document 1 and Document 2, respectively.

Our example uses the formula above to estimate the cosine similarity for two terms, rounding to two decimal places for all intermediate calculations. To calculate the cosine similarity measurement between the terms *fluffy* and *pink*, we use the row vectors from the LSA space, A_3, corresponding to these words.

fluffy:									
1.25	0.74	1.58	0.56	1.03	1.12	0.43	1.54	1.35	1.10

pink:									
0.71	0.17	1.24	0.74	0.78	0.78	0.75	1.50	0.98	1.22

$$fluffy^T\ pink = \sum_{i=1}^{10} fluffy_i * pink_i$$

$$fluffy^T\ pink = 0.89 + 0.13 + 1.96 + 0.41 + 0.80 + 0.87 + 0.32 + 2.31 + 1.32 + 1.34 = 10.35$$

The numerator is the cross product, or the dot product, of the two vectors, which is calculated to be 10.35. The denominator is the product of the matrix norms of the two vectors, which can be computed as the sum of the squared vector values.

$$fluffy = \sqrt{1.25^2 + 0.74^2 + 1.58^2 + 0.56^2 + 1.03^2 + 1.12^2 + 0.43^2 + 1.54^2 + 1.35^2 + 1.10^2} = 3.58$$

$$pink = \sqrt{0.71^2 + 0.17^2 + 1.24^2 + 0.74^2 + 0.78^2 + 0.78^2 + 0.75^2 + 1.50^2 + 0.98^2 + 1.22^2} = 3.02$$

$$fluffypink = 10.81$$

The cosine similarity measure of the terms *fluffy* and *pink* can then be calculated by $Cosine(fluffy, pink) = \frac{10.35}{10.81} = 0.96$. The cosine similarity values for *fluffy* and the remaining 11 words are calculated in the same way. Table 6.2 presents the cosine similarity values for the term *fluffy* in descending order. The terms *pink*, *dog*, and *coat* have high cosine similarity values with the word *fluffy*. On the other hand, *spots*, *cat*, and *leash* have low cosine values with the word *fluffy*.

The cosine similarity values for all term-term relationships are displayed in Table 6.3. Pairs of terms with the highest cosine similarity values around 1.0 are *coat* and *dog*, *coat* and *pink*, *dog* and *pink*, *favorite* and *tan*, and *fluffy* and *pink*. Pairs of terms with very high cosine similarity values around 0.90 are *brown* and *coat*,

Table 6.2 Cosine similarity measures for *fluffy*, in descending order

Term	Cosine
pink	0.96
dog	0.94
coat	0.94
hat	0.79
brown	0.79
favorite	0.75
white	0.55
tan	0.54
leash	0.32
cat	0.27
spots	0.19

Table 6.3 Term-term cosine similarity measures

	brown	*cat*	*coat*	*dog*	*favorite*	*fluffy*	*hat*	*leash*	*pink*	*spot*	*tan*	*white*
brown	*0.0*											
cat	0.8	0.0										
coat	0.9	0.3	0.0									
dog	0.8	0.3	1.0	0.0								
favorite	0.3	0.0	0.5	0.5	0.0							
fluffy	0.8	0.3	0.9	0.9	0.8	0.0						
hat	0.3	−0.4	0.7	0.7	0.8	0.8	0.0					
leash	−0.3	−0.8	0.2	0.3	0.4	0.3	0.8	0.0				
pink	0.7	0.1	1.0	1.0	0.6	1.0	0.9	0.5	0.0			
spots	0.3	0.0	0.5	0.5	−0.5	0.2	0.1	0.2	0.4	0.0		
tan	0.2	0.0	0.2	0.2	1.0	0.5	0.6	0.2	0.4	−0.7	0.0	
white	0.5	−0.1	0.8	0.8	−0.1	0.5	0.5	0.4	0.7	0.9	−0.4	0.0

fluffy and *coat*, *fluffy* and *dog*, *hat* and *pink*, and *spots* and *white*. The lowest cosine similarity value is for the terms *cat* and *leash*. This result seems reasonable, because these terms should be unrelated.

6.4 Queries in LSA

In the field of information retrieval (IR), the use of the LSA space to explore queries is an essential tool. Anytime you open your browser to a search engine and type in search keywords, you are using a query. Based on the keywords that you provide, the search engine returns websites that it believes match your search criteria. In a similar way, LSA uses the cosine measures to find documents that are similar to words that you designate as query terms (Deerwester et al. 1990). A query is represented as a scaled, weighted sum of the component term vectors. A query is equal to

$$query = q^T U_k \Sigma_k^{-1},$$

where q^T is a vector of the terms in the query, U_k is the term matrix, and Σ_k^{-1} is the inverse of the Σ_k matrix. Multiplying by the inverse of a matrix is equivalent to dividing by the matrix that is inverted. The query is a pseudo-document with a vector representation, which can be compared to the documents in the collection.

For instance, a query could include the component terms *tan*, *brown*, and *pink*. The q^T vector of this query is

q^T= [1 0 0 0 0 0 0 0 1 0 1 0], based on the pseudo query below.[1]

brown	*cat*	*coat*	*dog*	*favorite*	*fluffy*	*hat*	*leash*	*pink*	*spot*	*tan*	*white*
1	0	0	0	0	0	0	0	1	0	1	0

[1] Note: The q^T vector is created using binary frequency, because at this stage weighting cannot be calculated and applied to the pseudo-document.

Table 6.4 Cosine values between the query (*brown*, *pink*, *tan*) and documents in descending order by cosine similarity value

Document	Cosine
6	0.81
5	0.78
9	0.73
2	0.71
1	0.69
3	0.66
8	0.24
10	−0.08
7	−0.30
4	−0.30

Using the query formula above, we find that the query is equal to (−0.08, 0.03, −0.09). We use this result to determine the cosine similarity values for the query and each of the documents in the document collection to find the documents most associated with the query. Table 6.4 displays the list of documents in descending order of similarity. As the table illustrates, Documents 6 and 5 are most closely associated with the query, while 7 and 4 are the least associated with, or similar, to the query.

This similarity is based on the angles between the vectors, not physical proximity. Figure 6.6 shows the documents across the three dimensions. The origin is denoted (0, 0, 0), and the pseudo-document query is labeled "query." A thick black line is drawn between the origin and the query. Since Documents 6 and 5 have the highest cosine similarity, these vectors are drawn in green. The document vectors with the lowest cosine similarity, 4 and 7, are plotted in red. As shown, this line between Document 6 and the query nearly overlaps, resulting in a very small angle between the two vectors and the highest cosine similarity. On the other hand, while Document 7 is physically close to the query, the angle between the two vectors is much larger than between the query and Document 6.

6.5 Decision-Making: Choosing the Number of Dimensions

The choice of the number of singular values is an important decision in LSA modeling. If too few SVD dimensions are retained, we run the risk of losing important information. On the other hand, if we keep too many, our calculations and solution may be too complex to be meaningful. For this reason, simple solutions are preferable. Past research has suggested that in the case of big data and very large TDMs (or DTMs), using between 100 and 300 dimensions provides good performance (Berry and Browne 1999; Dumais 1991; Landauer and Dumais 1997).

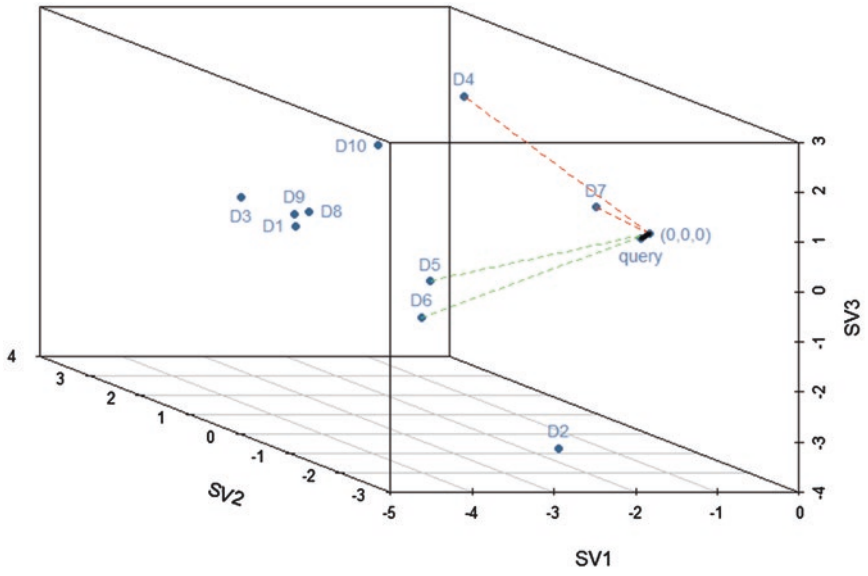


Fig. 6.6 Rotated plot of the query and Document 6 vectors in three-dimensional LSA vector space

In Fig. 6.7, known as a scree plot, we plot the number of singular vectors and the amount of variance explained. A scree plot is a visual aid used to evaluate the tradeoff between efficiency and complexity. In doing so, we look for an elbow in the graph, or the point at which the slope between points is very small, to determine the number of singular vectors to keep. As is evident in the figure, the identification of the elbows is a subjective decision, as is the number of singular values to retain. In the scree plot below, it appears that three dimensions are appropriate.

LSA has many benefits and is widely used in information retrieval and text mining applications. There are many advantages of using LSA, including its ability to handle the sparsity, size, and noise associated with a TDM. In the case of large document collections, the associated TDM will be large and sparse. In addition, due to the uncertainty involved and the qualitative nature of the text data, the data are noisy. LSA can cut through a lot of the noise of the TDM because it is rooted in dimension reduction. Additionally, in reducing the dimensionality, it uncovers latent factors that are otherwise hidden within the data. In LSA, indirect co-occurrences, where, for instance, two words are related through a third word, become important. LSA allows us to compute an association measure, cosine similarity, on a lower-rank matrix rather than our original TDM.

LSA has the innate ability to uncover deep semantic relationships in the terms and documents in the space (Landauer et al. 1998). It can identify similarity that stretches beyond just synonymy and is able to determine the importance of terms (Hu and Liu 2004). LSA handles the types of noisy, sparse matrices that are produced in text analysis through the use of SVD. Additionally, it can create pseudo-documents to measure similarity between existing documents and queries. While

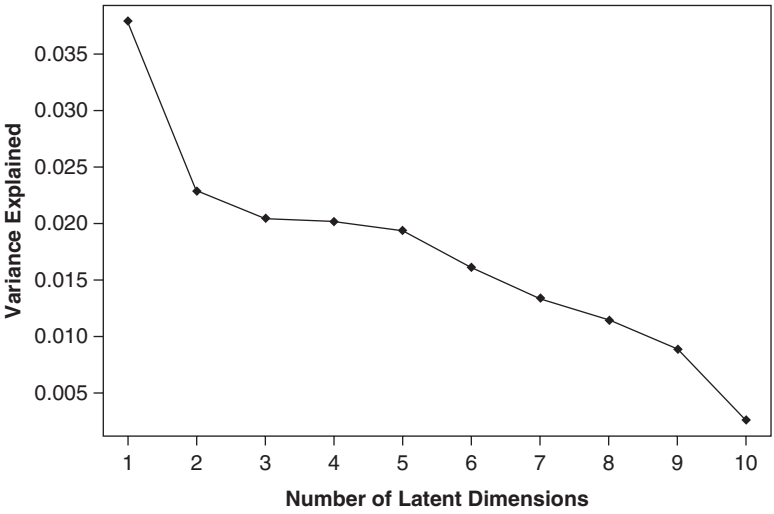


Fig. 6.7 Scree plot showing variance explained by number of singular vectors

the traditional LSA methods do not account for word ordering, because they assume a bag-of-words representation, newer methods extend the methods to incorporate word ordering. LSA can also be applied to TDMs that are based on n-grams or tokens larger than $n = 1$.

Despite the many benefits of LSA, there are limitations to its application. An LSA space is created for a particular document collection, and the results depend heavily on the type of weighting chosen and the number of singular vectors or latent factors retained. The decisions made by the analyst are particularly impactful, and in this sense, the analysis is both an art and a science.

Key Takeaways

- Latent semantic analysis (LSA) can uncover underlying or latent meaning in text.
- LSA uses singular value decomposition (SVD) to reduce the dimensionality of the TDM.
- Cosine similarity based on the LSA space can be used to assess the closeness of term-term and document-document relationships.
- Queries based on pseudo-documents can be calculated based on the LSA space to assess similarity between pseudo-documents and actual documents represented in the space.

References

Berry, M. W., & Browne, M. (1999). *Understanding search engines: Mathematical modeling and text retrieval*. Philadelphia: Society for Industrial and Applied Mathematics.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391.

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers, 23*(2), 229–236.

Dumais, S. T., Furnas, G. W., Landauer, T. K., & Deerwester, S. (1988). Using latent semantic analysis to improve information retrieval. In *Proceedings of CHI'88: Conference on Human Factors in Computing* (pp. 281–285). New York: ACM.

Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007). Topics in semantic representation. *Psychological Review, 114*(2), 211–244.

Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 168–177). ACM.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review, 104*(2), 211.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes, 25*(2–3), 259–284.

Martin, D. I., & Berry, M. W. (2007). Mathematical foundations behind latent semantic analysis. In *Handbook of latent semantic analysis*, 35–56.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research, 37*, 141–188.

Further Reading

For more about latent semantic analysis (LSA), see Landauer et al. (2007).

Chapter 8
Probabilistic Topic Models

Abstract In this chapter, the reader is introduced to an unsupervised, probabilistic analysis model known as topic models. In topic models, the full TDM (or DTM) is broken down into two major components: the topic distribution over terms and the document distribution over topics. The topic models introduced in this chapter include latent Dirichlet allocation, dynamic topic models, correlated topic models, supervised latent Dirichlet allocation, and structural topic models. Finally, decision-making and topic model validation are presented.

Keywords Topic models · Probabilistic topic models · Latent Dirichlet allocation · Dynamic topic models · Correlated topic models · Structural topic models · Supervised latent Dirichlet allocation

8.1 Introduction

Topic models, also referred to as probabilistic topic models, are unsupervised methods to automatically infer topical information from text (Roberts et al. 2014). In topic models, topics are represented as a probability distribution over terms (Yi and Allan 2009). Topic models can either be single-membership models, in which documents belong to a single topic, or mixed-membership models, in which documents are a mixture of multiple topics (Roberts et al. 2014). In this chapter, we will focus on mixed-membership models. In these models, the number of topics, k, is a fixed number that is chosen prior to building the model.

Latent semantic analysis (LSA), which is covered in Chap. 6, and topic models are both dimension reduction methods and use the document-term matrix (DTM) or term-document matrix (TDM) as the input for the analysis. While LSA discovers hidden semantic content, topic models reveal thematic structure. LSA aims to uncover hidden meaning in text, while topic models focus on the underlying subjects or themes that are present in the documents.

The most common type of topic model was created as an extension of the probabilistic latent semantic indexing (pLSI) model proposed by Hofmann (1999), which is a probabilistic LSA model. Figure 8.1 shows how the specific dimension

M. Anandarajan et al., *Practical Text Analytics*, Advances in Analytics and Data Science 2, https://doi.org/10.1007/978-3-319-95663-3_8

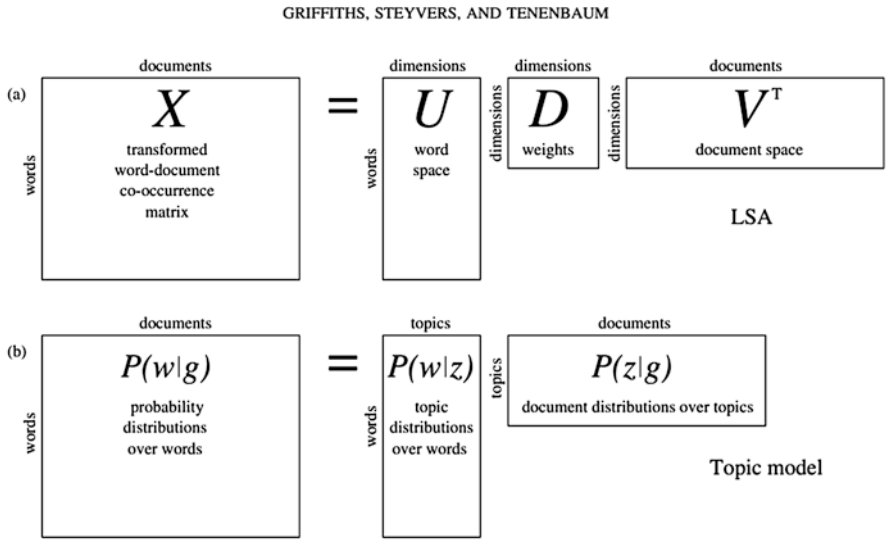


Fig. 8.1 LSA and topic models (Griffiths et al. 2007, p. 216)

reduction in LSA compares to that of topic models (Griffiths et al. 2007). As detailed in Chap. 6, LSA uses singular value decomposition (SVD) to break down the full TDM or DTM into three smaller component matrices. From there, the number of singular vectors can be reduced to create a smaller dimensional representation of the original. In topic modeling, the full TDM or DTM is broken down into two major components: the topic distribution over terms and the document distribution over topics. The first component tells us the importance of the terms in topics, and using that importance information, the second component tells us the importance of topics in the documents.

While there are many similarities between LSA and topic models, they also differ in many ways. Unlike LSA, topic models are generative probabilistic models. Based on the assigned probability, we can understand topics through their most likely terms. We are also able to better understand documents based on their most likely topics in their topic distribution. Unlike the latent factors in LSA, each topic is clearly identified and explainable.

The topic modeling examples in this chapter use text data based on 56 documents in which people describe the physical appearance of their pet dogs. There are four different breeds of dogs described in the documents: Bichon Frise, Dachshund, Great Dane, and Golden Retriever. The dogs vary in terms of height, weight, size, color, and fur. Each of the breeds has distinguishing features that are characteristic of that breed. For instance, Bichon Frises are small, fluffy dogs that are predominantly white and usually have black noses. On the other hand, a Great Dane is a very large dog with a short, straight coat that can be one of the several colors or a mix of colors. In the document sample, each dog type is described in 14 of the 56 documents.

Preprocessing and parsing are applied to the 56 text documents, including the removal of stop words and stemming. Since some of the pet owners describe their dogs in terms of weight, we do not remove numbers from our corpus. The *tfidf*-weighted DTM is used, and the resulting vocabulary contains 96 terms. The first topic model that we will present is the latent Dirichlet allocation (LDA). Many of the alternative topic models use LDA as the basis of their model.

8.2 Latent Dirichlet Allocation (LDA)

The latent Dirichlet allocation (LDA) model is a generative probabilistic model introduced in Blei et al. (2002, 2003). LDA assumes a bag-of-words (BOW) model representation, meaning that term ordering in a document is not considered when building the topic model (Blei et al. 2003). Additionally, LDA assumes that documents are exchangeable, meaning that there is no meaningful sequential ordering of the documents in the collection (Blei et al. 2010). Another assumption of LDA is the independence of topics. Figure 8.2 provides an illustration of the LDA.

In the LDA model, K is the total number of topics, D is the total number of documents, and N is the total number of words in a document, where $W_{d,\,n}$ is an observed word. Additionally, α is the Dirichlet parameter, and η is the topic hyperparameter. Each topic is a distribution over terms, with topic assignments $Z_{d,\,n}$. Each document is a mixture of topics, with topic proportions θ_d, and each term is drawn from one of the topics, with topic assignments β_k. Due to the intractability of computing the posterior distribution of the topics in a document, approximation methods are used, including mean field variational methods, expectation propagation, collapsed Gibbs sampling, and collapsed variational inference.

Using the data in the example, we build a model with four topics. We choose four topics as our starting point because we know that there are four dog breeds represented in the document collection. The top ten terms in each of the four topics based on the expected topic assignment are depicted in Fig. 8.3. Based on the figure, Topic 2 can be described using the word *white*, and we would expect to see the documents describing Bichon Frises assigned to this topic. Topic 1 can be described by the terms *long*, *tail*, and *short*; Topic 3 can be described by the terms *weigh*, *pound*, and *coat*; and Topic 4 can be described by *coat* and *ear*. One of the strengths of topic models can be seen in Topic 3. The terms *weigh* and *pound* are

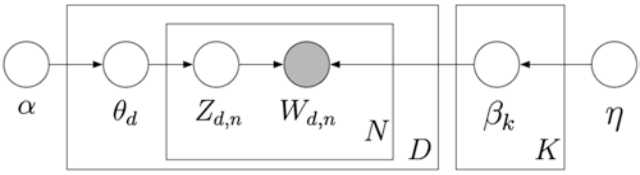


Fig. 8.2 Plate representation of the random variables in the LDA model (Blei 2012, p. 23)

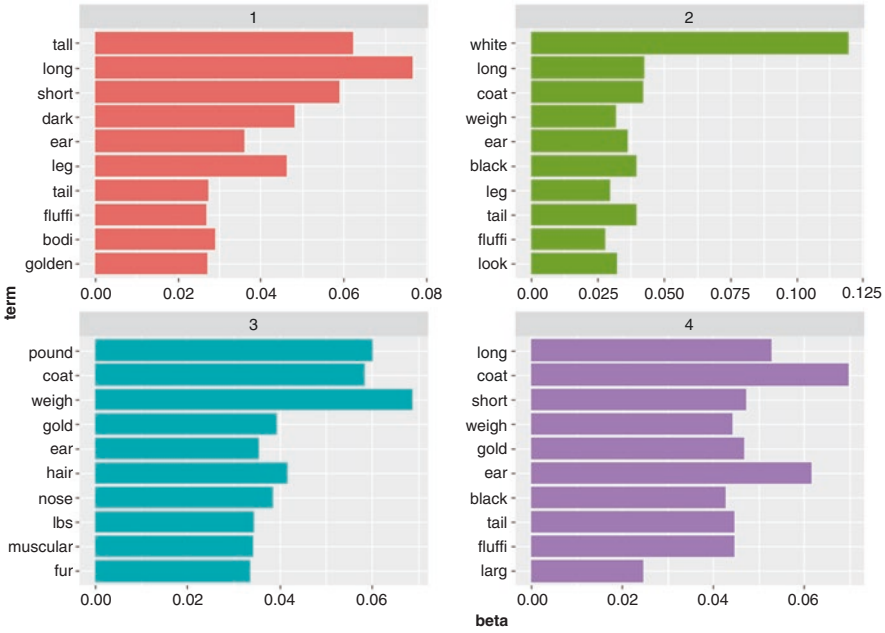


Fig. 8.3 Top ten terms per topic, four-topic model

synonyms, and they are most likely terms in the same topic. Unlike other methods, such as LSA, topic models are able to handle synonymous terms very well.

8.3 Correlated Topic Model (CTM)

The correlated topic model (CTM) is a hierarchical model that explicitly models the correlation of latent topics, allowing for a deeper understanding of relationships among topics (Blei and Lafferty 2007). The CTM extends the LDA model by relaxing the independence assumption of LDA. As in the LDA model, CTM is a mixture model and documents belong to a mixture of topics. CTM uses the same methodological approach as LDA, but it creates a more flexible modeling approach than LDA by replacing the Dirichlet distribution with a logistic normal distribution and explicitly incorporating a covariance structure among topics (Blei and Lafferty 2007). While this method creates a more computationally expensive topic modeling approach, it allows for more realistic modeling by allowing topics to be correlated. Additionally, Blei and Lafferty (2007) show that the CTM model outperforms LDA (Fig. 8.4).

As in the LDA model, K is the total number of topics, D is the total number of documents, and N is the total number of words in a document, where $W_{d,\,n}$ is an observed word. In the CTM model, η_d is the topic hyperparameter with mean μ and

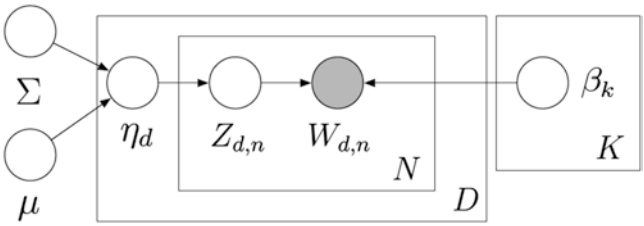


Fig. 8.4 Plate representation of the random variables in the CTM model (Blei et al. 2007, p. 21)

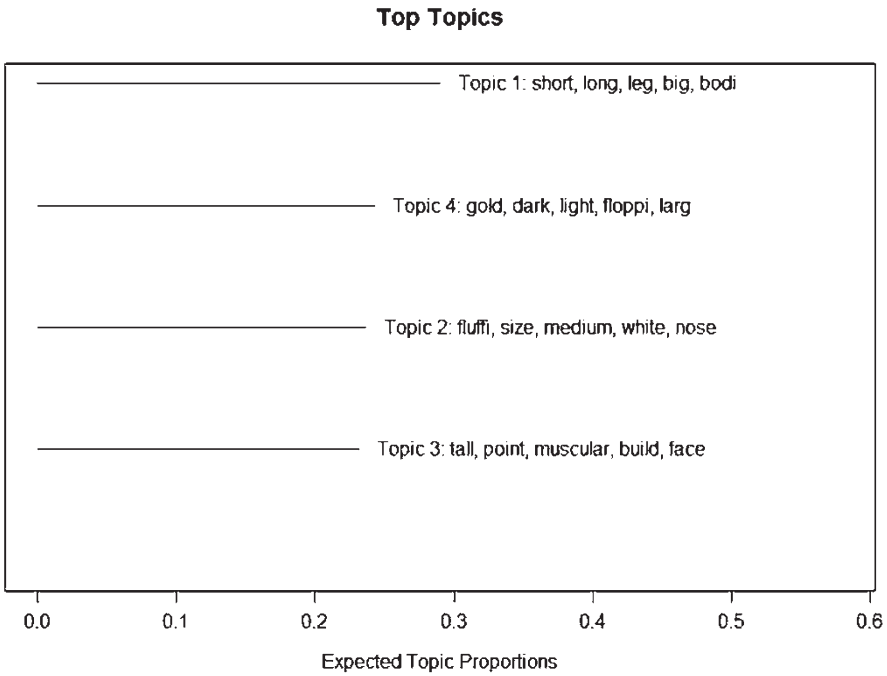


Fig. 8.5 Expected topic proportions of four categories in the CTM model with no covariates

covariance matrix $\sum$. Again, each topic is a distribution over terms, with topic assignments $Z_{d,n}$. Each document is a mixture of topics, with topic proportions θ_d, and each term is drawn from one of the topics, with topic assignments β_k. A fast variational inference algorithm is used to estimate the posterior distribution of the topics, because, as in LDA, the calculation is intractable. However, in practice, the computation is inefficient, particularly in comparison to LDA.

Using the data from the example, we build a CTM model with $k = 4$ topics. The top terms and expected topic proportions of this model are presented in Fig. 8.5. When considering the topic proportions, since we know that the four dogs are equally represented in the document collection, we would expect the topics to have the same expected proportions if the topics are dog specific. Based on the figure,

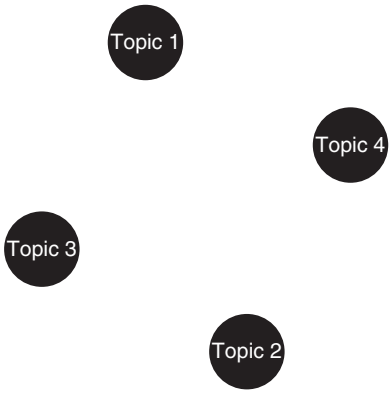


Fig. 8.6 CTM topic correlation plot

they do not appear to be topics based on the dog breeds. However, it does appear that Topic 1 could be about Dachshunds, Topic 2 about Golden Retrievers, Topic 3 about Bichon Frises, and Topic 4 about Great Danes. Topic 1 is the most prevalent expected topic, which contains *short*, *long*, and *leg* as the topic words. To try to explain the difference in topic proportions, we can look at a plot of the correlations among topics.

The CTM model has the advantage over the LDA model in that it models the correlations among topics. To investigate possible correlations, we can evaluate the correlations among topics and create an adjacency plot. Based on the adjacency plot in Fig. 8.6, in which no edges or straight lines connect the topic nodes, the four topics in the CTM model are not correlated.

8.4 Dynamic Topic Model (DT)

The dynamic topic model models topics in a sequentially ordered document collection to incorporate the evolution of topics over time by relaxing the exchangeability assumption of the LDA model (Blei and Lafferty 2006). The process involves splitting the data into smaller, time-dependent groups, such as by month or year. Dynamic topic models are built as an extension of the LDA model and thus do not model correlations among topics. The model uses the logistic normal distribution with mean α for each time period t (Fig. 8.7).

As in the LDA model, K is the total number of topics, D is the total number of documents, and N is the total number of words in a document. $W_{d,n}$ is an observed word. Additionally, α_t is the mean Dirichlet parameter α at time t. Each topic is a distribution over terms, with topic assignments $Z_{t,d,n}$. Each document is a mixture of topics, with topic proportions $\theta_{t,d}$, and each term is drawn from one of the topics, with topic assignments $\beta_{t,k}$ at time t. The model can use variational Kalman filtering or a variational wavelet regression to estimate the parameters of the DT model.

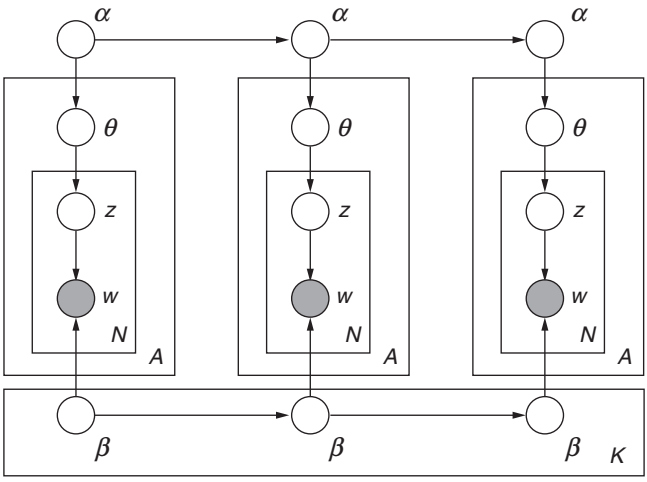


Fig. 8.7 Plate diagram of DT model (Blei and Lafferty 2006, p. 2)

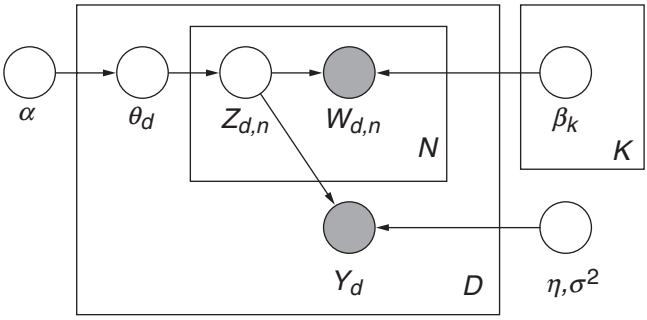


Fig. 8.8 Plate representation of the sLDA model (McAuliffe and Blei 2008, p. 3)

8.5 Supervised Topic Model (sLDA)

McAuliffe and Blei (2008) introduced the supervised latent Dirichlet allocation (sLDA), which is an extension of the LDA model with the use of labeled documents, as in the classification analysis covered in Chap. 9. The sLDA model has a class variable associated with each document, which serves as the response variable in the model (Fig. 8.8).

As in the LDA model, in the sLDA model, K is the total number of topics, D is the total number of documents, and N is the total number of words in a document, where $W_{d,n}$ is an observed word. Additionally, α is the Dirichlet parameter, η and σ^2 are response parameters, and y is the response variable. Each topic is a distribution over terms, with topic assignments $Z_{d,n}$. Each document is a mixture of topics, with topic proportions θ_d, and each term is drawn from one of the topics, with topic

assignments β_k. Rather than treat the parameters as random variables, the model treats them as unknown constants. As in the LDA model, a variational expectation-maximization (VEM) procedure is used for the model estimation.

8.6 Structural Topic Model (STM)

The structural topic model (STM) combines three common topic models to create a semiautomated approach to modeling topics, which can also incorporate covariates and metadata in the analysis of text (Roberts et al. 2014). Additionally, unlike the LDA model, topics in STM can be correlated. This model is particularly useful in the topical analysis of open-ended textual data, such as survey data.

Since STM allows for the addition of covariates, additional information from the data can be used in the model. Furthermore, effect estimation can be performed to investigate and compare selected covariates and topics. In particular, STM has the ability to account for topical content and prevalence, allowing us to compare groupings in the data. For instance, to consider content, we could compare the specific words that are used to describe the different types of dogs. We could also explore the topic's prevalence, or how often a topic occurs, for the different breeds.

STM is a mixture model, where each document can belong to a mixture of the designated k topics. Topic proportions, θ_d, can be correlated, and the topical prevalence can be impacted by covariates, X, through a regression model $\theta_d \sim \text{LogisticNormal}(X\gamma, \Sigma)$. This capability allows each document to have its own prior distribution over topics, rather than sharing a global mean. For each word, w, the topic, $z_{d,n}$, is drawn from a response-specific distribution. Conditioned on the topic, a word is chosen from a multinomial distribution over words with parameters, $\beta_{zd,n}$. The topical content covariate, U, allows word use within a topic to vary by content (Fig. 8.9).

We build an STM model with four topics and include the dog breed as a content variable. The top ten words in each topic in the STM model are shown in Fig. 8.10.

By incorporating the dog breed as a covariate, we can consider how the breeds vary for each of the four topics. Figure 8.11 shows the expected proportion of topics for each of the topics and dog breeds.

8.7 Decision Making in Topic Models

8.7.1 Assessing Model Fit and Number of Topics

Although there is no single, uniform measure for choosing the number of topics in building a topic model, several methods have been proposed to help the analyst decide on the number of topics, k. Two methods aim to minimize the metrics to determine the optimal number of topics. Cao Juan et al. (2009) uses minimum

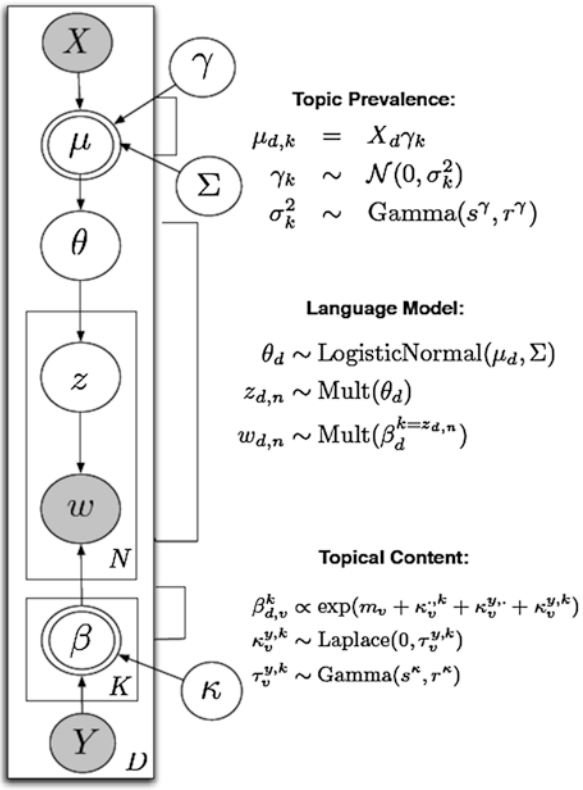


Fig. 8.9 Plate diagram representation of the structural topic model (Roberts et al. 2013, p. 2)

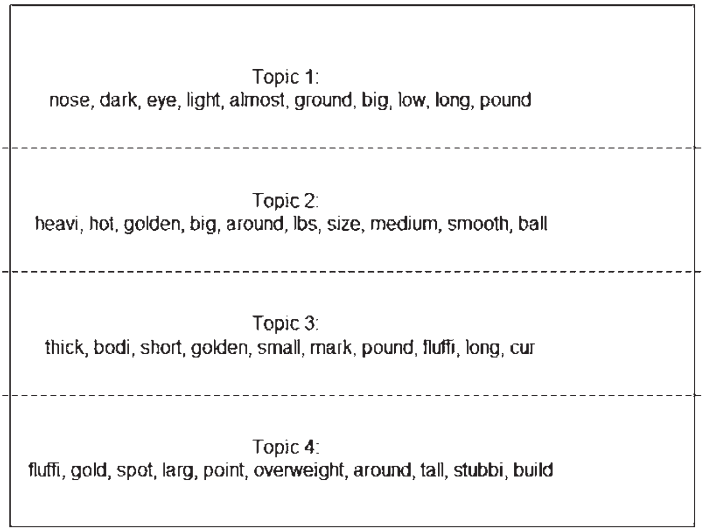


Fig. 8.10 Top ten terms in topics for STM model

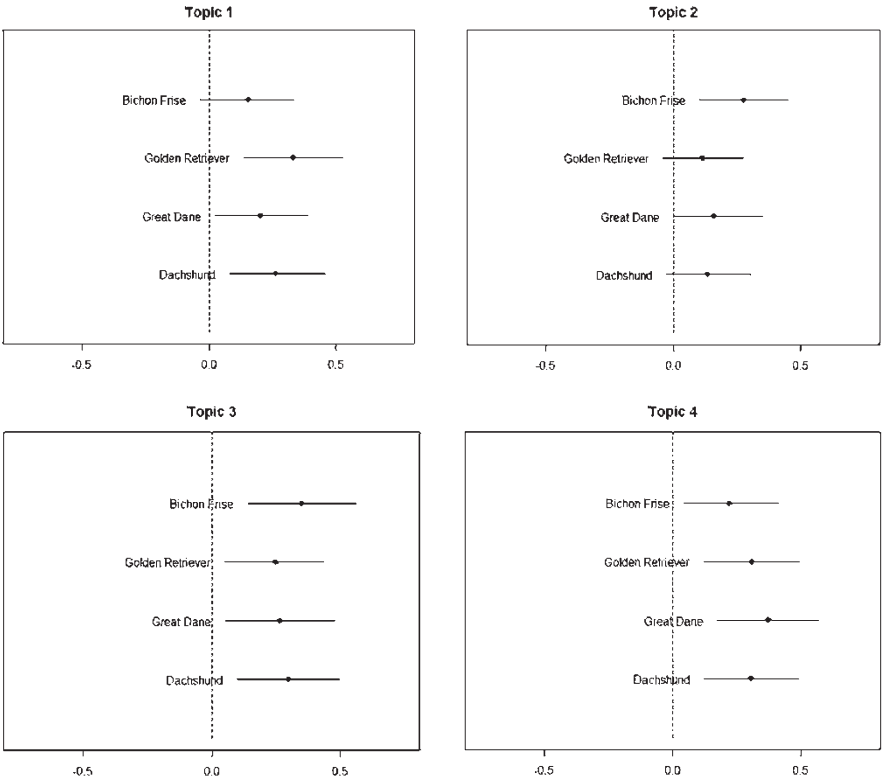


Fig. 8.11 Dog type content across topics

density measures to choose the number of topics. Arun et al. (2010) utilize a measure of divergence, where minimal divergence within a topic is preferred. Both methods use measures of distance to make decisions regarding k. On the other hand, Deveaud et al. (2014) utilize a measure maximizing the divergence across topics, and Griffiths and Steyvers (2004) maximize the log-likelihood of the data over different values of k. We use these four measures across 2–30 topics in building the LDA models, and the results are displayed in Fig. 8.12. Based on the figure, including five topics in an LDA model appears to be a good tradeoff between the four measures and is highlighted in red.

8.7.2 Model Validation and Topic Identification

Topic models can be supervised or unsupervised and, thus, can rely on either internal or external validity measures, depending on the type of data being used. Model validity and interpretability should go hand-in-hand in topic model analysis, and therefore, we will consider them together. We will focus on internal validity

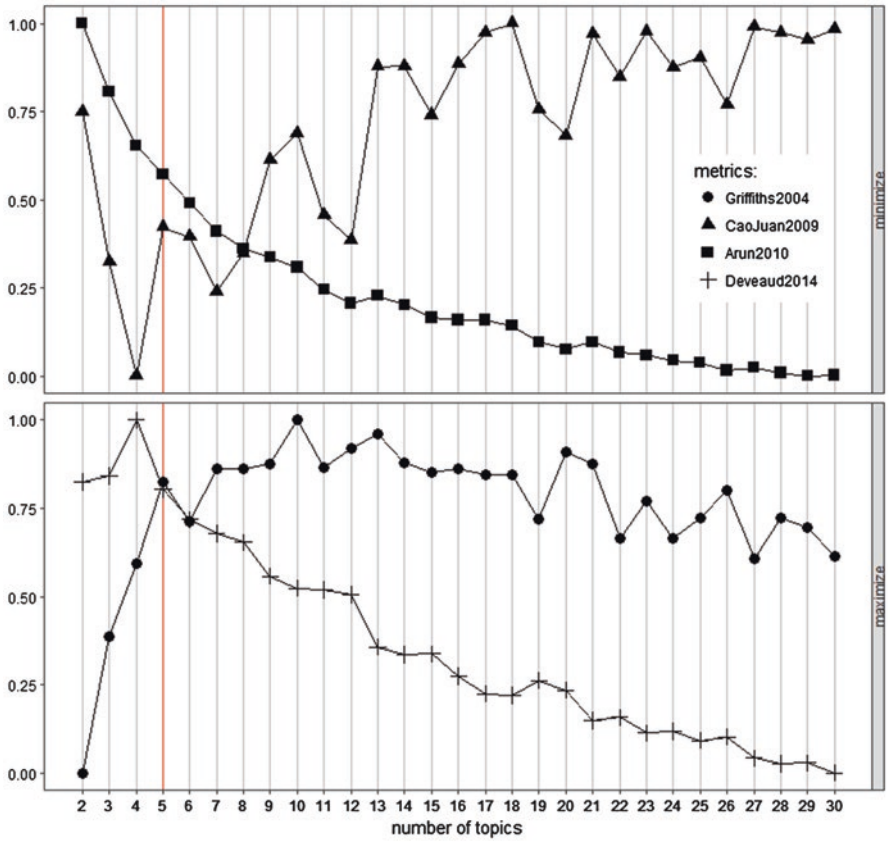
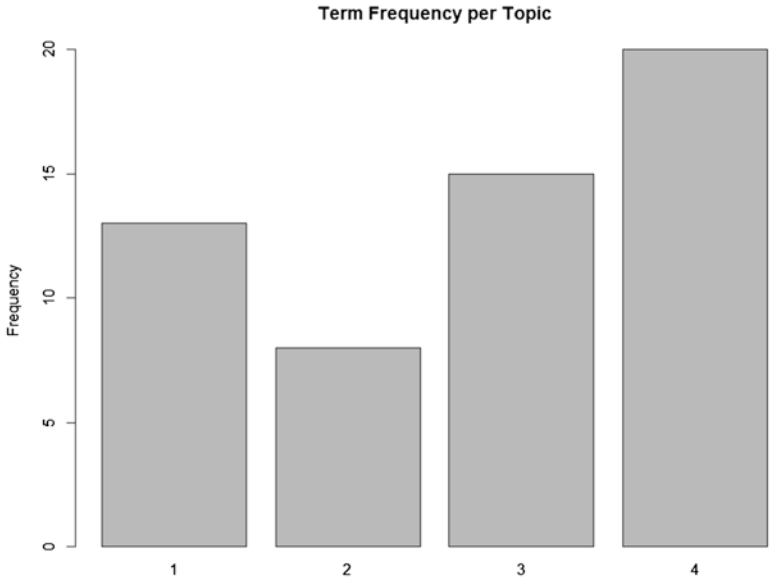


Fig. 8.12 Four measures across a number of topics, *k*, for 2–30 LDA topics

measures, since we performed an unsupervised LDA analysis. Mimno et al. (2011) suggest using topic size or the frequency of terms assigned to the topic as a good indicator of topic quality. Figure 8.13 displays the term frequency for the four-topic LDA solution and the five most probable terms in those topics.

Topics 4, 3, and 1, respectively, are the topics with the highest number of terms and are believed to be of higher quality than Topic 2. Topic models are built to identify latent topics existing in a document collection. In most cases, topic models are used to gain an understanding of the collection and to find ways to categorize and characterize documents. While the method is automatic, it requires interpretable output to be useful to the modeler. In this sense, it requires a combination of art and science. To this end, human coders are oftentimes used to evaluate the topics to determine if there is a natural label that can be assigned to the topic assignments from the topic model. For this reason, Chang et al. (2009) investigate the interpretability of models compared to their quantitative performance measures. They propose the use of word intrusion and topic intrusion methods, which involve presenting



long	ear	black	fluffi
weigh	white	tail	white
coat	long	short	weigh
lbs	weigh	coat	coat
ear	hair	larg	short

Fig. 8.13 Topic frequency and the five most probable terms per topic

the most probable terms and topics and an intruder. Then, human coders are instructed to identify the intruder. Another common approach is the use of coding by field experts in the relevant domain.

Alternatively, the model can be built on the sample, with a portion removed as a holdout sample. In doing so, two measures, perplexity and held-out likelihood, can be used to assess the model. Perplexity measures how well the model predicts the held-out sample. Perplexity values that are lower are preferred and indicate that the model is a good fit. We can also compute the log-likelihood of the held-out documents. The higher the log-likelihood, the better the model fit.

8.7.3 When to Use Topic Models

When determining if topic modeling should be used in the analysis, there are several considerations to keep in mind. First, most topic model approaches are unsupervised and assume that there is uncertainty in the documents. If the true topics of

the documents in the document collection are known, it may be more beneficial to apply a supervised analysis method, such as classification analysis, which is covered in Chap. 9. Topic models form soft clusters because they are typically mixed-membership models. The results of the analysis will produce the most likely topics to assign to documents and the most probable terms for each of the topics. If hard clusters are preferred and the topical content across terms and documents does not need to be considered simultaneously, cluster analysis can be used to cluster either terms or documents into hard clusters. Topic models, however, are particularly useful in making predictions. Since topic models are probabilistic models, predictions can be made about new documents based on the model.

Key Takeaways

- Topic models were created as an extension of a probabilistic version of latent semantic analysis.
- Topic models are unsupervised analysis methods that produce a topic distribution over terms and document distribution over topics.
- Popular topic modeling approaches include latent Dirichlet allocation, correlated topic models, dynamic topic models, supervised latent Dirichlet allocation, and structural topic models.

References

Arun, R., Suresh, V., Madhavan, C. V., & Murthy, M. N. (2010, June). On finding the natural number of topics with latent dirichlet allocation: Some observations. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 391–402). Berlin/Heidelberg: Springer.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM, 55*(4), 77–84.

Blei, D. M., & Lafferty, J. D. (2006). Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 113–120). ACM.

Blei, D. M., & Lafferty, J. D. (2007). A correlated topic model of science. *The Annals of Applied Statistics, 1*(1), 17–35.

Blei, D. M., & Lafferty J. D. (2009). Topic models. In A. Srivastava & M. Sahami (Eds.), *Text mining: Classification, clustering, and applications.* London: Chapman & Hall/CRC Data Mining and Knowledge Discovery Series.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2002). Latent Dirichlet allocation. In *Advances in neural information processing systems* (pp. 601–608). Cambridge, MA: MIT Press.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research, 3*, 993–1022.

Blei, D., Carin, L., & Dunson, D. (2010). Probabilistic topic models. *IEEE Signal Processing Magazine, 27*(6), 55–65.

Blei, David M., & Lafferty, J.D. (2007). A Correlated Topic Model of Science. *The Annals of Applied Statistics. 1*(1): 17–35.

Cao, J., Xia, T., Li, J., & Zhang Y., & Tang, S. (2009). A density-based method for adaptive lDA model selection. *Neurocomputing — 16th European Symposium on Artificial Neural Networks 2008, 72*(7–9), 1775–1781.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems* (pp. 288–296). Cambridge, MA: MIT Press.

Deveaud, R., SanJuan, E., & Bellot, P. (2014). Accurate and effective latent concept modeling for ad hoc information retrieval. *Document numérique, 17*(1), 61–84.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences, 101*(suppl 1), 5228–5235.

Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007). Topics in semantic representation. *Psychological Review, 114*(2), 211–244.

Hofmann, T. (1999, July). Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 289–296).

Mcauliffe, J. D., & Blei, D. M. (2008). Supervised topic models. In *Advances in neural information processing systems* (pp. 121–128). Cambridge, MA: MIT Press.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., & McCallum, A. (2011, July). Optimizing semantic coherence in topic models. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 262–272). Association for Computational Linguistics.

Roberts, M. E., Stewart, B. M., Tingley, D., & Airoldi, E. M. (2013, January). The structural topic model and applied social science. In *Advances in neural information processing systems workshop on topic models: computation, application, and evaluation* (pp. 1–20).

Roberts, M., et al. (2014). Structural topic models for open-ended survey responses. *American Journal of Political Science, 58*, 1064–1082.

Yi, X., & Allan, J. (2009, April). A comparative study of utilizing topic models for information retrieval. In *European conference on information retrieval* (pp. 29–41). Berlin/Heidelberg: Springer.

Further Reading

To learn more about topic models, see Blei (2012), Blei and Lafferty (2009), and Griffiths et al. (2007).