10

Kernels for text

The last decade has seen an explosion of readily available digital text that has rendered attempts to analyse and classify by hand infeasible. As a result automatic processing of natural language text documents has become a main research interest of Artificial Intelligence (AI) and computer science in general. It is probably fair to say that after multivariate data, natural language text is the most important data format for applications. Its particular characteristics therefore deserve specific attention.

We will see how well-known techniques from Information Retrieval (IR), such as the rich class of vector space models, can be naturally reinterpreted as kernel methods. This new perspective enriches our understanding of the approach, as well as leading naturally to further extensions and improvements. The approach that this perspective suggests is based on detecting and exploiting statistical patterns of words in the documents. An important property of the vector space representation is that the primal–dual dialectic we have developed through this book has an interesting counterpart in the interplay between term-based and document-based representations.

The goal of this chapter is to introduce the Vector Space family of kernel methods highlighting their construction and the primal–dual dichotomy that they illustrate. Other kernel constructions can be applied to text, for example using probabilistic generative models and string matching, but since these kernels are not specific to natural language text, they will be discussed separately in Chapters 11 and 12.

10.1 From bag of words to semantic space

Although the task of automatically analysing the meaning of a document falls under the heading of natural language processing, it was the IR community that developed the relatively simple representation that is today most commonly used to assess the topic of a document. This representation is known as the vector space model (VSM) or the 'bag-of-words' approach. It is this technique that forms the basis of the kernels developed in this chapter. As mentioned above, other representations of text are certainly possible including those relying on a string level analysis or on probabilistic models. These approaches will be mentioned as special cases in later chapters that are mainly concerned with other data formats.

One of the aims of this chapter is to show that by using the VSM to create kernels, it is possible to extend its applicability beyond IR to the full range of tasks that can be solved using the kernel approach, including correlation analysis, novelty-detection, classification, ranking, clustering and so on. Some of these tasks reappear with different names in the context of document analysis. For example document classification is often referred to as *categorisation*, on-line classification as *filtering*, while novelty detection is known as *new topic detection.*

10.1.1 Representing text

The simplest possible version of the VSM is the representation of a document as a *bag-of-words*. A bag is a set in which repeated elements are allowed, so that not only the presence of a word but also its frequency is taken into account. Hence, a document is represented by the words it contains, with the ordering of the words and punctuation being ignored. This implies that some grammatical information is lost, since there is no representation of the word ordering. This furthermore implies that phrases are also broken into their constituent words and hence the meaning of a phrase such as 'bread winner' is lost. We begin with some definitions.

Definition 10.1 [Vector space model] *Words* are any sequence of letters from the basic alphabet separated by punctuation or spaces. We use *term* synonymously with word. A dictionary can be defined by some permanent predefined set of terms, but in practice we only need to consider those words actually appearing in the documents being processed. We refer to this full set of documents as the *corpus* and the set of terms occurring in the corpus as the *dictionary*. Hence, we can view a document as a bag of terms or *bag-of-words*. We can represent a bag as a vector in a space in which each dimension is associated with one term from the dictionary

$$\boldsymbol{\phi} : d \longmapsto \boldsymbol{\phi}(d) = (\mathrm{tf}(t_1, d), \mathrm{tf}(t_2, d), \ldots, \mathrm{tf}(t_N, d)) \in \mathbb{R}^N,$$

where $\mathrm{tf}(t_i, d)$ is the frequency of the term t_i in the document d. Hence, a document is mapped into a space of dimensionality N being the size of the dictionary, typically a very large number. ■

Despite the high-dimensionality of the dictionary, the vector associated with a given document is sparse, i.e. has most of its entries equal to 0, since it will contain only very few of the vast number of possible words.

Definition 10.2 [Document–term matrix] The document–term matrix of a corpus is the matrix whose rows are indexed by the documents of the corpus and whose columns are indexed by the terms. The (i, j)th entry gives the frequency of term t_j in document d_i

$$\mathbf{D} = \begin{pmatrix} \mathrm{tf}(t_1, d_1) & \cdots & \mathrm{tf}(t_N, d_1) \\ \vdots & \ddots & \vdots \\ \mathrm{tf}(t_1, d_\ell) & \cdots & \mathrm{tf}(t_N, d_\ell) \end{pmatrix}.$$

The term–document matrix is the transpose $\mathbf{D}'$of the document–term matrix. The term-by-term matrix is given by $\mathbf{D}'\mathbf{D}$ while the document-by-document matrix is $\mathbf{D}\mathbf{D}'$. ■

Note that the document–term matrix is simply the data matrix $\mathbf{X}$ we have used consistently throughout this book. Hence, the term-by-term matrix is ℓ times the covariance matrix, while the document-by-document matrix is the corresponding kernel matrix.

Remark 10.3 [Interpreting duality] An interesting feature of the VSM will be an additional interpretation of the duality between representations of kernel algorithms. Here the dual representation corresponds to a document view of the problem, while the primal description provides a term view. In the same way that a document can be seen as the counts of the terms that appear in it, a term can be regarded as the counts of the documents in which it appears. These two views are represented by the rows of a document–term matrix with the rows of its transpose giving the representation of terms. ■

Since it is often the case that the corpus size is less than the number of terms or dictionary size, it is sometimes useful to compute in a document-based or dual representation, whereas for intuitive and interpretational purposes it is usually better to work in the term-based or primal representation.

We can exploit the power of duality in kernel methods by stating the problem in the intuitive term representation, and then dualising it to provide a document-based implementation.

10.1.2 Semantic issues

The representation of documents provided by the VSM ignores any semantic relation between words. One important issue is to improve the vector space representation to ensure that documents containing semantically equivalent words are mapped to similar feature vectors.

For example synonymous words give two ways of saying the same thing, but are assigned distinct components. Hence, the VSM despite retaining enough information to take account of this similarity, is unable to do so without extra processing. We will present a range of methods that aim to address this and related problems.

On the other hand, the case of homonymy when a single word has two distinct meanings is an example that the VSM will be less able to handle since it has thrown away the contextual information that could disambiguate the meaning. Despite this, some context can be derived from the statistics of the words in the document. We begin, however, by mentioning two simpler operations that can improve the quality of the embedding.

The first is to apply different weights w_i to each coordinate or equivalently to give varying weights to the terms. Perhaps the simplest example of this is the removal of uninformative terms such as 'and', 'of', 'the', and so on. This is equivalent to assigning a weight of 0 to these coordinates. This technique was pioneered in the IR literature, where such words are referred to as *stop words*, with the complete set known as the *stop list.* We consider more general weighting schemes in the next section.

The second effect that can cause problems is the influence of the length of a document. Clearly, the longer a document the more words it contains and hence, the greater the norm of its associated vector. If the length of the document is not relevant for the tasks we are trying to solve, for example categorisation by topic, it makes sense to remove this effect by normalising the embedding vectors. Using the technique developed in Chapter 5 and given in (5.1), we define a kernel that discards this information as follows

$$\hat{\kappa}(\mathbf{x},\mathbf{y}) = \left\langle \frac{\boldsymbol{\phi}(\mathbf{x})}{\|\boldsymbol{\phi}(\mathbf{x})\|}, \frac{\boldsymbol{\phi}(\mathbf{y})}{\|\boldsymbol{\phi}(\mathbf{y})\|} \right\rangle = \frac{\kappa(\mathbf{x},\mathbf{y})}{\sqrt{\kappa(\mathbf{x},\mathbf{x})\kappa(\mathbf{y},\mathbf{y})}}.$$

Typically the normalisation is implemented as either the first transformation or as the final embedding. We will not consider the normalisation for each of the different semantic embeddings but assume that when required it is included as a final stage.

Remark 10.4 [On successive embeddings] The operations can be performed in sequence, creating a series of successive embeddings, each of which adds some additional refinement to the semantics of the representation, for example term weighting followed by normalisation. This chapter will describe a number of operations designed to refine the semantic quality of the representation. Any of these refinements can be included in a sequence. As described in the last chapter, if we compose these successive embeddings we create a single map that incorporates different aspects of domain knowledge into the representation. ■

10.2 Vector space kernels

Given a document, we have seen how it can be represented by a row vector

$$\boldsymbol{\phi}(d) = (\mathrm{tf}(t_1, d), \mathrm{tf}(t_2, d), \dots, \mathrm{tf}(t_N, d)) \in \mathbb{R}^N,$$

in which each entry records how many times a particular term is used in the document. Typically $\boldsymbol{\phi}(d)$ can have tens of thousands of entries, often more than the number of training examples. Nonetheless, for a particular document the representation is extremely sparse, having only relatively few non-zero entries. This preliminary embedding can then be refined by successive operations that we will examine later in the chapter.

As indicated in Definition 10.2 we can create a document–term matrix $\mathbf{D}$ whose rows are the vectors associated with the documents of the corpus. It is common in IR to work with the transpose or term–document matrix $\mathbf{D}'$, but we have chosen to maintain consistency with our use in earlier chapters of the matrix $\mathbf{X}$, whose rows are the feature vectors of the training examples. Hence, there is a direct correspondence between $\mathbf{X}$ and $\mathbf{D}$ where features become terms and examples become documents.

Definition 10.5 [Vector space kernel] Within the VSM we can create a kernel matrix as

$$\mathbf{K} = \mathbf{D}\mathbf{D}'$$

corresponding to the *vector space kernel*

$$\kappa(d_1, d_2) = \langle \boldsymbol{\phi}(d_1), \boldsymbol{\phi}(d_2) \rangle = \sum_{j=1}^{N} \mathrm{tf}(t_j, d_1)\, \mathrm{tf}(t_j, d_2).$$

■

Cost of the computation In order to compute the kernel κ we must first convert the document into a list of the terms that they contain. This process is known as *tokenisation.* It has been a fundamental processing technique of computer science since the design of the first compilers. Each term encountered in the corpus is assigned its own unique number. This ensures that the list of terms in a document can be reordered into ascending term order together with an associated frequency count. In this way the document is converted into a list $L(d)$ rather than the impractically long explicit vector $\boldsymbol{\phi}(d)$. It is now a relatively simple and efficient task to compute

$$\kappa(d_1, d_2) = A(L(d_1), L(d_2)),$$

using the lists $L(d_1)$ and $L(d_2)$ as inputs, where the algorithm $A(\cdot,\cdot)$ traverses the lists, computing products of frequencies whenever the term numbers match. In summary, the computation of the kernel does not involve explicit evaluation of the feature vectors $\boldsymbol{\phi}(d)$, but uses an intermediate representation of a list $L(d)$ of terms. Hence, although we are working in a space that is typically of very high dimension, the computation of the projections and subsequent inner product evaluation can be implemented in a time proportional to the sum of the lengths of the two documents

$$O(|d_1| + |d_2|).$$

We should contrast this approach with systems that require explicit weights to be maintained for each component.

Remark 10.6 [Nonlinear embeddings] Throughout this chapter we will restrict ourselves to considering linear transformations of the basic VSM, always laying emphasis on the power of capturing important domain knowledge. However, it is of course possible to consider nonlinear embeddings using standard kernel constructions. For example a polynomial kernel over the normalised bag-of-words representation

$$\bar{\kappa}(d_1, d_2) = (\kappa(d_1, d_2) + 1)^d = (\langle \boldsymbol{\phi}(d_1), \boldsymbol{\phi}(d_2)\rangle + 1)^d,$$

uses all n-tuples of words for $0 \leq n \leq d$ as features. The same approach can be used for any of the vector space kernels using polynomial kernels or other kernel constructions such as the Gaussian kernel. ■

10.2.1 Designing semantic kernels

As indicated above the bag-of-words representation has many shortcomings, partly the neglecting of the word order but also of the semantic content of the words themselves. In order to address this second omission, we will consider transformations of the document vectors $\boldsymbol{\phi}(d)$. The VSM considers only the simplest case of linear transformations of the type $\tilde{\boldsymbol{\phi}}(d) = \boldsymbol{\phi}(d)\,\mathbf{S}$, where $\mathbf{S}$ is a matrix that could be diagonal, square or, in general, any $N \times k$ matrix. Using this transformation the corresponding kernel takes the form

$$\tilde{\kappa}(d_1, d_2) = \boldsymbol{\phi}(d_1)\,\mathbf{S}\mathbf{S}'\boldsymbol{\phi}(d_2)' = \tilde{\boldsymbol{\phi}}(d_1)\,\tilde{\boldsymbol{\phi}}(d_2)'.$$

That this is a kernel follows directly from the explicit construction of a feature vector. We refer to $\mathbf{S}$ as the *semantic matrix*. Different choices of the matrix $\mathbf{S}$ lead to different variants of the VSMs. As indicated in Remark 10.4 we will often create $\mathbf{S}$ as a composition of several stages. We will examine some of these in the coming subsections. Hence, we might define

$$\mathbf{S} = \mathbf{RP},$$

where $\mathbf{R}$ is a diagonal matrix giving the term weightings or relevances, while $\mathbf{P}$ is a *proximity* matrix defining the semantic spreading between the different terms of the corpus.

Term weighting As mentioned in the previous section not all words have the same importance in establishing the topic of a document. Indeed, in IR the so-called stop words are removed before the analysis starts. The entropy or the frequency of a word across the documents in a corpus can be used to quantify the amount of information carried by a word. This is an 'absolute' measure in the sense that it takes no account of particular topics or tasks. For categorisation one could also define a relative measure of the importance of a word with respect to the given topic, such as the mutual information. Here we consider an absolute measure known as *idf* that weights terms as a function of their *inverse document frequency*. Suppose that there are ℓ documents in the corpus and let $\mathrm{df}(t)$ be the number of documents containing the term t. The usual measure of inverse document frequency for a term t is then given by

$$w(t) = \ln\left(\frac{\ell}{\mathrm{df}(t)}\right).$$

Implicit in this weighting is the possibility of stop words, since if a term is contained in every document then $\mathrm{df}(t) = \ell$ and $w(t) = 0$, effectively excluding the term from the dictionary and hence reducing the dimensionality of the feature space. For efficiency reasons it is still preferable to create an explicit stop list of terms that should a priori be excluded from the features.

The use of the logarithmic function ensures that none of the weights can become too large relative to the weight of a term that occurs in roughly half of the documents. This is occasionally further controlled by removing words that occur in fewer than one or two documents, since they are viewed as too exceptional to be of use in analysing future documents.

Given a term weighting $w(t)$ whether defined by the *idf* rule or some alternative scheme, we can now define a new VSM by choosing the matrix $\mathbf{R}$ to be diagonal with entries

$$\mathbf{R}_{tt} = w(t).$$

The associated kernel simply computes the inner product

$$\tilde{\kappa}(d_1, d_2) = \boldsymbol{\phi}(d_1)\mathbf{R}\mathbf{R}'\boldsymbol{\phi}(d_2)' = \sum_t w(t)^2 \operatorname{tf}(t, d_1) \operatorname{tf}(t, d_2),$$

again clearly implementable by a weighted version A_w of the algorithm A:

$$\tilde{\kappa}(d_1, d_2) = A_w(L(d_1), L(d_2)).$$

The evaluation of this kernel involves both term frequencies and inverse document frequencies. It is therefore often referred to as the *tf–idf* representation. Since these measures do not use label information, they could also be estimated from an external, larger unlabelled corpus that provides background knowledge for the system.

Term proximity matrix The *tf–idf* representation implements a downweighting of irrelevant terms as well as highlighting potentially discriminative ones, but nonetheless is still not capable of recognising when two terms are semantically related. It is therefore not able to establish a connection between two documents that share no terms, even when they address the same topic through the use of synonyms. The only way that this can be achieved is through the introduction of semantic similarity between terms. Within the VSM this requires that a proximity matrix $\mathbf{P}$ has non-zero off-diagonal entries $\mathbf{P}_{ij} > 0$ when the term i is semantically related to the term j. Given such a matrix, the vector space kernel

$$\tilde{\kappa}(d_1, d_2) = \boldsymbol{\phi}(d_1)\mathbf{P}\mathbf{P}'\boldsymbol{\phi}(d_2)' \tag{10.1}$$

corresponds to representing a document by a less sparse vector $\boldsymbol{\phi}(d)\mathbf{P}$ that has non-zero entries for all terms that are semantically similar to those present in the document $\dot{d}$. This is closely related to a technique from IR known as ‘query expansion’, where a query is expanded to include not only the actual query terms but any terms that are semantically related. Alternatively, we can view the matrix $\mathbf{PP}'$ as encoding a semantic strength between terms. This is perhaps most clearly apparent if we let $\mathbf{Q} = \mathbf{PP}'$ and expand the equation (10.1)

$$\tilde{\kappa}(d_1, d_2) = \sum_{i,j} \boldsymbol{\phi}(d_1)_i \mathbf{Q}_{ij} \boldsymbol{\phi}(d_2)_j ,$$

so that we can view $\mathbf{Q}_{ij}$ as encoding the amount of semantic relation between terms i and j. Note that defining the similarity by inferring $\mathbf{Q}$ requires the additional constraint that $\mathbf{Q}$ be positive semi-definite, suggesting that defining $\mathbf{P}$ will in general be more straightforward.

Remark 10.7 [Stemming] One method of detecting some semantic similarities is known as *stemming*. This technique, implemented as part of the tokeniser, automatically removes inflexions from words, hence ensuring that different forms of the same word are treated as equivalent terms. For example ‘structural’, ‘structure’ and ‘structured’ would all be mapped to the common stem ‘structure’ and so be treated as the same term. This effectively performs a dimension reduction, while linking the different forms of the word as semantically identical. We can implement this within the VSM by setting $\mathbf{P}_{ij} = s^{-1/2}$, for all terms i, j that are reduced to the same stem, where s is the number of distinct terms involved. Of course this is just of theoretical interest as in practice the terms are all mapped to a single stemmed term in the list $L(d)$. ■

10.2.2 Designing the proximity matrix

Remark 10.7 suggests a way of defining the proximity matrix $\mathbf{P}$ by putting non-zero entries between those terms whose semantic relation is apparent from their common stem. The question remains of how we can obtain more general semantic matrices that link synonymous or related terms that do not share a common core. We now give a series of methods for obtaining semantic relationships or learning them directly from a corpus be it the training corpus or a separate set of documents. Though we present the algorithms in a term-based representation, we will in many cases show how to implement them in dual form, hence avoiding the explicit computation of the matrix $\mathbf{P}$.

Explicit construction of the proximity matrix Perhaps the most natural method of incorporating semantic information is by inferring the relatedness of terms from an external source of domain knowledge. In this section we briefly describe one such approach. In the next section we will see how we can use co-occurrence information to infer semantic relations between terms, hence avoiding the need to make use of such external information.

A semantic network such as Wordnet provides a way to obtain term-similarity information. A semantic network encodes relationships between words of a dictionary in a hierarchical fashion, where the more general terms occur higher in the tree structure, so that for example 'spouse' occurs above 'husband' and 'wife', since it is their hypernym. We can use the distance between two terms in the hierarchical tree provided by Wordnet to give an estimate of their semantic proximity. This can then be used to modify the metric of the vector space of the bag-of-words feature space.

In order to introduce this information into the kernel, we can handcraft the matrix $\mathbf{P}$ by setting the entry $\mathbf{P}_{ij}$ to reflect the semantic proximity between the terms i and j. A simple but effective choice is to set this equal to the inverse of their distance in the tree, that is the length of the shortest path connecting them. The use of this semantic proximity gives rise to the vector space kernel

$$\tilde{\kappa}(d_1, d_2) = \boldsymbol{\phi}(d_1)\,\mathbf{P}\mathbf{P}'\boldsymbol{\phi}(d_2)'.$$

Generalised vector space model An early attempt to overcome the limitations of the VSMs by introducing semantic similarity between terms is known as the generalised VSM or GVSM. This method aims at capturing term–term correlations by looking at co-occurrence information. Two terms are considered semantically related if they frequently co-occur in the same documents. Indeed this simple observation forms the basis of most of the techniques we will develop in this chapter. This means that two documents can be seen as similar even if they do not share any terms, but the terms they contain co-occur in other documents. The GVSM method represents a document by a vector of its similarities with the different documents in the corpus, in other words a document is represented by the embedding

$$\boldsymbol{\phi}(d) = \boldsymbol{\phi}(d)\,\mathbf{D}',$$

where $\mathbf{D}$ is the document–term matrix, equivalent to taking $\mathbf{P} = \mathbf{D}'$. This definition does not make immediately clear that it implements a semantic similarity, but if we compute the corresponding kernel

$$\tilde{\kappa}(d_1, d_2) = \boldsymbol{\phi}(d_1)\,\mathbf{D}'\mathbf{D}\boldsymbol{\phi}(d_2)'.$$

Now we can observe that the matrix $\mathbf{D}'\mathbf{D}$ has a nonzero (i, j)th entry if and only if there is a document in the corpus in which the ith and jth terms co-occur, since

$$\left(\mathbf{D}'\mathbf{D}\right)_{ij} = \sum_d \mathrm{tf}(i, d)\,\mathrm{tf}(j, d).$$

So two terms co-occurring in a document are considered related with the strength of the relationship given by the frequency and number of their co-occurrences. If there are fewer documents than terms, this has the effect of dimensionality reduction by mapping from the vectors indexed by terms to a lower-dimensional space indexed by the documents of the corpus. Though appealing, the GVSM is too naive in its use of the co-occurrence information. The coming subsections examine more subtle uses of this information to create more refined semantics.

Latent semantic kernels The IR community developed a very effective vector space representation of documents that can capture semantic information through the use of co-occurrence information known as latent semantic indexing (LSI). Conceptually, LSI follows the same approach as GVSMs, only that the technique used to extract the semantic information from the co-occurrences is very different making use as it does of singular value decomposition (SVD). We will see that this amounts to a special choice of the matrix $\mathbf{P}$ with some useful properties.

Recall from Section 6.1 that the SVD of the matrix $\mathbf{D}'$ is

$$\mathbf{D}' = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}'$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix of the same dimensions as $\mathbf{D}$, and $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices whose columns are the eigenvectors of $\mathbf{D}'\mathbf{D}$ and $\mathbf{D}\mathbf{D}'$ respectively. LSI now projects the documents into the space spanned by the first k columns of $\mathbf{U}$, using these new k-dimensional vectors for subsequent processing

$$d \longmapsto \boldsymbol{\phi}(d)\,\mathbf{U}_k,$$

where $\mathbf{U}_k$ is the matrix containing the first k columns of $\mathbf{U}$. This choice can be motivated by recalling that the eigenvectors define the subspace that minimises the sum-squared differences between the points and their projections, that is it defines the subspace with minimal sum-squared residuals as spelled out in Proposition 6.12. Hence, the eigenvectors for a set of documents can be viewed as concepts described by linear combinations of terms chosen in such a way that the documents are described as accurately as possible using only k such concepts.

Note that terms that co-occur frequently will tend to align in the same eigenvectors, since SVD merges highly correlated dimensions in order to define a small number of dimensions able to reconstruct the whole feature vector. Hence, the SVD is using the co-occurrence information in a sophisticated algorithm that can maximise the amount of information extracted by a given number of dimensions.

The definition of LSI shows that it is identical to performing PCA in the feature space as described in (6.12). Hence, the new kernel becomes that of kernel PCA

$$\tilde{\kappa}(d_1, d_2) = \boldsymbol{\phi}(d_1)\,\mathbf{U}_k\mathbf{U}_k'\boldsymbol{\phi}(d_2)',$$

showing that the matrix $\mathbf{P}$ has in this case been chosen equal to $\mathbf{U}_k$. If we compare this with the GVSM we can write its projection as

$$\kappa_{\text{GSVM}}(d_1, d_2) = \boldsymbol{\phi}(d_1)\,\mathbf{D}'\mathbf{D}\boldsymbol{\phi}(d_2)' = \boldsymbol{\phi}(d_1)\,\mathbf{U}'\boldsymbol{\Sigma}\mathbf{U}\boldsymbol{\phi}(d_2)'.$$

Hence, there are two adaptations that have taken place when compared to GVSM. Firstly, dimensionality reduction has been introduced through the restriction to k eigenvectors, and secondly the removal of the matrix $\boldsymbol{\Sigma}$ has ensured that the projections of the data are orthonormal.

The fact that LSI is equivalent to PCA also implies that we can implement the algorithm in the dual representation to obtain from (6.12) the evaluation of the projection as follows.

Computation 10.8 [Latent semantic kernels]Latent semantic kernels are implemented by projecting onto the features

$$\boldsymbol{\phi}(d)\mathbf{U}_k = \left(\lambda_i^{-1/2}\sum_{j=1}^{\ell}(\mathbf{v}_i)_j\,\kappa(d_j, d)\right)_{i=1}^{k},$$

where κ is the base kernel, and $\lambda_i, \mathbf{v}_i$ are the eigenvalue, eigenvector pairs of the kernel matrix. ■

The base kernel can now be chosen as the bag-of-words kernel or more generally a complex kernel involving term weightings or even the polynomial construction. For this reason we refer to this dual construction as *latent semantic kernels (LSK)*.

Again we see the sculpting of the feature space through a series of transformations each using additional domain knowledge to further refine the semantics of the embedding. If we wish to represent the LSKs with a proximity matrix we can do so by taking

$$\mathbf{P} = \mathbf{U}_k \mathbf{U}'$$

since this leads to an identical kernel, while the matrix $\mathbf{P}$ is now square and so defines connection strengths between the different terms. Notice how as k increases, the matrix tends to the identity, returning to the treatment of all terms as semantically distinct. Hence, the value of k controls the amount of semantic smoothing that is introduced into the representation.

Exploiting multilingual corpora Aligned corpora are formed of pairs of documents that are translations of each other in two languages. They are examples of the paired training sets introduced when we considered canonical correlation analysis in Section 6.5. We can treat the translation as a complex label for the first document, or consider the two versions as two views of the same object.

In this way we can use a bilingual corpus to learn a semantic mapping that is useful for tasks that have nothing to do with the second language. The translations can be seen as complex labels that enable us to elicit refined semantic mappings that project the documents onto key directions that are able to remove semantically irrelevant aspects of the representation.

LSKs can be used to derive a more refined language independent representation. This is achieved by concatenating the two versions of the document into a single bilingual text. LSI is then applied to create a set of k semantic dimensions. A new document or query in only one language is projected using the projections derived from the eigenvectors obtained from the concatenated documents.

The latent semantic approach looks for directions of maximum variance but is restricted by its concatenation of the feature spaces of the two languages. An alternative method for creating semantic features is to use kernel CCA. This method treats the two versions of the document as two views of the same semantic object. It therefore finds independent projections for the two languages that map to a common semantic space. This space can be used for further document analysis.

Remark 10.9 [Cross-lingual information retrieval] Cross-lingual information retrieval (CLIR) is concerned with the task of retrieving documents in one language with queries from a different language. The technique is intended for users who have a passive knowledge of a language enabling them to read documents, but not sufficient expertise to choose a suitable query for the information they require. The semantic space provides an ideal representation for performing CLIR. ■

Semantic diffusion kernels We would like to continue our exploitation of the dual perspective between document-based and term-based representations. If we regard documents as similar that share terms, we have seen how we can equally well regard as related terms that co-occur in many documents. This suggests we might extend the similarity implications even further by, for example, regarding documents that share terms that co-occur as similar – this is the effect of using the co-occurrence analysis in further processing. Extrapolating this interaction suggests the introduction of a recurrence relation. For this section we will denote the matrix $\mathbf{D}'\mathbf{D}$ with $\mathbf{G}$ and as usual $\mathbf{D}\mathbf{D}'$ is the base kernel matrix $\mathbf{K}$. Consider refined similarity matrices $\hat{\mathbf{K}}$ between documents and $\hat{\mathbf{G}}$ between terms defined recursively by

$$\hat{\mathbf{K}} = \mu\mathbf{D}\hat{\mathbf{G}}\mathbf{D}' + \mathbf{K} \quad \text{and} \quad \hat{\mathbf{G}} = \mu\mathbf{D}'\hat{\mathbf{K}}\mathbf{D} + \mathbf{G}. \tag{10.2}$$

We can interpret this as augmenting the similarity given by $\mathbf{K}$ through indirect similarities measured by $\mathbf{G}$ and vice versa. The factor $\mu < \|\mathbf{K}\|^{-1}$ ensures that the longer range effects decay exponentially (recall that $\|\mathbf{K}\| = \max_{\lambda\in\lambda(\mathbf{K})}|\lambda|$ denotes the spectral norm of $\mathbf{K}$).

Recall Definition 9.34 of the von Neumann diffusion kernel $\bar{\mathbf{K}}$ over a base kernel $\mathbf{K}_1$ in Section 9.4

$$\bar{\mathbf{K}} = (\mathbf{I} - \lambda\mathbf{K}_1)^{-1}$$

We can characterise the solution of the above recurrences in the following proposition.

Proposition 10.10 *Provided $\mu < \|\mathbf{K}\|^{-1} = \|\mathbf{G}\|^{-1}$, the kernel $\hat{\mathbf{K}}$ that solves the recurrences (10.2) is $\mathbf{K}$ times the von Neumann kernel over the base kernel $\mathbf{K}$, while the matrix $\hat{\mathbf{G}}$ satisfies*

$$\hat{\mathbf{G}} = \mathbf{G}(\mathbf{I} - \mu\mathbf{G})^{-1}.$$

The proof of the proposition is given in Appendix A.3.

Note that we can view $\hat{\mathbf{K}}(\mu)$ as a kernel based on the proximity matrix $\mathbf{P} = \sqrt{\mu\hat{\mathbf{G}} + \mathbf{I}}$ since

$$\mathbf{D}\mathbf{P}\mathbf{P}'\mathbf{D}' = \mathbf{D}(\mu\hat{\mathbf{G}} + \mathbf{I})\mathbf{D}' = \mu\mathbf{D}\hat{\mathbf{G}}\mathbf{D}' + \mathbf{K} = \hat{\mathbf{K}}(\mu).$$

Hence, the solution $\hat{\mathbf{G}}$ defines a refined similarity between terms.

The kernel $\hat{\mathbf{K}}$ combines these indirect link kernels with an exponentially decaying weight. This and the diffusion kernels suggest an alternative weighting scheme that shows faster decay. Given a kernel $\mathbf{K}$, consider the exponential diffusion kernel of Definition 9.33

$$\bar{\mathbf{K}}(\mu) = \mathbf{K}\sum_{t=1}^{\infty}\frac{\mu^t\mathbf{K}^t}{t!} = \mathbf{K}\exp(\mu\mathbf{K}).$$

One advantage of this definition is that the series is convergent for all values of μ, so that no restrictions need to be placed on this parameter. The next proposition gives the semantic proximity matrix corresponding to $\bar{\mathbf{K}}(\mu)$.

Proposition 10.11 *Let* $\bar{\mathbf{K}}(\mu) = \mathbf{K}\exp(\mu\mathbf{K})$. *Then* $\bar{\mathbf{K}}(\mu)$ *corresponds to a semantic proximity matrix*

$$\exp\left(\frac{\mu}{2}\mathbf{G}\right).$$

Again the proof of the proposition can be found in Appendix A.3.

Remark 10.12 [Applying the diffusion kernels to unseen data] Note that for the case of text the application of the derived kernel is not restricted to the examples used in the kernel matrix. As with kernel PCA we can project new examples into the coordinate system determined by the principal axes and reweight them with the appropriate function of the corresponding eigenvalue. Hence, we can compute the diffusion construction on a modestly sized training set, but apply it to previously unseen data. ■

10.3 Summary

- The VSM developed in the IR community provides a rich source of kernels for the analysis of text documents.
- The simplest example is the bag-of-words kernel that treats all terms in the corpus as independent features.
- Refinements of the VSM are characterised by the use of different semantic matrices $\mathbf{S}$.

  Term weighting schemes lead to diagonal semantic matrices

  Off-diagonal entries can be set using external sources such as semantic nets.

  Alternatively we can learn semantic relations through co-occurrence analysis of the training corpus.

The GVSM gives the simplest approach, with latent semantic kernels and diffusion kernels making more refined use of the co-occurrence information.

Multilingual corpora can further refine the choice of semantic concepts as well as enabling cross-lingual information analysis.

10.4 Further reading and advanced topics

Kernel methods provide a natural framework for pattern analysis in text. This is not just through their interpretation of duality, but also because the function of a kernel in deciding if two documents, sentences or words have a similar meaning is much easier than actually analysing that meaning. This similarity measure can often be obtained by borrowing methods from different disciplines, as is evident from the diverse literature on which this chapter has been based.

The conventional VSM was introduced in 1975 by Salton *et al.* [112], and became the standard representation of documents in IR. Its use as a kernel, proposed by Joachims [66], is equivalent to setting the matrix $\mathbf{P} = \mathbf{I}$. A number of variations on that theme have been suggested, for example the GVSM [164], LSI [37] and many others, all of which have an obvious kernel counterpart.

Following this first embedding, a number of other, possibly nonlinear, mappings can be performed. Both Joachims, and Dumais *et al.* [41] used polynomial and Gaussian kernels to further refine the vector space representation, so further mapping the data into a richer space, for classification tasks using support vector machines.

The European project KerMIT has developed a number of new and sophisticated representations, kernels and algorithms, and their website contains a number of publications concerned with kernels for text, see www.euro-kermit.org. Also the book by Thorsten Joachims [67] is entirely devoted to text classification using kernel methods.

The survey paper [64] contains a framework using as a unifying concept the proximity matrix. The paper [126] discusses the semantic smoothing as performed by means of semantic networks and a particular choice of proximity matrix. The paper on latent semantic kernels [35, 31] discusses the use of those ideas within kernel methods, and the paper by [84] compares a number of choices of term weighting schemes, and hence indirectly of embedding maps.

The use of kCCA for cross language analysis is presented in [150] and the use of diffusion kernels in [69] (more background information about kernel

CCA can be found in Section 6.9 and diffusion kernels in Section 9.10). Other methods for cross-language analysis include adaptations of LSI described in [90]. Combinations of text and hyperlink information, in hypertext kernels, are described in [67].

Of course other kernels for text can be devised which do not use the VSM, as those based on string matching [91], and and probabilistic modeling, as in [57], but are better addressed within the context of Chapters 11 and 12. See Section 11.9 and Section 12.4 for more references.

For constantly updated pointers to online literature and free software see the book's companion website: www.kernel-methods.net