Kernels for Text

Topics

- BOW-based representation interpreted as kernels;
- Generalized vector space model;
- Semantic kernels.

Representing text

- Bag-of-words or Vector Space Model (VSM)

dictionary

t1	*t2*	….	*tN*

$$\phi : d \rightarrow \phi(d) = \left(tf\left(t_1,d\right), tf\left(t_2,d\right), \cdots, tf\left(t_N,d\right) \right) \in \Re^N$$

- A bag-of-words is a vector in a space in which each dimension is associated with one term from the dictionary

Document-term matrix

$$D = \begin{pmatrix} tf\left(t_1,d_1\right) & \cdots & f\left(t_N,d_1\right) \\ \vdots & \ddots & \vdots \\ tf\left(t_1,d_l\right) & \cdots & tf\left(t_N,d_l\right) \end{pmatrix}$$

- D' $\left[N \times l\right]$ is the *term-document* matrix
- $D'D$ $\left[N \times N\right]$ is the *term-by-term* matrix
- DD' $\left[l \times l\right]$ is the *document-by-document* matrix

Semantic issues

- 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.

Semantic issues

- <u>Synonymous words</u>: different words that carry the same meaning;
- The VSM assigns distinct components to synonymous words;
- Extra processing is necessary to embed the semantic relatedness of such words in the representation.

Semantic issues

- <u>Homonyms</u>: single word with two distinct meanings depending on context (e.g., bank, book);
- The VSM throws away the contextual information to disambiguate the meaning;
- Nevertheless, some context can still be derived from the statistics of the words in the document.

Improving the Embedding: <u>Weighting of Terms</u>

- Apply different weights to each coordinate, i.e., assign different weights to the terms;
- In its simplest form: *binary weights*
- A weight value of *0* is assigned to uninformative terms such as *and*, *of* , *the*, *a*, etc.
- Effectively removes *stop words,* considered uninformative for the task at hand;
- More general weighting schemes are also used.

Improving the Embedding: Normalization

- The longer a document the more words it contains - thus, the greater the norm of its associated vector;
- If the length of the document is not relevant for the task at hand, e.g. categorization by topic, we should remove its effect from the embedding vectors;

[Normalization]

- Let $\phi(\boldsymbol{x}),\phi(\boldsymbol{y})$ be our representation of documents $\boldsymbol{x},\boldsymbol{y}$
- Note: we can explicitly construct the mapping ϕ by capturing important domain knowledge, e.g.:

  $$\phi : d \rightarrow \phi(d) = \left(tf\left(t_1,d\right),tf\left(t_2,d\right),\cdots,tf\left(t_N,d\right)\right) \in \Re^N$$

  or define it implicitly through a standard kernel function k;
- In both cases: $k(\boldsymbol{x},\boldsymbol{y}) = \langle \phi(\boldsymbol{x}),\phi(\boldsymbol{y}) \rangle$

[Normalization]

- To remove the length of the documents from the embedding vectors:

$$\phi(\boldsymbol{x}) \to \hat{\phi}(\boldsymbol{x}) = \frac{\phi(\boldsymbol{x})}{\|\phi(\boldsymbol{x})\|}, \quad \phi(\boldsymbol{y}) \to \hat{\phi}(\boldsymbol{y}) = \frac{\phi(\boldsymbol{y})}{\|\phi(\boldsymbol{y})\|}$$

- This also defines a new kernel function:

$$\hat{k}(\boldsymbol{x},\boldsymbol{y}) = \left\langle \hat{\phi}(\boldsymbol{x}),\hat{\phi}(\boldsymbol{y}) \right\rangle = \left\langle \frac{\phi(\boldsymbol{x})}{\|\phi(\boldsymbol{x})\|},\frac{\phi(\boldsymbol{y})}{\|\phi(\boldsymbol{y})\|} \right\rangle$$

[Normalization]

$$\hat{k}(\boldsymbol{x},\boldsymbol{y}) = \left\langle \hat{\phi}(\boldsymbol{x}),\hat{\phi}(\boldsymbol{y}) \right\rangle = \left\langle \frac{\phi(\boldsymbol{x})}{\|\phi(\boldsymbol{x})\|},\frac{\phi(\boldsymbol{y})}{\|\phi(\boldsymbol{y})\|} \right\rangle$$

- [Norm of feature vectors]:

$$\|\phi(\boldsymbol{x})\|_2 = \sqrt{\|\phi(\boldsymbol{x})\|^2} = \sqrt{\left\langle \phi(\boldsymbol{x}),\phi(\boldsymbol{x}) \right\rangle} = \sqrt{k(\boldsymbol{x},\boldsymbol{x})}$$

$$\hat{k}(\boldsymbol{x},\boldsymbol{y}) = \left\langle \frac{\phi(\boldsymbol{x})}{\sqrt{k(\boldsymbol{x},\boldsymbol{x})}},\frac{\phi(\boldsymbol{y})}{\sqrt{k(\boldsymbol{y},\boldsymbol{y})}} \right\rangle = \frac{\left\langle \phi(\boldsymbol{x}),\phi(\boldsymbol{y}) \right\rangle}{\sqrt{k(\boldsymbol{x},\boldsymbol{x})}\sqrt{k(\boldsymbol{y},\boldsymbol{y})}}$$

$$= \frac{k(\boldsymbol{x},\boldsymbol{y})}{\sqrt{k(\boldsymbol{x},\boldsymbol{x})}\sqrt{k(\boldsymbol{y},\boldsymbol{y})}}$$

[Normalization]

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

- Normalization is implemented as the first transformation or as the final embedding;
- We can assume that when required, normalization is added as a final stage.

Important observations

- In general, working in a kernel-defined feature space means that we are not able to explicitly represent points;
- The image of an input point $\boldsymbol{x}$ is $\phi(\boldsymbol{x})$ but we do not have access to the components of this vector;
- We only have access to the evaluation of inner products between $\phi(\boldsymbol{x})$ and the images of other points;
- Despite this limitation, there is a surprising amount of useful information than can be derived by such inner products…

Linear combinations in feature space

$$\left\| \sum_{i=1}^{l} \alpha_i \phi(\boldsymbol{x}_i) \right\|^2 = \left\langle \sum_{i=1}^{l} \alpha_i \phi(\boldsymbol{x}_i), \sum_{j=1}^{l} \alpha_j \phi(\boldsymbol{x}_j) \right\rangle$$

$$= \sum_{i=1}^{l} \alpha_i \sum_{j=1}^{l} \alpha_j \left\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \right\rangle$$

$$= \sum_{i,j=1}^{l} \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Distance between feature vectors

$$\left\| \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) \right\|^2 = \left\langle \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}), \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) \right\rangle$$

$$= \left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}) \right\rangle - 2\left\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \right\rangle + \left\langle \phi(\boldsymbol{y}), \phi(\boldsymbol{y}) \right\rangle$$

$$= k(\boldsymbol{x}, \boldsymbol{x}) - 2k(\boldsymbol{x}, \boldsymbol{y}) + k(\boldsymbol{y}, \boldsymbol{y})$$

Successive embeddings

- The operations, for example term weighting and normalization, can be performed in sequence;
- This creates a series of successive embeddings: each one adds some refinement to the semantic of the representation;
- The composition of the successive embeddings generates a single map that incorporates different aspects of domain knowledge into the representation.

Vector space kernels

- Given a document, we know how to represent it as a vector:

$$\phi : d \rightarrow \phi(d) = \left(tf(t_1,d), tf(t_2,d), \cdots, tf(t_N,d)\right) \in \Re^N$$

- This preliminary embedding can then be refined by successive operations.
- Given a document-by-term matrix D, we can create the document-by-document matrix:

$$K = DD'$$

Vector space kernels

- Note:

$$K_{ij} = \left(DD'\right)_{ij} = \sum_{k=1}^{N} tf\left(t_k,d_i\right) tf\left(t_k,d_j\right)$$
$$= \left\langle \phi\left(d_i\right), \phi\left(d_j\right) \right\rangle = k\left(d_i,d_j\right)$$

- K is called the ***kernel matrix*** or ***Gram matrix***
- $k\left(d_i,d_j\right)$ is called the ***vector space kernel***

Vector space kernels

- Standard notation to display kernel matrices:

K	1	2	$\cdots$	l
1	$k(\boldsymbol{x}_1,\boldsymbol{x}_1)$	$k(\boldsymbol{x}_1,\boldsymbol{x}_2)$	$\cdots$	$k(\boldsymbol{x}_1,\boldsymbol{x}_l)$
2	$k(\boldsymbol{x}_2,\boldsymbol{x}_1)$	$k(\boldsymbol{x}_2,\boldsymbol{x}_2)$	$\cdots$	$k(\boldsymbol{x}_2,\boldsymbol{x}_l)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
l	$k(\boldsymbol{x}_l,\boldsymbol{x}_1)$	$k(\boldsymbol{x}_l,\boldsymbol{x}_2)$	$\cdots$	$k(\boldsymbol{x}_l,\boldsymbol{x}_l)$

- It contains all the information needed to compute pairwise distances within the data set;
- The only information received by an algorithm about the training set comes from the kernel matrix, and the associated labeling information.

Nonlinear embeddings

- We focus on linear transformations of the basic VSM by leveraging the power of capturing important domain knowledge;
- It is also possible to consider nonlinear embeddings using standard kernel constructions;
- For example, a polynomial kernel over the normalized bag-of-words representation:

$$\bar{k}(d_1,d_2) = \left(k(d_1,d_2)+1\right)^d = \left(\left\langle \phi(d_1),\phi(d_2)\right\rangle + 1\right)^d$$

Designing Semantic Kernels

- Objective: Extend the VSM representation to capture the semantic content of the words.
- We consider transformations of the document vectors $\phi(d)$:

$$\tilde{\phi}(d) = \phi(d)S$$

where S is a matrix that could be diagonal, square, or in general any $N \times k$ matrix.

Designing Semantic Kernels

- Using the transformation $\tilde{\phi}(d) = \phi(d)S$ the corresponding kernel takes the form:

$$\tilde{k}(d_1,d_2) = \left\langle \tilde{\phi}(d_1),\tilde{\phi}(d_2) \right\rangle = \left\langle \phi(d_1)S,\phi(d_2)S \right\rangle$$
$$= \left(\phi(d_1)S\right)\left(\phi(d_2)S\right)' = \phi(d_1)SS'\phi(d_2)' = \tilde{\phi}(d_1)\tilde{\phi}(d_2)'$$

- That is: the kernel follows directly from the explicit construction of a feature vector
- We refer to S as the ***semantic matrix***.

Designing Semantic Kernels

- Different choices of the matrix S lead to different variants of the VSM;
- We can generate S as a composition of several stages;
- We might define:

$$S = RP$$

- R is diagonal matrix giving the term weightings or term relevance;
- P is a proximity matrix defining the semantic relationships between the terms in the corpus of documents.

Term weighting: construction of R

- Not all words have the same importance in determining the topic of a document;
- Unsupervised measure: The frequency of a word across the documents in a corpus can be used to quantify the amount of information carried out by a word;
- Supervised measure: importance of a word with respect to a given topic, i.e., mutual information;

Term weighting: construction of R

- Inverse document frequency (idf): weights terms as a function of their inverse document frequency
  - l documents;
  - $df(t)$ = the number of documents containing the term t;
  - The usual measure of inverse document frequency for a term t is:

$$w(t) = \ln\left(\frac{l}{df(t)}\right)$$

Term weighting: resulting kernel

- Given a term weighting $w(t)$ (whether obtained via *idf* or some alternative scheme), we can define a new VSM;
- We can choose the matrix R to be diagonal with entries

$$R_{tt} = w(t)$$

- Thus, the associated kernel computes the inner product between documents in the new VSM representation $\phi(d)R$:

$$\tilde{k}(d_1,d_2) = \phi(d_1)RR'\phi(d_2)' = \sum_{t=1}^{N} w(t)^2\, tf(t,d_1)\, tf(t,d_2)$$

Term proximity matrix

- The previous *tf-idf* representation down-weights irrelevant terms and highlights discriminative ones;
- But it's not capable of recognizing when two terms are semantically related;
- Thus, cannot 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 to achieve this connection is through the introduction of semantic similarities between terms.

Term proximity matrix

- The embedding of semantic similarity within the VSM can be achieved through the matrix P
- A proximity matrix P should have positive off-diagonal terms $P_{ij} > 0$ when the term i is semantically related to term j
- Given such a matrix, a document is represented as a new *less sparse* vector

$$\phi(d)P$$

Term proximity matrix

$$\phi(d)P = \underbrace{(d_1, d_{2,\dots,} d_N)}_{\phi(d)} \underbrace{\begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{pmatrix}}_{P}$$

- The new vector has non-zero entries for all terms that are semantically similar to those present in the document d

Term proximity matrix

$$\phi(d)P = \underbrace{(d_1, d_{2,\ldots,} d_N)}_{\phi(d)} \underbrace{\begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{pmatrix}}_{P}$$

- *This is similar to a 'document expansion', where the document is expanded to include not only the actual terms that appear in the document, but also those that are semantically related.*

Term proximity matrix: resulting kernel

- Given a proximity matrix P, the corresponding vector space kernel is:

$$\tilde{k}(d_1,d_2) = \phi(d_1)PP'\phi(d_2)'$$

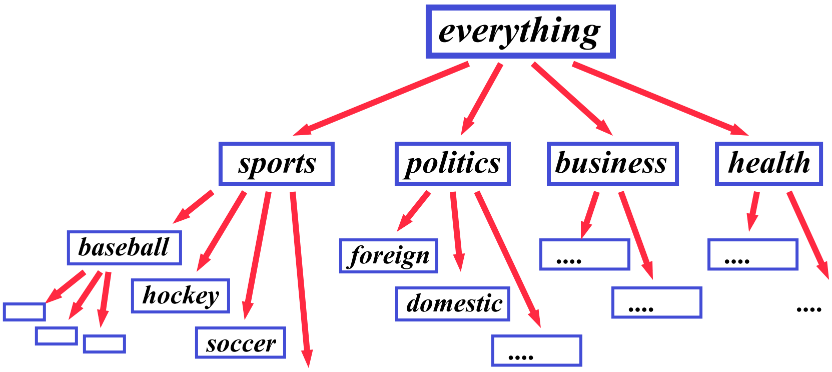
- Alternatively we can view $PP' = Q$ $\quad Q_{ij} = \sum_{k=1}^{N} s_{ik}s_{jk}$
- Thus Q_{ij} encodes the amount of semantic relation between terms i and j

$$\tilde{k}(d_1,d_2) = \sum_{i,j} \phi(d_1)_i Q_{ij} \phi(d_2)'_j$$

Explicit construction of the proximity matrix

- Construct *P* by using an external source of domain knowledge
- A semantic network such *Wordnet* provides a way to obtain term-similarity information
- A semantic network encodes relationships between words in a hierarchical fashion, where the more general terms are placed higher in the tree structure

Explicit construction of the proximity matrix



- We can use the distance between two terms on the hierarchical tree provided by Wordnet to give an estimate of their semantic proximity

Term proximity matrix: resulting kernel

- We can embed this information in the matrix P by setting P_{ij} equal to the *inverse of the distance between terms* i *and* j *in the tree* (i.e., *inverse of the length of the shortest path connecting them*).

- The use of this semantic proximity gives rise to the vector space kernel:

$$\tilde{k}(d_1, d_2) = \phi(d_1) P P' \phi(d_2)'$$

Generalized Vector Space Model (GVSM)

- Construct P directly from the data;

- Main idea: two terms are considered semantically related if they frequently co-occur in the same documents;

- Thus, two documents can be seen as similar even they do not share any terms, but the terms they contain co-occur in other documents.

Generalized Vector Space Model (GVSM)

- In *GVSM* a document is represented by a vector of its similarities with the different documents in the corpus:

$$\tilde{\phi}(d) = \phi(d)D'$$

  where D is the document-term matrix.

- This is equivalent to setting $P = D'$

- Why such document representation captures semantic similarities?

Generalized Vector Space Model (GVSM)

- Lets compute the corresponding kernel:

$$\tilde{k}(d_1,d_2) = \phi(d_1)D'D\phi(d_2)'$$

  where $(D'D)_{ij} = \sum_{d} tf(i,d)tf(j,d)$

- Thus $(D'D)_{ij}$ is nonzero if and only if there is at least one document in the corpus in which the terms i and j co-occur.
- The strength of the association between two terms i and j depends on how often (in how many documents) they co-occur in the given corpus.

Latent semantic kernels

- Though appealing, the GVSM is too naïve it its use of the co-occurrence information.
- Latent semantic kernels provide a more subtle use of this information to create refined semantics.
- Conceptually, latent semantic indexing (LSI) follows the same approach as GVSM: it extracts semantic information from the co-occurrences of terms.
- The technique used to extract the information is different though: LSI makes use of SVD.
- We'll see that this technique amounts to a special choice of the matrix P.

Latent semantic kernels

- Recall that the SVD of the term-by-document matrix D' is:

$$D' = U \Sigma V'$$

- Σ is a diagonal matrix, the columns of U are the eigenvectors of $D'D$
- LSI projects the documents into the space spanned by the first k columns of U, and uses these new k-dimensional vectors for subsequent processing:

$$d \to \phi(d) U_k$$

where U_k is the matrix containing the first k columns of U.

Latent semantic kernels

- Recall that the eigenvectors define the subspace that minimizes the sum of the squared differences between the points and their projections;
- So, the eigenvectors define the subspace with minimal sum of squared residuals;
- 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 well as possible using only k such concepts.

Latent semantic kernels

- 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 new dimensions that can reconstruct the whole feature vector.
- Hence: SVD exploits co-occurrence information to maximize the amount of information extracted by a given number of dimensions.

Latent semantic kernels

- The resulting latent semantic kernel is:

$$\tilde{k}(d_1,d_2) = \phi(d_1)U_k U_k^{'}\phi(d_2)^{'}$$

  which shows that $P = U_k$

- $P = U_k$ introduces a dimensionality reduction through the restriction to k eigenvectors;
- As k increases, we return to the treatment of all terms being semantically distinct. Hence, the value of k controls the amount of *semantic smoothing* that is introduced into the representation.