Text classification using Semantic Information and Graph Kernels

Miguel Gaspar, Teresa Gonçalves, and Paulo Quaresma

Universidade de Évora, Portugal
miguel.ferreira.gaspar@gmail.com, tcg@uevora.pt, pq@uevora.pt

Abstract. The most common approach to the text classification problem is to use a bag-of-words representation of documents to find the classification target function. Linguistic structures such as morphology, syntax and semantic are completely neglected in the learning process. This paper uses another document representation that, while including its context independent sentence meaning, is able to be used by a structured kernel function, namely the direct product kernel. The semantic information is obtained using the Discourse Representation Theory and similarity function between documents represented by graphs is defined.

1 Introduction

The problem of automatic classification of text documents have great practical importance, given the huge volume of texts available in the *World Wide Web*, catalogues, news, email, corporate databases, medical records of patients, digital libraries and others. The learning algorithms can be trained to classify documents, given a sufficient set of training examples previously classified. Some algorithms have been used to automatically classify new articles [21], web pages [10], automatically learn the reading interests of users [27], and automatically sort email [28][29].

A common approach to text classification is to use a bag-of-words representation, where each document is represented by the words it contains and syntax, punctuation and semantic are ignored.

This paper presents another approach to this problem: using the Discourse Representation Theory [24], one obtains a semantic representation of each document that is then transformed into a graph [16]. This allows the use of kernel functions [14][25] for structured data. Using Support Vector Machines (SVM) [30] [31] with these kernels appears to be a good approach since this algorithm is known to produce good results on text classification tasks.

The paper is organised as follows: Section 2 introduces the used concepts and tools for Discourse Representation Theory and Kernel Methods; Section 3 proposes a representation of documents that includes their semantic and is fitted for graph kernel functions and presents a measure of similarity between documents; Section 4 describes the experiments and presents the results. Conclusions and future work are pointed out on Section 5.

2 Concepts and Tools

This section introduces basic graph theory concepts, the Discourse Representation Theory and Kernel Methods for structured data and presents the used tools.

2.1 Basic graph concepts

A graph G is described usually by a finite set of vertex V, a finite set of edges E and a function Ψ, $G = (V, E, \psi)$. In labelled graphs there is also a label set $\mathcal{L}$ and a labelling function Λ, that attaches a label to each edge and vertex. In directed graphs, the function Ψ maps each edge to the pair formed by its initial and final vertex, $\Psi : \mathcal{E} \rightarrow (u, v) \in \mathcal{V} \times \mathcal{V}$ and the edges for which $\Psi(e) = (v, v)$ are called ties.

A path p is a sequence of vertices $v_i \in \mathcal{V}$ and edges $e_i \in E$ with $\Psi(e_i) = (v_i, v_{i+1})$ and $p = v_1, e_1, v_2, e_2, ..., e_n, v_{n+1}$. The path length is equal to the number of edges in the sequence (n in the previous case).

The adjacency matrix A of a graph is the matrix whose elements $[A]_{ij}$ correspond to the number of edges from v_i to v_j. If we consider the n^{th} power of the adjacency matrix ($n \in \mathbb{N}, n \geq 0$), the interpretation is similar: each array element $[A]_{ij}$ is the number of paths of length n from vertex v_i to vertex v_j.

2.2 Discourse Representation Theory

Discourse Representation Theory, or DRT, is a specific name for the work of Hans Kamp as a theoretical framework for dealing with issues in the semantics and pragmatics of anaphora and tense [23]. It is one of a number of theories of dynamic semantics, which have come upon the scene in the course of the past twenty years. The central concern of these theories is to account for the context dependence of meaning. The distinctive features of DRT, are that it is a mentalist and representationalist theory of interpretation, and that it is a theory of the interpretation not only of individual sentences but of discourse, as well.

The main component of DRT is the Discourse Representation Structure (DRS) [24]. A DRS consists of two parts: the set of discourse referents and the set of conditions. The discourse referents are variables representing all of the entities in the DRS. The conditions are the logical statements about these entities. Here is an example of a DRS representation of the sentence *"He throws the ball."*.

```
[   x1, x2, x3:
    male(x1), ball(x2), throw(x3),
    event(x3), agent(x3, x1), patient(x3, x2)
]
```

Figure 1 presents the usual graphical representation for the same sentence. It is possible to observe the referents `X1`, `X2` e `X3` and the conditions `male(X1)`, `ball(X2)`, `throw(X3)`, `event(X3)`, `agent(X3,X1)` and `patient(X3,X2)`. These restrictions indicate that `X1` is male, `X2` is a ball and `X3` is an event whose agent is `X1` and patient is `X2`.

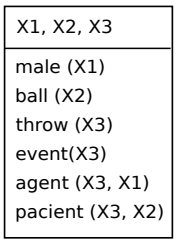

Fig. 1. Graphical DRS representation for sentence *"He throws the ball."*

2.3 Kernel Methods

Kernel Methods are a class of algorithms to analyse patterns, whose most known example is the Support Vector Machine (SVM). Kernel Methods constitute a popular field of research in the area of machine learning.

Kernel-based machine learning algorithms abandon the explicit representation of data items in the vector space in which the patterns are to be detected. Instead, they implicitly mimic the geometry of the feature space by means of the kernel function – a similarity function that maintains a geometric interpretation as the inner product of two vectors.

Formally, a kernel is a function $k : X \times X \rightarrow R$, such that for all $x, z \in X$ $k(x, z) = \langle \Phi(x), \Phi(z) \rangle$ where Φ is a mapping from X to an inner product feature space.

Support Vector Machines. Support Vector Machines (SVM) [6][30][31] can be seen as the first practical application of the principle of structural risk minimisation. In SVM [22]:

- a predictor variable is called an attribute;
- a transformed attribute used to define the hyperplane is called a feature;
- the task of choosing the most suitable representation is known as feature selection;
- a set of features that describes one case is called a vector.

The goal of SVM modelling is to find the optimal hyperplane that separates clusters of vectors in such a way that cases of one value of target variable are

on one side of the hyperplane and cases of the other value are on the other side. The nearest vectors to the hyperplane are called support vectors.

More formally [7], a support vector machine constructs a hyperplane or a set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalisation error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that in that space the cases are not linearly separable. For this reason the original finite dimensional space is mapped into a much higher dimensional space, presumably making the separation easier in that space. SVM schemes use a mapping into a larger space so that inner products may be computed easily in terms of the variables in the original space, making the computational load reasonable. The inner products in the larger space are defined in terms of a kernel function selected to suit the problem.

Graph Kernels. The application of kernel functions to graph structured data was introduced independently by Gartner [14] and Kashima [25]. Conceptually they are based on measures over graphs' walks with common labels: the first counts walks with initial and final common labels and the last calculates probabilities of equal label sequences on random walks.

Gartner *et al.* [15] prove that the computation of a kernel function able to completely recognise graph structure is NP-hard and introduce a walk based kernel function that computes in polynomial time including both previous kernels as special cases. This kernel, known as *product graph kernel* is based on the concept of the direct product graph, counting walks on that graph. Product graphs are a discrete mathematics tool [20] and the direct product graph is between the four most important ones.

Formally, the product graph kernel between two graphs G_1 e G_2 is defined as:

$$k(G_1, G_2) = \sum_{i,j=0}^{|V_x|} \left[\sum_{n=0}^{\infty} \lambda_n A_x^n \right]_{ij} \tag{1}$$

where A_x is product adjacency matrix , $E_x = E(G_1 \times G_2)$; V_x is the set of vertices of the direct product, $V_x = V(G_1 \times G_2)$; λ_i is sequence of weights $\lambda_0, \lambda_1, \ldots$ such that $\lambda_i \in \mathbb{R}$ and $\lambda_i \geq 0$ for all $i \in \mathbb{N}$. The adjacency matrix E_x^n is a square matrix where the element (i, j) counts the paths of size n between node i and node j of the direct product graph.

This kernel [8][26] has a major importance because it enables to determine the degree of similarity between two graphs by returning a new graph that is the direct product between the original ones. The adjacency matrix of the product graph counts the number of walks between any two vertex.

Direct Product Graph. Given two labelled graphs $G_1 = (V_1, E_1, \psi_1)$ and $G_2 = (V_2, E_2, \psi_2)$, the direct product graph is denoted by $G_1 \times G_2$. The vertex set of the direct product is defined as:

$$V(G_1 \times G_2) = (v_1, v_2) \in V_1 \times V_2 : equal(\Lambda(v_1), \Lambda(v_2))$$

and edge set is defined as:

$$\begin{aligned} E(G_1 \times G_2) = \{\ (e_1, e_2) \in E_1 \times E_2 : \exists (u_1, u_2), (v_1, v_2) \in V(G_1 \times G_2) \\ \wedge\ \Psi_1(e_1) = (u_1, v_1) \wedge \Psi_2(e_2) = (u_2, v_2) \\ \wedge\ equal(\Lambda(e_1), \Lambda(e_2))\ \} \end{aligned}$$

The vertex and edges labels in the product graph $G_1 \times G_2$ correspond to the factors labels. More formally, given an edge $(e_1, e_2) \in E(G_1 \times G_2)$ with $\Psi_1(e_1) = (u_1, v_1)$ and $\Psi_2(e_2) = (u_2, v_2)$, the value of $\Psi_{G_1 \times G_2}$ is:

$$\Psi_{G_1 \times G_2}(e_1, e_2) = ((u_1, u_2), (v_1, v_2))$$

2.4 Tools

Here, we briefly present the graph modelling language, the linguistic information and machine learning tools used in this work.

Graph Modelling Language. Graph Modelling Language (GML) [19] was used to represent graphs. GML's key features include a simple syntax and portability, extensibility and flexibility. A GML file consists of a hierarchical key-value lists. Also, in GML graphs can be annotated with arbitrary data structures.

Natural Language Toolkit. This software was originally created in 2001 as part of a computational linguistics course in the Department of Computer and Information Science at the University of Pennsylvania. Since then it has been developed and expanded with the help of dozens of contributors. It has now been adopted in courses in dozens of universities, and serves as the basis of many research projects.

The Natural Language Toolkit (NLTK) [1][2][13] is a suite of program modules, data sets, tutorials and exercises, covering symbolic and statistical natural language processing. It is written in Python and distributed under the GPL open source license. NLTK implements among other, prototypes for morphosyntactic tagging, text classification, information extraction, parsing, semantics and anaphora resolution.

The absence of an English grammar for generating DRS from texts, implied the need for other tools. Nevertheless, its use allows to apply anaphora resolution to the DRS obtained.

C&C Tools (Candc and Boxer). The C&C tools system is built around a wide-coverage Combinatory Categorial Grammar (CCG) parser. The parser not only recovers the local dependencies output by tree-bank parsers, but also the long-range dependencies inherent in constructions such as extraction and coordination. CCG is a lexicalised grammar formalism, so each word in a sentence is assigned an elementary syntactic structure. Statistical tagging techniques makes possible to assign lexical categories with high accuracy and low ambiguity.

C&C also contains a number of Maximum Entropy taggers, including the CCG supertagger, a POS tagger [12], a chunker, and a named entity recogniser [11]. Finally, the various components, including the morphological analyser are combined into a single application. The output from this application – a CCG derivation, POS tags, lemmas, and named entity tags – is used by the Boxer module [3] to produce a semantic interpretation in the form of Discourse Representation Structures.

Some well known work with this tool have been made, like automatically finding answers for a collection of questions [4] or recognising textual entailment [5].

LibSMV. This tool was used to obtain the text classifier and is an implementation of Support Vector Machines for classification, regression and estimation of distribution [9]. Is was chosen among others implementations since it allows the use of pre-calculated kernel matrixes from personalised kernel functions.

3 Similarity between documents

Most common approaches use a bag-of-words representation, ignoring text syntax and semantics. So the use of semantic information presents a novel approach to text classification [17].

Discourse Representation Structures allow to represent semantic information and their transformation into graphs allows the application of the Support Vector Machine algorithm with the direct product kernel to text classification.

This section introduces the process for obtaining document's semantic information as a graph and the definition of the similarity measure between these graphs.

3.1 From a text to a DRS

Given a text document, the corresponding DRS is obtained by using NLTK and C&C tools. The document is presented to NLTK that, in turn, presents each sentence to Boxer and requests the corresponding DRS. Then, Boxer calls CANDC obtaining the CCG derivations that allows Boxer to generate the semantic representation as a DRS. This structure is returned to NLTK, allowing it to subsequent apply anaphora resolution. A scheme of operation can be seen in Figure 2.

The DRS generated by this system when presented with the sentence *"He throws the ball."* can be graphically viewed in Figure 1.

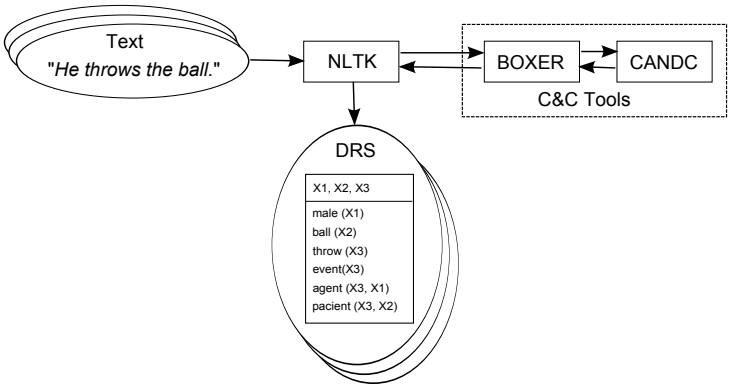

Fig. 2. Functional diagram for obtaining a DRS from a text.

3.2 From a DRS to a graph

As presented in Section 2.1, a graph is made of vertices and edges. If one considers that a vertex is an element, an edge can be seen as a relationship between the connected elements [18].

On the other way and as presented in Section 2.2, a DRS is composed of a set of discourse referents and a set of conditions: referents identify entities and conditions represent the constraints associated with those entities.

With these considerations in mind, one can consider that there is some similarity in them: both are made of elements and relations between them. This leads us to our proposal of representing a DRS as a graph: referents are vertices and conditions are edges. Figure 3 displays the graph representation for the sentence *"He throws the ball."*. It corresponds to the DRS presented in Figure 1.

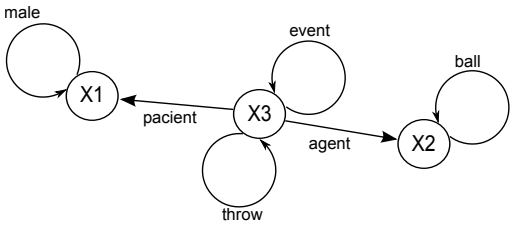

Fig. 3. Graph representation for the sentence *"He throws the ball."*

3.3 Similarity between graphs

The construction of the direct product graph (see Section 2.3) depends on an equality function between vertices and edges of graph factors. This function

measures the similarity between the two factors by deciding which vertices and edges will compose the product. When applied to text documents, this function will indirectly measure the semantic similarity between two documents.

The analysis of the graphs generated using the approach described in Section 3.2 and given that DRSs are obtained independently for each text document, vertices' equality can not use vertices labels, since different referents may have same label in different texts. This implies that vertices' equality must consider edges connections.

To obtain the product graph (that considers similarity between graph factors) there is a need to map the origin and destination vertices that have equal edges. Since most times, this mapping is not unique, we want to obtain the 'best' mapping. Our proposal for this mapping is done using a greedy search that observes the following rules:

1. identify the origin vertex with highest priority being processed:
   (a) the highest priority vertex is the one that has a larger number of edges with identical labels;
   (b) in case of a tie priority goes to the farthest path of identical labels edges;
2. ignore destination vertices where there is no equal edges on both graphs, from the vertex of origin;
3. continue the process sequentially until there is no more vertices to process.

Figure 4 presents two graphs $\mathcal{A}$ and $\mathcal{B}$. Figure 5 presents an step-by-step example of applying the direct product to graphs $\mathcal{A}$ and $\mathcal{B}$ from figure 4: since there is edge '`a`' between vertices `A` and `B` (graph $\mathcal{A}$) and also between vertices `Z` and `W` (graph $\mathcal{B}$) we match vertex `A` with vertex `Z` and vertex `B` with vertex `W`, obtaining the graph displayed in Step 1; Step 2 is obtained by matching vertices `F` and `M` through edge '`b`'; graphs displayed in Step 3 and Step 4 are obtained in the same way.

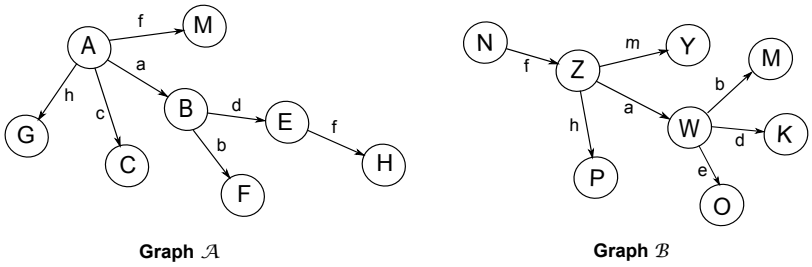

Fig. 4. The graphs $\mathcal{A}$ and $\mathcal{B}$ to obtain their direct product.

3.4 Classification process

For the classification, it is necessary to first calculate the direct product graph between all pairs of documents. Having these graph products it is then possible

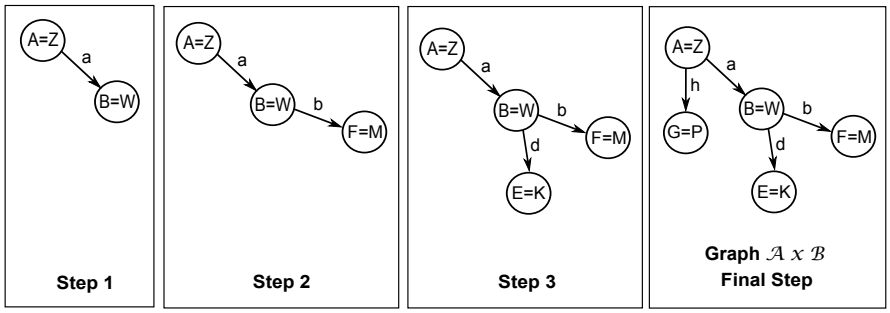

Fig. 5. Steps performed to obtain direct product from graphs $\mathcal{A}$ and $\mathcal{B}$.

to calculate the value of the kernel function (equation 1) that will give us the degree of semantic similarity between each two documents.

Having the value function for all pairs of documents, we have the kernel matrix that can be given to the Support Vector Machines algorithm to optimise the separating hyperplane between classes of documents. Figure 6 illustrates the process for obtaining the classifier: the matrix kernel of the graph product function is the LibSVM input that, in turn, generates a classification model.

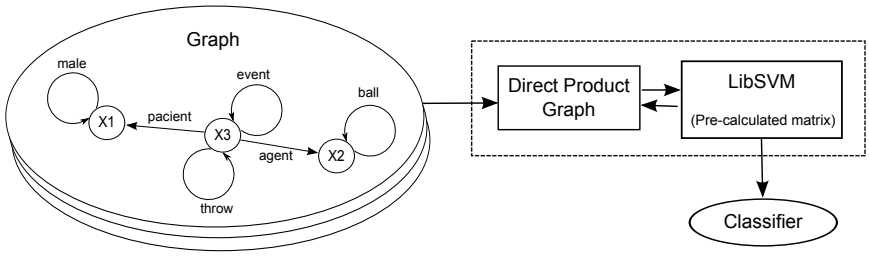

Fig. 6. Functional diagram for obtaining a classifier from text semantic information (represented as a graph).

4 Experiments

To assess the quality of the tools for obtaining the semantic information from documents and a valid use of this information in the construction of text classifiers, we performed a set of experiments. This section describes and evaluates them.

For the evaluation we used a subset of the most known corpus for text classification – `Reuters-21578`[1] dataset. The Reuters-21578 dataset is a multi-label

[1] `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

classification problem. It was compiled by David Lewis and originally collected by the Carnegie group, from the Reuters newswire in 1987. We randomly chose 50 documents for each of the five most common classes: `earn`, `acq`, `money-fx`, `grain` and `crude`.

Each document from each class was processed to obtain semantic representation. The average processing time for generating each DRS was 26.44 seconds with an average success rate (ratio between the number of DRSs obtained and the number of texts analysed) of 92.23%.

4.1 Classifier

The graphs generated from the DRSs were then used as the text semantic representation for classification. This subsection presents the preliminar results.

Since the process to generate a classifier model is very time consuming, we randomly selected 25 documents for each of the five classes: `acq`, `crude`, `interest`, `money` and `trade` (taking into attention that documents should belong to only one of those classes).

We performed 5 random runs for each pair of classes.We tried several values for SVM cost parameter (C) in range [0.1..2]. The best results were obtained for $C = 0.2$.

For the direct product kernel, we chose $\lambda_n = \gamma^{-n}$ (the parameter that weights the adjacency matrix E_x^n for paths of size n), with $\gamma = 2$, for distances of size less than or equal to 4. It corresponds to weights of 1, $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ for paths of length 1, 2, 3 and 4, respectively. For greater path we considered $\lambda_n = 0$.

Table 1 shows the mean accuracy (and standard deviation) for each binary problem.

	`crude`	`interest`	`money-fx`	`trade`
`acq`	51%± 1.6	54%± 3.9	51%± 1.5	52%± 1.5
`crude`		52% ± 1.3	50% ± 2.2	52% ± 2.4
`interest`			50% ± 2.2	52% ± 1.5
`money-fx`				52% ± 1.5

Table 1. Mean and standard deviation accuracy values using graphs to represent text semantic information.

5 Conclusions and future work

This paper presents a proposal for using documents semantic information for text classification. Although the set of conducted experiments was limited one can say that this work presents an interesting proposal since:

- the semantic information is represented in a way that enables the use of machine learning methods;

– the DRS referents mapping allows the use of this kernel function in a generic Support Vector Machines algorithm;
– the obtained classifier have accuracy values that are a proof of concept that is possible to classify documents based on his semantic representation;

However some improvements can be made:

– by analysing the results, a better mapping between referents should be found to improves the similarity measure between documents;
– by revising the weights assigned to paths of different lengths, an evaluation of the importance of longer paths could be studied (unlike the geometric series used that penalise paths with greater length).

As future work we intend to evaluate other mappings between referents. One such mapping could be to consider as vertex labels the label of its ties. When having more than one tie, statistical measures may be applied to choose the most profitable one. This mapping will allow the simple application of direct product graph, where vertices will be matched by their label.

Other line of work is to perform more tests both extending the number of documents and classes considered. The use of other datasets should also be considered.

References

1. Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*, volume 1. O'Reilly Media Inc, 2009.
2. Steven Bird, Ewan Klein, Edward Loper, and Jason Baldridge. Multidisciplinary instruction with the natural language toolkit. *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics, ACL*, 2008.
3. Johan Bos. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, STEP '08, pages 277–286, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.
4. Johan Bos, James R. Curran, and Edoardo Guzzetti. The pronto qa system at trec-2007: harvesting hyponyms, using nominalisation patterns, and computing answer cardinality. In E. M. Voorhees and L. P. Buckland, editors, *The Sixteenth Text RETrieval Conference, TREC 2007*, pages 726–732, Gaitersburg, MD, 2007.
5. Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 628–635, 2005.
6. Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM.
7. Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.
8. Francesco Camastra and Alfredo Petrosino. Kernel methods for graphs: A comprehensive approach. In *Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part II*, KES '08, pages 662–669, Berlin, Heidelberg, 2008. Springer-Verlag.

9. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.
10. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. *Proceedings of the Fifteenth National Conference on Artificial Intellligence (AAAI-98*, pages pp.509–516, 1998.
11. James R. Curran and Stephen Clar. Language independent ner using a maximum entropy tagger. *In Proceedings of CoNLL-03*, pages pp.164–167, 2003.
12. James R. Curran and Stephen Clark. Investigating gis and smoothing for maximum entropy taggers. *In Proceedings of the 10th Meeting of the EACL*, pages pp.91–98, 2003.
13. Dan Garrette and Ewan Klein. An extensible toolkit for computational semantics. *Proceedings of the Eighth International Conference on Computational Semantics*, 2009.
14. T. Gartner. Exponential and geometric kernels for graphs. *In NIPS-02, 16th Neural Information Processing Systems*, Workshop on Unreal Data: Principles of Modeling Nonvectorial Data, 2002.
15. T. Gartner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alter-natives. *Proc. Annual Conf. Computational Learning Theory*, pages pp.129–143, 2003.
16. T. Gonçalves. *Utilização de Informação Linguística na classificação de documentos em Língua Portuguesa*. Phd, Universidade de Évora, Évora, Portugal, 2007.
17. T. Gonçalves and P. Quaresma. Using graph-kernels to represent semantic information in text classification. In *MLDM-09 – International Conference on Machine Learning and Data Mining*, volume 5632 of *LNAI*, pages 632–646, Leipzig, DE, July 2009. Springer-Verlag.
18. Jonathan Gross and Jay Yellen. *Graph theory and its applications*. CRC Press, Inc., Boca Raton, FL, USA, 1999.
19. M. Himsolt. Gml: A portable graph file format. *Technical Report, Universität Passau*, 1997.
20. W. Imrich and S. Klavzar. *Product Graphs: Structure and Recognition*. John Wiley, 2000.
21. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *In Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pages pp.137–142, 1998.
22. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer Berlin / Heidelberg, 1998.
23. H. Kamp. A theory of truth and semantic representation. *J.A.G. Groenendijk*, 1981.
24. Han Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume Studies in Linguistics and Philosophy, Vol. 42. Springer, 1993.
25. H. Kashima and A. Inokuchi. Kernels for graph classification. *In ICDM-02, IEEE International Conference on Data Mining - Workshop on Active Mining*, 2002.
26. Nikhil S. Ketkar, Lawrence B. Holder, and Diane J. Cook. Faster computation of the direct product kernel for graph classification. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 267–274, 2009.
27. L. Lang. Newsweeder: Learning to filter netnews. *In Machine Learning: Proceedings of the Twelfth International Conference (ICML '95)*, pages pp.331–339, 1995.

28. D.D. Lewis and K.A. Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, pages pp.209–217, 1997.
29. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A baysian approach to filtering junk e-mail. *AAAI-98 Workshop on Learning for Text Categorization. Tech. rep. WS-98*, 1998.
30. Vladimir Vapnik. *The Nature of Statistical Learning Theory*. New York, 1995.
31. Vladimir Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, pages 988–999, 1999.