Automatic Annotation of a Dynamic Corpus by Label Propagation

Thomas Lansdall-Welfare, Ilias Flaounas, and Nello Cristianini

Abstract We are interested in the problem of automatically annotating a large, constantly expanding corpus, in the case where potentially neither the dataset nor the class of possible labels that can be used are static, and the annotation of the data needs to be efficient. This application is motivated by real-world scenarios of news content analysis and social-web content analysis. We investigate a method based on the creation of a graph, whose vertices are the documents and the edges represent some notion of semantic similarity. In this graph, label propagation algorithms can be efficiently used to apply labels to documents based on the annotation of their neighbours. This paper presents experimental results about both the efficient creation of the graph and the propagation of the labels. We compare the effectiveness of various approaches to graph construction by building graphs of 800,000 vertices based on the Reuters corpus, showing that relation-based classification is competitive with support vector machines, which can be considered as state of the art. We also show that the combination of our relation-based approach and support vector machines leads to an improvement over the methods individually.

Keywords Graph construction • Label propagation • Large scale • Text categorisation

1 Introduction

A standard approach to annotation of documents in a large corpus is to use content-based classifiers, e.g. Support Vector Machines (SVMs), specialised in the detection of specific topics [20]. In the case where the corpus grows over time, these classifiers

T. Lansdall-Welfare • I. Flaounas • N. Cristianini
Intelligent Systems Laboratory, University of Bristol, Bristol, UK
e-mail: Thomas.Lansdall-Welfare@bris.ac.uk; Ilias.Flaounas@bris.ac.uk; Nello.Cristianini@bris.ac.uk

are applied to all new documents. In the case where new classifications need to be added to the system, new classifiers need to be trained and added to the set. We are interested in the design of highly autonomous systems for the management of corpora, and as part of this effort we have developed the news monitoring infrastructure 'News Outlets Analysis and Monitoring system' (NOAM) [14]. As part of adding more autonomy and flexibility to that infrastructure, we have been investigating various ways to propagate annotation across documents in a corpus that does not involve training content-based classifiers. We are further interested in the situation where the annotation of a corpus improves with time, that is with receiving new labelled data. We want the accuracy of existing labels to improve with more data, where old errors in classification are possibly being amended, and if entirely new labels start being used in a data stream, the system will be able to accommodate them automatically and efficiently.

In this paper we explore the use of label propagation algorithms to label graph nodes, so that the knowledge of our system about the corpus is represented both in the topology of the graph and in the labels attached to its nodes. This also involves using scalable graph creation methods. The naïve approach to graph construction, comparing all documents to all documents or building a complete kernel matrix [28], will not work in large-scale systems due to high computational complexity. The cost of label propagation is also an important factor on the time needed to process incoming documents.

We present a method to propagate labels across documents by creating a sparse graph representation of the data, and then propagating labels along the edges of the graph. Much recent research has focused on methods for propagation of labels, taking for granted that the graph topology is given in advance [8, 18]. In reality, unless working with web pages, textual corpora rarely have a predefined graph structure. Graph construction alone has a worst case cost of $\mathcal{O}(N^2)$ when using a naïve method due to the calculation of the full $N \times N$ pairwise similarity matrix. Our proposed method can be performed efficiently by using an inverted index, and in this way the overall cost of the method has a time complexity of $\mathcal{O}(N \log N)$ in the number of documents N.

We test our approach by creating a graph of 800,000 vertices using the Reuters RCV1 corpus [22], and we compare the quality of the label annotations obtained by majority voting against those obtained by using SVMs. We chose to compare the graph-based methods to SVMs because they are considered the state of the art for text categorisation [27]. We show that our approach is competitive with SVMs, and that the combination of our relation-based approach with SVMs leads to an improvement in performance over either of the methods individually. It is also important to notice that our methods can be easily distributed to multiple machines.

1.1 Related Work

There is a growing interest in the problem of propagating labels in graph structures. Previous work by Angelova and Weikum [2] extensively studied the propagation of labels in web graphs including a metric distance between labels, and assigning weights to web links based upon content similarity in the webpage documents. Many alternative label propagation algorithms have also been proposed over the years, with the survey [31] giving an overview of several different approaches cast in a regularisation framework. A common drawback of these approaches is the prohibitively high cost associated with label propagation. A number of recent works on label propagation [8,9,18] concentrate on extracting a tree from the graph, using a very small number of the neighbours for each node. While many graph-based methods do not address the problem of the initial graph construction, assuming a fully connected graph is given, or simply choosing to work on data that inherently has a graph structure, there is a large number of papers dedicated to calculating the nearest neighbours of a data point. One such approximate method, NN-Descent [13], shows promising results in terms of accuracy and speed for constructing k-Nearest Neighbour graphs, based upon the principle that ‘a neighbour of a neighbour is also likely to be a neighbour’. The All-Pairs algorithm [5] tackles the problem of computing the pairwise similarity matrix often used as the input graph structure in an efficient and exact manner, showing speed improvements over another inverted-list-based approach, ProbeOpt-sort [26] and well-known signature-based methods such as Locality Sensitive Hashing (LSH) [15]. In this paper we take a broader overview, considering both the task of creating a graph from text documents, and then propagating labels for text categorisation simultaneously. We are interested in an approach that can be applied to textual streams, with the previously mentioned additional benefits offered by moving away from classical content-based classifiers. This paper is an extended version of the paper [21].

2 Graph Construction

Graph construction $\mathcal{X} \rightarrow \mathcal{G}$ deals with taking a corpus $\mathcal{X} = \{x_1, \ldots, x_n\}$, and creating a graph $\mathcal{G} = (V, E, W)$, where V is the set of vertices with document x_i being represented by the vertex v_i, E is the set of edges, and W is the edge weight matrix. There are several ways the construction can be adapted, namely the choice of distance metric and the method for maintaining sparsity.

The distance metric is used to determine the edge weight matrix W. The weight of an edge w_{ij} encodes the similarity between the two vertices v_i and v_j. The choice of metric used is mostly task dependent, relying on an appropriate selection being made based upon the type of data in $\mathcal{X}$. A common measure used for text, such as cosine similarity [24], may not be appropriate for other data types, such as when dealing with histogram data where the χ^2 distance is more meaningful [30].

Typically, a method for maintaining sparsity is required since it is not desirable to work with fully connected graphs for reasons of efficiency, and susceptibility to noise in the data [19]. This can be solved by working with sparse graphs, which are easier to process. Two popular methods for achieving sparsity include k-nearest neighbour (kNN) and ϵ-neighbourhood, both utilizing the local neighbourhood properties of each vertex in the graph [7, 19, 23]. Local neighbourhood methods are important for efficiency since a data point only relies on information about other points close by, with respect to the distance metric, to determine the neighbours of a vertex. This means that no global properties of the graph need to be calculated over the entire graph each time a new vertex is added, a consideration that has implications both for the scalability and, more generally, for the parallelisation.

The first step of creating the graph usually involves calculating the pairwise similarity score between all pairs of vertices in the graph using the appropriately chosen distance metric. Many studies assume that it is feasible to create a full $N \times N$ distance matrix [19] or that a graph is already given [8, 18]. This assumption can severely limit the size of data that is managable, limited by the $\mathcal{O}(N^2)$ time complexity for pairwise calculation. Construction of a full graph Laplacian kernel, as required by standard graph labelling methods [6, 17, 32] is already computationally challenging for graphs with 10,000 vertices [18]. Jebara et al. [19] introduce β-matching, an interesting method of graph sparsification where each vertex has a fixed degree β and show an improved performance over k-nearest neighbour, but at a cost to the complexity of the solution and the assumption that a fully connected graph is given.

We can overcome the issue of $\mathcal{O}(N^2)$ time complexity for computing the similarity matrix by using an alternative method, converting the corpus into an inverted index where each term has a pointer to the documents the term appears within. The advantage of this approach is that the corpus is mapped into a space based upon the number of terms, rather than the number of documents. This assumption relies on the size of the vocabulary $|t|$ being much smaller than the size of the corpus. According to Heaps' Law, the number of terms $|t|$ appearing in a corpus grows as $\mathcal{O}(N^\beta)$, where β is a constant between 0 and 1 dependent on the text [16]. Some experiments on English text have shown that in practice β is between 0.4 and 0.6 [3,4]. The inverted index can be built in $\mathcal{O}(NL_d)$ time where L_d is the average number of terms in a document, with a space complexity of $\mathcal{O}(NL_v)$ where L_v is the average number of unique terms per document [29].

Finding the neighbours of a document is also trivial because of the inverted index structure. A classical approach is to use the Term Frequency-Inverse Document Frequency (TF-IDF) weighting [24] to calculate the cosine similarity between two documents. This can be performed in $\mathcal{O}(L_d \log|t|)$ time for each document by performing L_d binary searches over the inverted index. Assuming β from Heaps' Law is the average value of 0.5, the time complexity for finding the neighbours of a document can be rewritten as $\mathcal{O}(\frac{L_d}{2}\log N)$. Therefore, there is a total time complexity $\mathcal{O}(N + \frac{NL_d}{2}\log N)$ for building the index and finding the neighbours of all vertices in the graph. This is equivalent to $\mathcal{O}(N \log N)$ under the assumption that the average document length L_d is constant.

A further advantage of this method is that the number of edges per vertex is limited a priori, since it is infeasible to return the similarity with all documents in the inverted index for every document. This allows the construction of graphs that are already sparse, rather than performing graph sparsification to obtain a sparse graph from the fully connected graph, e.g. [19].

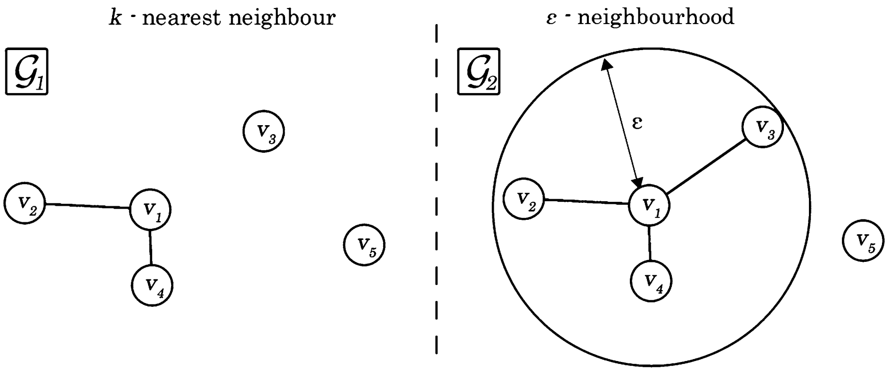


Fig. 1 Illustration of an example where two graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, are being constructed using the two methods we investigate: k-nearest neighbour and ϵ-neighbourhood. In the example, the possible edges for vertex v_1 are being considered. The k-nearest neighbour method ranks the closeness of the adjacent vertices with respect to a given similarity measure, then adds edges to the closest k vertices. For this example, $k = 2$. The ϵ-neighbourhood method adds all edges which connect v_1 to a vertex inside the large circle which visualises the radius ϵ

We investigate two popular local neighbourhood methods, k-nearest neighbour (kNN) and ϵ-neighbourhood, for keeping the graph sparse during the initial construction phase and also when new vertices are added to the graph [7, 19, 23]. Figure 1 shows intuitively how each of the methods chooses the edges to add for a given vertex. The first method, kNN, connects each vertex to the k most similar vertices in V, excluding itself. That is, for two vertices v_i and v_j, an edge is added if and only if the similarity between v_i and v_j is within the largest k results for vertex v_i. The second method we investigate, ϵ-neighbourhood, connects all vertices within a distance ϵ of each other, a similar approach to classical Parzen windows in machine learning [25]. This places a lower bound on the similarity between any two neighbouring vertices, i.e. only edges with a weight above the threshold ϵ are added to the graph. A simple way of visualising this is by drawing a sphere around each vertex with radius ϵ, where any vertex falling within the sphere is a neighbour of the vertex. While the first method fixes the degree distribution of the network, the second does not, resulting in fundamentally different topologies. We will investigate the effect of these topologies on labelling accuracy.

3 Label Propagation

Label propagation aims to use a graph $\mathcal{G} = (V,E,W)$ to propagate labels from labelled vertices to unlabelled vertices. Each vertex v_i can have multiple labels, i.e. a document can have multiple annotations, and each label is considered independently of the other labels assigned to a vertex. The labels assigned to the set of labelled vertices $\mathcal{Y}_l = \{y_1, \ldots, y_l\}$ are used to estimate the labels $\mathcal{Y}_u = \{y_{l+1}, \ldots, y_{l+u}\}$ on the unlabelled set.

Carreira-Perpinan et al. [7] suggest constructing graphs from ensembles of minimum spanning trees (MST) as part of their label propagation algorithm, with their two methods Perturbed MSTs (PMSTs) and Disjoint MSTs (DMSTs), having a complexity of approximately $\mathcal{O}(TN^2 \log N)$ and $\mathcal{O}(N^2(\log N + t))$ respectively, where N is the number of vertices, T is the number of MSTs ensembled in PMSTs, and t is the number of MSTs used in DMSTs, typically $t << \frac{N}{2}$. However, to the best of the authors' knowledge, no studies have performed experiments on constructed graphs with more than several thousand vertices, with the exception of Herbster et al. [18] who build a shortest path tree (SPT) and MST for a graph with 400,000 vertices from web pages. Herbster et al. [18] also note that constructing their MST and SPT trees using Prim and Dijkstra algorithms [11], respectively, takes $\mathcal{O}(N \log N + |E|)$ time, with the general case of a non-sparse graph having a time complexity of $\Theta(N^2)$.

In this paper we adopt Online Majority Voting (OMV) [8], a natural adaptation of the Label Propagation (LP) algorithm [32], as our algorithm for the label propagation step due to its efficiency, simplicity, and experimental performance [1]. OMV is based closely upon the locality assumption that vertices that are close to one another, with respect to a distance or measure, should have similar labels. Each vertex is sequentially labelled as the unweighted majority vote on all labels from the neighbouring vertices. The time complexity for OMV is $\Theta(|E|)$, a notable reduction from the $\mathcal{O}(kN^2)$ required for LP algorithm, where k is the neighbours per vertex. The complexity being dependent on the number of edges in the graph further benefits from the a priori limit we impose upon the maximum edges per vertex, ensuring that $|E| = bN$ for some maximum edge limit b.

4 Experiments and Evaluation

We present an experimental study of the feasibility of our approach on a large dataset, the Reuters RCV1 corpus [22]. We split the corpus into a training and test set, where the test set is the last 7 weeks of the corpus, and the training set covers everything else. The test set is further subdivided into 7 test weeks. The performance was evaluated using the F_1 Score, which is the harmonic mean of the precision and recall, a widely used performance metric for classification tasks [24]. We evaluate the performance of each method on the 7 test weeks, where all previous weeks have already been added to the graph, to simulate an online learning environment. The F_1 Scores reported are the mean performance, over the 50 most common topics, averaged over the 7 test weeks.

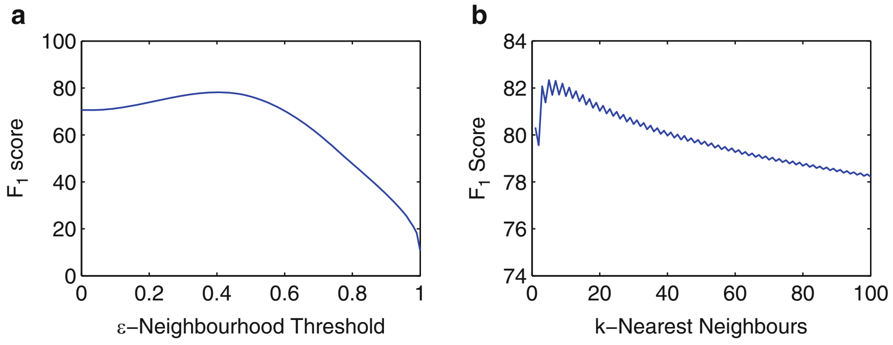


Fig. 2 F_1 performance for the 50 most common topics evaluated on the training set using LOO-CV for (a) $\epsilon = \{0, 0.01, \ldots, 1.00\}$ and (b) $k = \{1, 2, \ldots, 100\}$. It can be seen that there is a peak at $\epsilon = 0.4$ and $k = 5$

4.1 Graph-Based Content Classification

For the graph-based methods, the hyperparameters k and ϵ require careful selection in order to achieve comparable performance with current methods. This is the most expensive step as it often requires a search of the parameter space for the best value. We use Leave-One-Out Cross Validation (LOO-CV) on the training set to tune the parameters. This involves constructing graphs for a range of values of ϵ and k on the training set by iterating over all vertices, predicting the labels of the vertex based upon the majority vote of its neighbours. The predictions are checked against the true labels, with the highest performing parameter value being chosen. The performance for values of ϵ and k can be seen in Fig. 2(a) and 2(b). The best parameters for each topic individually were also recorded, allowing for a multi-parameter graph where each topic label uses a different parameter value. This could informally be thought of as each label being able to travel a certain distance along each edge. Figure 3(a) and 3(b) show the performance difference on each topic between using the general parameter values $\epsilon = 0.4$ and $k = 5$ for every topic, and using the optimal value found for each topic individually. It can be seen that for some topics a small increase in performance can be achieved, but the performance gain is minimal (with some loss for ϵ-Neighbourhood) at the expense of constructing multiple graphs, and so this approach is not considered further. Figure 4 shows a direct comparison of the graph-based methods with each other. Out of the 50 most common topics, kNN has a higher performance on 46 of the possible 50 topics. Clearly, ϵ-Neighbourhood is the weaker of the graph-based methods.

4.2 Comparison with Content-Based SVMs Performance

A comparison of the graph-based methods with the current state of the art in content-based classification was performed. The SVMs were deployed using the

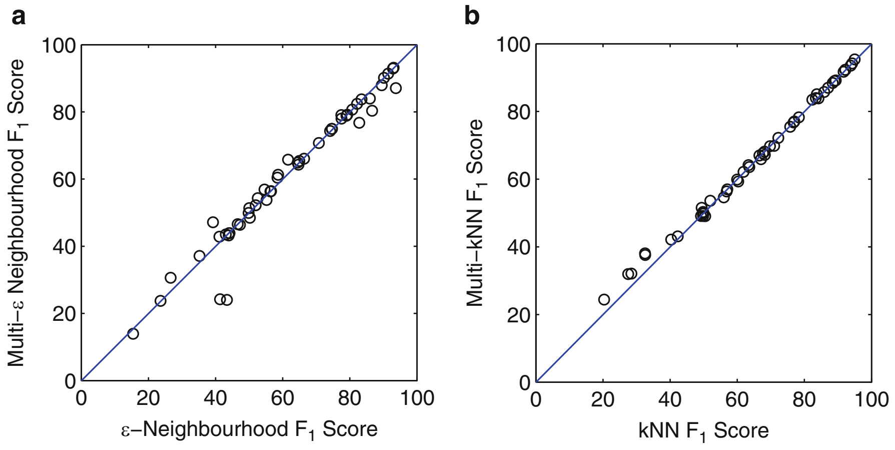
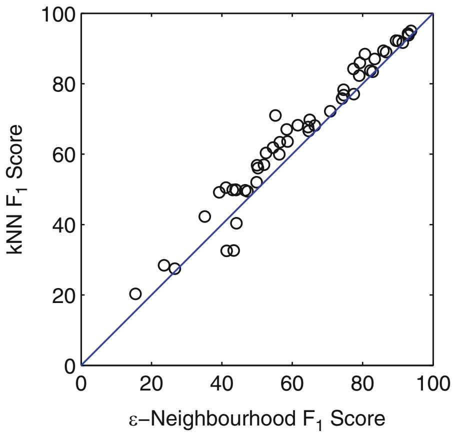
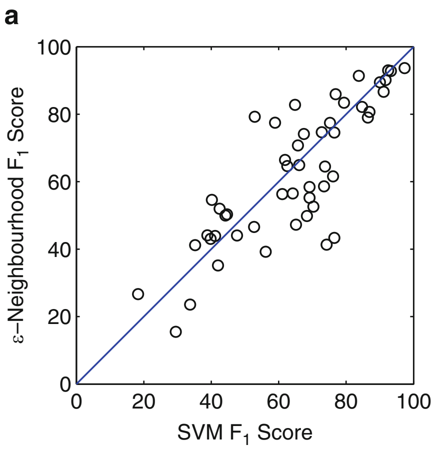


Fig. 3 Comparison of the mean F_1 Score, averaged over all test set weeks, for the graph-based methods with a single best parameter against a multi-parameter approach on the 50 most common topics. Points below the diagonal line indicate when the single parameter method achieved a higher performance, with points above the diagonal line indicating that the multi-parameter method achieved a higher performance on that topic

Fig. 4 Comparison of the mean F_1 Score, averaged over all test set weeks, for the graph-based methods on the 50 most common topics. Points below the diagonal line indicate when ϵ-Neighbourhood achieved a higher performance than *k*NN, with points above the diagonal line indicating that *k*NN achieved a higher performance than ϵ-Neighbourhood on that topic



LibSVM toolbox [10]. We trained one SVM per topic using the Cosine kernel, which is a normalised version of the Linear kernel [28]. For each topic, training used a randomly selected 10,000 positive examples, and a randomly selected 10,000 negative examples picked from the training set. The examples were first stemmed and stop words were removed as for the graph-based methods. The last week of the training corpus was used as a validation set to empirically tune the regularisation parameter C out of the set [0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100]. For each topic, C was tuned by setting it to the value achieving the highest F_1 performance on that topic in the validation set. Figure 5(a) and 5(b) show a comparison of



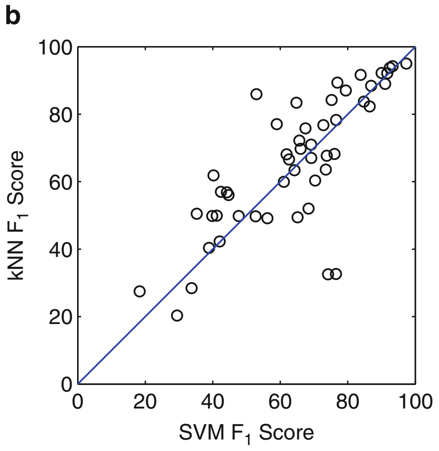


Fig. 5 Comparison of the mean F_1 Score, averaged over all test set weeks, for (**a**) ϵ-Neighbourhood and (**b**) k-NN against SVMs on the 50 most common topics. Points below the diagonal line indicate when SVMs achieved a higher performance than the graph-based method, with points above the diagonal line indicating that the graph-based method achieved a higher performance than SVMs on that topic

the graph-based methods with SVMs. Out of the 50 most common topics, SVM achieved a higher performance than ϵ-Neighbourhood on 29 topics, but only beat kNN on 19 of the topics, that is kNN performed better than SVMs on 31 out of the 50 topics. This shows that the graph-based methods are competitive with the performance of SVMs.

4.3 Building Ensembles of Graph-Based and Content-Based Approaches

Further to the comparison of the graph-based methods with SVMs, an ensemble [12] of the graph-based and content-based classification methods was evaluated. For each vertex, a majority vote for each class label c is taken by counting the supporting votes from k votes of the kNN method, supplemented with s votes from the SVMs for a total of $\upsilon = k + s$ votes. That is, each vertex has the k votes from the kNN method, but also s votes assigned by the SVMs. The number of votes from the SVM is chosen in the interval $s = [0, k+1]$. This moves the combination method from purely graph-based at $s = 0$ ($\upsilon = k$), to purely content-based at $s = k+1$ ($\upsilon = 2k+1$).

Given a set of p class labels $C = \{c_1, c_2, \ldots, c_p\}$, a set of n vertices $V = \{v_1, v_2, \ldots, v_n\}$, a graph matrix $A \in \{0,1\}^{n \times n}$ where $A_{i,j}$ indicates whether v_j is a neighbour of v_i, a label matrix $Y \in \{0,1\}^{n \times p}$ where $Y_{j,c}$ indicates if vertex v_j

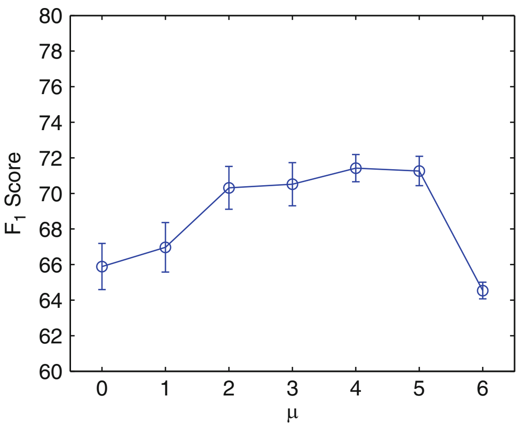


Fig. 6 Comparison of the mean F_1 Score, averaged over all test set weeks, for the combined method at different μ values on the 50 most common topics. It can be seen that the combined method offers an improvement over the kNN approach ($\mu = 0$) and the SVM approach ($\mu = 6$)

has class label c, an SVM assigned label matrix $S \in \{0,1\}^{n \times p}$ where $S_{i,c}$ indicates if class label c has been assigned to vertex v_i by the SVMs and a regularisation parameter $\lambda = [0,1]$, $\widetilde{Y}_{i,c}$ is the decision whether label c is to be assigned to vertex v_i. Formally, a linear combination of the methods was created as

$$\widetilde{Y}_{i,c} = \theta\left(\lambda \sum_j (A_{i,j} Y_{j,c}) + (1-\lambda) S_{i,c}\right) \tag{1}$$

$$\theta(x) = \begin{cases} 1 & \text{if } x > \frac{\upsilon}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Equation 1 can be reformulated so that it is easier to interpret by setting $\mu = \frac{1-\lambda}{\lambda}$, giving

$$\widehat{Y}_{i,c} = \theta\left(\sum_j (A_{i,j} Y_{j,c}) + \mu S_{i,c}\right) \tag{3}$$

where μ represents the number of SVM votes s in the interval $[0, k+1]$.

For our experiments, the value of μ for combining the kNN and SVM methods was evaluated between 0 and 6 since the kNN method uses $k = 5$ neighbours.

Next, we consider the best value of μ for combining the kNN methods and SVMs in a linear combination. Figure 6 shows the performance of the combined method averaged over the 50 most common topics for each value of μ. Out of the 50 most common topics, the combined method with $\mu = 4$ provided an improvement over the performance of both the SVM and kNN methods for 36 of the topics. Using $\mu = 1$ showed an improvement over both methods for the greatest number of topics, with 38 of the 50 topics seeing an improvement. The mean performance of the combined method with $\mu = 1$ is lower than for $\mu = 4$ however, indicating that when $\mu = 4$ the improvements are greater on average, even if there are slightly fewer of them.

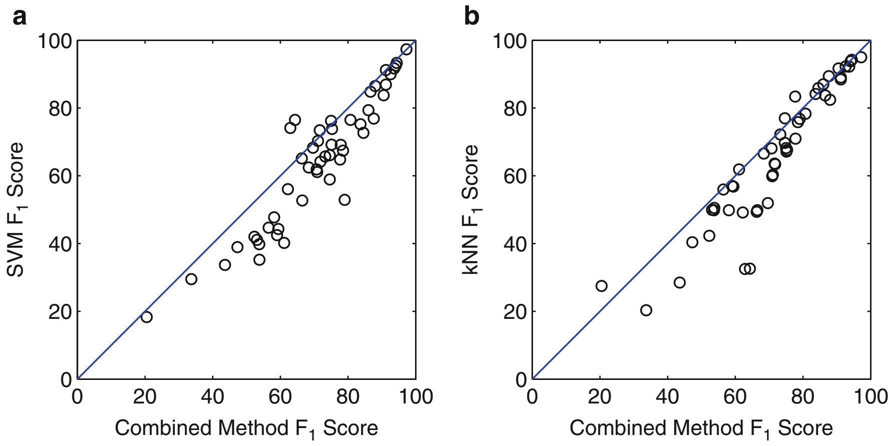


Fig. 7 Comparison of the mean F_1 Score, averaged over all test set weeks, for the combined method using $\mu = 4$ against (**a**) SVMs and (**b**) *k*NN on the 50 most common topics. Points below the diagonal line indicate when the combined method achieved a higher performance, with points above the diagonal line indicating that the individual method achieved a higher performance than the combined method on that topic

When comparing the combined method with SVM and *k*NN as seen in Fig. 7(a) and 7(b) respectively, the performance of the combined method was higher than SVM on 45 of the 50 topics and higher than *k*NN on 41 out of the 50 topics. This shows that the combined method does not only improve on SVM and *k*NN on average, but also provides an improvement for 90% and 82% of the 50 most common topics, respectively. It should be noted that in the cases where the combined method does not provide an improvement on one of the methods, it does still have a higher performance than the lowest performing method for that topic. That is, there were no cases where combining the methods gives a performance below both of the methods individually.

A summary of the overall performance of each method can be seen in Fig. 8. The ϵ-Neighbourhood method is the weaker of the two methods proposed with a performance of 62.2%, while the *k*NN method achieved a performance of 65.9%, beating the 64.5% for SVMs. Combining the *k*NN and SVM methods reached the highest performance at 71.4% with $\mu = 4$, showing that combining the relation-based and content-based approaches is an effective way to improve performance.

5 Conclusion

We have investigated a scalable method for annotating a large and growing corpus. This is achieved by efficiently creating a sparse graph and propagating the labels along its edges. In our case study the edges were created by using bag of words

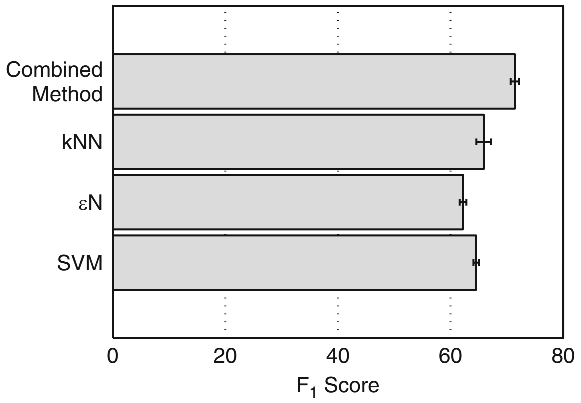


Fig. 8 Summary of the mean F_1 Score, averaged over all test set weeks, for the graph-based methods and SVMs along with the best combined method ($\mu = 4$) on the 50 most common topics. It can be seen that the graph-based methods are comparable with SVMs, with the combined method showing a further improvement. It should be noted that the performance of the combined method is slightly bias due to selecting for the best μ. ϵ-Neighbourhood has been abbreviated to ϵN

similarity, but potentially we could use any other measure that is correlated to the labels being propagated. There has been an increased theoretical interest on methods of label propagation, and some interest on how graph construction interplays with the propagation algorithms. Findings suggest that the method of graph construction cannot be studied independently of the subsequent algorithms applied to the graph [23]. We claim that label propagation has many advantages over the traditional content-based approach such as SVMs. New labels that are introduced into the system can be adopted with relative ease, and will automatically begin to be propagated through the graph. In contrast, a new SVM classifier would need to be completely trained to classify documents with the new class label. A second advantage of label propagation is that incorrectly annotated documents can be reclassified based upon new documents in a self-regulating way. That is, the graph is continuously learning from new data and improving its quality of annotation, while the SVM is fixed in its classification after the initial training period. In this paper, we have investigated two different local neighbourhood methods, ϵ-Neighbourhood and k-Nearest Neighbour, for constructing graphs for text. We have shown that sparse graphs can be constructed from large text corpora in $\mathcal{O}(N \log N)$ time, with the cost of propagating labels on the graph linear in the size of the graph, i.e. $\mathcal{O}(N)$. Our results show that the graph-based methods are competitive with content-based SVM methods. We have further shown that combining the graph-based and content-based methods leads to an improvement in performance. The proposed methods can easily be scaled out into a distributed setting using currently available open source software such as Apache Solr, or Katta, allowing a user to handle millions of texts with similarly effective performance. Research into novel ways of combining the relation and content-based methods could lead to further improvements in the categorisation performance while keeping the cost of building and propagating labels on the graph to a minimum.

Acknowledgements I. Flaounas and N. Cristianini are supported by FP7 under grant agreement no. 231495 (ComPLACS Project). N. Cristianini is supported by Royal Society Wolfson Research Merit Award. All authors are supported by Pascal2 Network of Excellence.

References

1. Ali, O., Zappella, G., De Bie, T., Cristianini, N.: An empirical comparison of label prediction algorithms on automatically inferred networks. In: Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods, SciTePress, pp. 259–268 (2012)
2. Angelova, R., Weikum, G.: Graph-based text classification: learn from your neighbors. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 485–492. ACM, New York (2006)
3. Araujo, M., Navarro, G., Ziviani, N.: Large Text Searching Allowing Errors In: Proceedings of the 4th South American Workshop on String Processing (WSP'97), pp. 2-20. Carleton University Press (1997)
4. Baeza-Yates, R., Navarro, G.: Block addressing indices for approximate text retrieval. J. Am. Soc. Inform. Sci. **51**(1), 69–82 (2000)
5. Bayardo, R., Ma, Y., Srikant, R.: Scaling up all pairs similarity search. In: Proceedings of the 16th International Conference on World Wide Web, pp. 131–140. ACM, New York (2007)
6. Belkin, M., Matveeva, I., Niyogi, P.: Regularization and semi-supervised learning on large graphs. In: Learning Theory, pp. 624–638. Springer, Berlin (2004)
7. Carreira-Perpinan, M., Zemel, R.: Proximity graphs for clustering and manifold learning. In: Advances in Neural Information Processing Systems 17. NIPS-17, MIT Press (2004)
8. Cesa-Bianchi, N., Gentile, C., Vitale, F., Zappella, G.: Random spanning trees and the prediction of weighted graphs. In: Proceedings of ICML, Citeseer, OmniPress, pp. 175–182 (2010)
9. Cesa-Bianchi, N., Gentile, C., Vitale, F., Zappella, G.: Active Learning on Graphs via Spanning Trees, In NIPS Workshop on Networks Across Disciplines (2010)
10. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**, 27:1–27:27 (2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
11. Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms, MIT Press and McGraw-Hill (1990)
12. Dietterich, T.: Ensemble methods in machine learning. Multiple Classifier Systems, LNCS, Vol. 1857, Springer, pp. 1–15, (2000)
13. Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th International Conference on World Wide Web, pp. 577–586. ACM, New York (2011)
14. Flaounas, I., Ali, O., Turchi, M., Snowsill, T., Nicart, F., De Bie, T., Cristianini, N.: NOAM: news outlets analysis and monitoring system. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 1275–1278. ACM, New York (2011)
15. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 518–529. Morgan Kaufmann Publishers Inc., Los Altos (1999)
16. Heaps, H.: Information Retrieval: Computational and Theoretical Aspects. Academic, Orlando (1978)

17. Herbster, M., Pontil, M.: Prediction on a graph with a perceptron. Adv. Neural Inform. Process. Syst. **19**, 577 (2007)
18. Herbster, M., Pontil, M., Rojas-Galeano, S.: Fast prediction on a tree. Adv. Neural Inform. Process. Syst. **21**, 657–664 (2009)
19. Jebara, T., Wang, J., Chang, S.: Graph construction and b-matching for semi-supervised learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 441–448. ACM, New York (2009)
20. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Machine Learning: ECML-98, Springer, pp. 137–142 (1998)
21. Lansdall-Welfare, T., Flaounas, I., Cristianini, N.: Scalable corpus annotation by graph construction and label propagation. In: Proceedings of the 1st International Conference on Pattern Recognition Applications and Method, SciTePress, pp. 25–34 (2012)
22. Lewis, D., Yang, Y., Rose, T., Li, F.: RCV1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. **5**, 361–397 (2004)
23. Maier, M., Von Luxburg, U., Hein, M.: Influence of graph construction on graph-based clustering measures. Adv. Neural Inform. Process. Syst. **22**, 1025–1032 (2009)
24. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
25. Parzen, E.: On estimation of a probability density function and mode. Ann. Math. Statist. **33**(3), 1065–1076 (1962)
26. Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 743–754. ACM, New York (2004)
27. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. (CSUR) **34**(1), 1–47 (2002)
28. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
29. Yang, Y., Zhang, J., Kisiel, B.: A scalability analysis of classifiers in text categorization. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pp. 96–103. ACM, New York (2003)
30. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: a comprehensive study. Int. J. Comput. Vision **73**(2), 213–238 (2007)
31. Zhu, X.: Semi-supervised learning literature survey. In: Computer Science. University of Wisconsin-Madison. Madison (2007)
32. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: International Conference of Machine Learning, AAAI Press, vol. 20, p. 912 (2003)