Two Stage SVM and kNN Text Documents Classifier

Marcin Kępa and Julian Szymański(✉)

Department of Computer Systems Architecture,
Gdańsk University of Technology, Gdańsk, Poland
markepa@pg.gda.pl, julian.szymanski@eti.pg.gda.pl

Abstract. The paper presents an approach to the large scale text documents classification problem in parallel environments. A two stage classifier is proposed, based on a combination of k-nearest neighbors and support vector machines classification methods. The details of the classifier and the parallelisation of classification, learning and prediction phases are described. The classifier makes use of our method named *one-vs-near*. It is an extension of the *one-vs-all* approach, typically used with binary classifiers in order to solve multiclass problems. The experiments were performed on a large scale dataset, with use of many parallel threads on a supercomputer. Results of the experiments show that the proposed classifier scales well and gives reasonable quality results. Finally, it is shown that the proposed method gives better performance compared to the traditional approach.

Keywords: SVM · k-nearest neighbor · Wikipedia · Documents categorization · Parallel classification

1 Introduction

Since the beginning of the Internet, its size and the amount of globally stored data has been growing. With every year, the estimated number of indexed web pages is increasing and today it is somewhere between 20 and 50 billion pages [1]. Because of the size of an average dataset, a need for automatic categorization arises. Repositories, such as Wikipedia, reaching 4.5 million articles, organized with hundreds of thousands of categories, could benefit from automatic categorization. There are many existing approaches to this problem, with different results both in terms of accuracy and performance [2–4], but there is still need for improvements in this area.

The aim of the work presented here is to propose a two stage classifier, capable of automatic categorization of text documents, from repositories containing over 100 k categories and millions of articles. The proposed classification is performed in two stages. The first one is a fast, initial classification stage, done by the k-nearest neighbours (kNN) classifier, where the dataset is limited to selected categories. The second stage is the final, accurate classification stage, done by

© Springer International Publishing Switzerland 2015
M. Kryszkiewicz et al. (Eds.): PReMI 2015, LNCS 9124, pp. 279–289, 2015.
DOI: 10.1007/978-3-319-19941-2_27

the support vector machines (SVM)classifier, trained on the limited dataset. The experiments designed to evaluate our approach are performed using Wikipedia data, processed with our application that allows us to construct its machine-processable representation [5]. The original contribution of this paper is the application of our method named *one-vs-near* in classification of large scale text documents repositories. This is done in order to improve the performance of a typical linear SVM in the *one-vs-all* setting.

The next section briefly describes SVM and kNN classifiers and the way they are incorporated to solve multiclass classification problems. Section 3 presents the details of the solution. Then, Sect. 4 briefly describes the Galera supercomputer, used to test the performance of our classifier in highly parallel environments and with big datasets. The experiments using our implementation, along with empirical results based on Wikipedia datasets, are given in Sect. 5. The last section summarizes the paper and gives ideas for future research in this area.

2 kNN and SVM Classifiers in a Typical Multiclass Setting

2.1 kNN in Multiclass Setting

One of the simplest, as well as the oldest machine classification techniques, is the approach called kNN [6]. In a typical setting each test object is assigned to a certain class, based on majority of its k nearest neighbors [6,7]. However, this approach can be computationally expensive for datasets containing millions of test objects. Some papers have shown [7,8] that kNN classifiers trained with the use of pre-labled examples can highly improve the quality of classification. Since a standard kNN approach can be very demanding performance-wise, modified solutions are introduced, eg. the centroid kNN [3]. The idea is to calculate a centroid for groups of feature vectors belonging to the same category and apply the similarity metric on these centroids, instead of on each feature vector individually. Given a set of S documents we can define the centroid vector as:

$$\boldsymbol{C} = \frac{1}{|S|} \times \sum_{d \varepsilon S} \boldsymbol{d} \tag{1}$$

where $|S|$ is the number of articles in a class S and $\boldsymbol{d}$ are the vectors representing the articles. After computing the centroids, we can use any similarity metric to compare them in the prediction phase. The complexity of prediction in such case (assigning labels to m_{test} test objects) is at most $O\left(m_{\text{test}} \cdot N\right)$, where N is the number of categories. The complexity of computing the model is at best only $O\left(m_{\text{train}}\right)$, where m_{train} is the number of training examples.

2.2 SVM in Multiclass Setting

SVM's are one of the most effective methods of text classification [9]. In its base form an SVM is a binary classifier that constructs a hyperplane $h()$ in a high dimensional feature space (examples are typically projected into that space by a kernel function), which is convex-optimized during training so that it separates the classes leaving maximal possible space (margins) between them. The prediction step can be summarized in a simple equation $a = h(x)$, where a is the activation of the hyperplane, and x is the feature vector (possibly transformed by a kernel) of a testing object. The sign of a decides which class is predicted, whereas the absolute value of a indicates the confidence of this decision. With advanced optimization algorithms used by an SVM, time complexity of training such hyperplane is $O\left(m_{\text{train}}\right)$. Although there are attempts to directly deal with multiclass problems using reformulated SVMs [10], most often such problems are divided into binary classifications and incorporate typical SVM classifiers summarized above.

In a popular *one–vs–all* scheme for each class a separate hyperplane is trained, by treating examples from that class as positives and all the remaining examples in the dataset as negatives. During prediction, a test object is assigned to a class which hyperplane's activation a is the highest (*winner takes all* strategy). Complexity of calculating the whole model in this setting is $O\left(m_{\text{train}} \cdot N\right)$. For such a classifier the prediction can be performed in a $O\left(m_{\text{test}} \cdot N\right)$ time, the same as for the kNN classifier. The comparative study of this multiclass SVM setting, as well as other less popular ones, can be found in [11]. It is important to note that the classification of Wikipedia belongs to a multi-label family of problems. In such cases, the *winner takes all* algorithm is replaced, each article is tested against every category in the dataset and the final result consists of categories with activation scores that exceed a specified threshold.

2.3 Hybrid Approaches

In order to improve the effectiveness of classification hybrid approaches of kNN and SVM are introduced. The approaches vary in the way the classification stages are combined and the types of datasets used. One of such methods is the HKNNSVM classifier, proposed in [12] that improves kNN classier's accuracy by limiting the dataset only to the support vectors of each category's hyperplane. It should be noticed the accuracy of the kNN classifier is slightly increased in this approach. Both the training and the prediction phase of the HKNNSVM require a bigger amount of computations than in our approach, which might be a problem when classifying big repositories such as Wikipedia.

Another approach proposed in [13] uses the kNN classifier to select the nearest neighbours for a given query. An SVM classifier (DAGSVM) is then employed in order to make the final decision. The classifier shows excellent accuracy in character recognition however, a similar approach wouldn't be as effective in the case of large scale text documents classification. The initial search for nearest neighbours amongst millions of articles, each containing thousands of features, would be very demanding performance-wise. Moreover, the fact that an SVM has to be trained for each query is also an issue in a dataset containing millions of examples. Because of that our method should prove to be more efficient at classifying sparse textual datasets. A similar approach to character recognition is also proposed in [14], however, just as the previous solution, it is not practical for a large and sparse dataset such as Wikipedia.

A different approach is also proposed in [15]. The solution uses the kNN rule in order to assign real value weights to the examples in the training dataset. This is unlike the standard SVM, where examples belonging to a class are assigned a 1 and all the others are assigned a -1. Just as in the previous examples this approach proves to be more accurate than a single SVM. Again, the computational complexity of this approach makes it impractical in case of large scale text documents classification. It is worth mentioning that unlike the other solutions our approach deals with multi-label problems.

3 Details of Our Approach

Our classification system consists of four modules: the data preparation module, the initial classification module, the final classification module, and the results evaluation module. The results of every stage are saved in the file system, which allows us to run the stages independently. The data preparation module is designed to filter the dataset in order to meet requirements of the classifier. The data evaluation module consists of programs designed to return quality scores of the classifier such as its precision and recall.

3.1 Two Stage Classification

The classifier uses the *one–vs–near* approach instead of the *one–vs–all* approach in order to limit the dataset during the learning phase [16]. For each category, for which the classifier is trained, the dataset is limited to its closest neighbours. The category neighbour list (used for limiting the dataset) is computed by the kNN classifier in the first stage. It is important for the initial stage to be lightweight, in order to minimize its impact on the overall performance of the classifier. The kNN computes the distances between every centroid in the form of an ordered list. This list is then saved in the filesystem and later used by the second stage classifier. The second stage SVM uses the saved neighbour list to limit the dataset used for training the classifier for a single category. Apart from this, the second stage classifier works as a standard *one–vs–all* approach for SVM. However, thanks to this difference it is possible to greatly limit the training dataset size for each binary classifier. Because of that, the training performance should be improved. Furthermore, the accuracy of the resulting classifier should be comparable to one trained on the entire dataset. This two stage approach is presented in Algorithm 1.

Algorithm 1. Two stage training

```
1. Get a category to train or end if no more categories exist
2. Get neighbours of that category from the neighbours list
3. Prepare the dataset containing only articles from neighbouring categories
4. Train the SVM classifier on that dataset and go to 1.
```

3.2 Parallelisation of the Computations

One of the main goals of our research is for the final classifier to be easily scalable. Both the training and the prediction tasks related to each SVM hyperplane and kNN centroid are intrinsically independent, therefore the job of dividing the problem between parallel compute nodes is straightforward. Each node is on its own responsible for downloading tasks from a task queue. Each task is in fact either a category to train (in the training phase) or an article for which classes are to be predicted (in the prediction phase). Each compute node picks up tasks from the task queue in batches.

In addition to machine level parallelisation, each node runs its computations in parallel threads. Managing to distribute the training and prediction procedures related to all classes over different compute nodes allows us to construct a scalable classifier. The classifier accesses its files through a Network File System (NFS) so that every machine works in the same directory and has access to the same files. As mentioned before, the jobs to be done are stored in a single file queue – the TODO file. The TODO file contains names of hyperplanes to train, in case of training and a list of objects to predict labels for, in case of prediction. Every node can obtain a certain number of jobs from the TODO file and run these jobs using available cores. Having done that it can receive new jobs and so on. Synchronization between nodes is obtained using Message Passing Interface (MPI) implementation Open MPI [17].

Because many parallel nodes need access to the TODO file, there is a need for some synchronization mechanism. The solution to this problem is to use the MPI in order to implement a master–slave scheme. The processes of the application are divided into a single master process and many slave processes. The master process is used to distribute the tasks between the slave processes and the slave processes are in turn used to conduct the computations. The access to the TODO file is granted to slave processes by the master process. Each slave has to request access to the queue from the master before downloading its tasks.

4 Test Environment

In order to test our approach on big datasets a parallel computations environment was needed. The classifier was tested on the Galera supercomputer, in Academic Computer Centre (CI TASK), part of Gdansk Univeristy of Technology. The cluster consists of 1344 2,33 GHz Intel Xeon QuadCore processors (5376 cores), 25 TB total system memory, 100 TB disk storage and Mellanox InfiniBand interconnect with 20 Gb/s bandwidth. The cluster is operated under a Linux family operating system. The total theoretical peak performance of the cluster is 50 TFLOPS. Upon its launch, the cluster performance was measured to be 38.2 TFLOPS.

The environment is configured to use the message passing interface (MPI) implementation for communication between different nodes of the cluster. The tasks are queued for execution with a portable batch system (PBS) based queue. For the purpose of this work only a fraction of the cluster was used, comprising of 500 cores. This was more than enough to test the classifier in a massively parallel environment. The results of these experiments can be seen in the next section.

5 Experiments

To evaluate the effectiveness of our approach a series of tests was performed. They were planned to check performance, scalability and F-score of the classifier. Initial tests have been conducted with smaller size data and without cross validation. The final tests have been performed using large scale datasets and with evaluation based on cross validation. The datasets used in this paper were created from the entire Wikipedia, based on 8th March 2013 dump [18]. This dump was processed using Matrix'u application [5] in order to create a bag of words [19] representation of the dataset. The dataset was then filtered, which among other things deleted administrative categories and merged small categories with their parents. Remaining very small categories (for small categories there is not enough examples in order to train an accurate classifier) were removed.

5.1 KNN and SVM Training Scalability

The first test was designed in order to test the scalability of the solution by comparing training phase performance of both classifiers (the initial kNN and the final SVM classifier). The dataset for this test was limited to 530 categories, containing 853 283 documents. Apart form testing the scalability in a highly parallel environment (between 8 and 160 logical processors) another important result of this test is the comparison of the fast initial kNN classifier and the final accurate SVM classifier. The results in this test represent only the time needed for creating the classification models. All additional time is subtracted form the results, they can be seen in Fig. 1. As expected the fast initial classifier is faster by an order of magnitude. This result shows that it is indeed feasible to use the kNN classifier in order to conduct a fast initial classification and then to use that data to improve performance of the SVM classifier. Moreover, the scalability of both classifiers is very good.

5.2 Classification Quality for Big Data

After performing the scaling tests for small data, the next step was to run the classifier on the entire Wikipedia. After filtering, the dataset for this test consisted of 2 992 212 articles grouped in 18 335 categories. The quality of the classifier was validated in 10-fold cross validation. The calculated values are the precision, recall and the F-score. In order to compare the quality of the stage classifier with the standard one–vs–all approach, as well as the centroid kNN classifier, all three classifiers were trained on the same dataset. Together all the models trained for a single classifier took around 50 gigabytes of disk space.

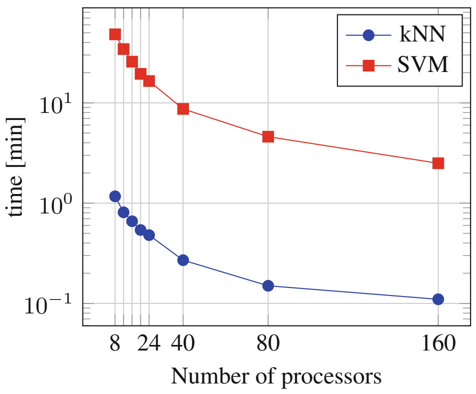

Fig. 1. SVM training performance on Galera cluster

First, the one–vs-all SVM classifier was run. In order to find best results (modify the recall of the classifier), different acceptance thresholds were tested. This means that the SVM classification rule was altered, by an additional parameter t, changing the hyperplanes $h(x)$ position:

$$Category(x) = sign(h(x) - t) \tag{2}$$

Additional explanation of this threshold, as well as more advanced approaches to its optimization can be found in [20]. The results of this test can be seen in Table 1. As we can see, the optimal results were achieved for 0.05 acceptance threshold, further increase of the threshold gave better precision but the recall suffered greatly. On the other hand, decreasing the threshold gave better recall, but the precision deteriorated quickly.

As mentioned before, the same dataset was classified using the centroid kNN classifier, with cosine similarity used as the distance metric. The F-score of this classification is much lower however, it is still acceptable. For the kNN classifier the acceptance threshold is the minimal level of cosine similarity, between a category and the article feature vector, used to determine whether it belongs to that category. The recall (depending on the acceptance threshold) was as high as 50 %, which is still usable. On the other hand, limiting the results even more, by increasing the acceptance threshold, gave good precision of over 50 % with small

Table 1. SVM classifier precission

Accept thresh.	True pos.	False pos.	False neg.	Precision	Recall	F-score
0.30	3 681 502	1 700 894	7 033 603	68.39 %	34.35 %	45.74 %
0.05	4 239 491	2 905 837	6 475 614	59.33 %	39.56 %	**47.47 %**
0	4 365 491	3 405 395	6 349 614	56.17 %	40.74 %	47.23 %

Table 2. kNN classifier precission

Accept thresh.	True pos.	False pos.	False neg.	Precision	Recall	F-score
0.2	4 831 716	22 737 151	5 883 389	17.52 %	45.09 %	25.06 %
0.3	3 542 955	12 079 258	7 172 150	22.67 %	33.06 %	**26.90 %**
0.9	1 644 218	1 431 920	9 070 887	53.45 %	15.34 %	23.84 %

Table 3. Two stage classifier precission

Accept thresh.	True pos.	False pos.	False neg.	Precision	Recall	F-score
0.1	4 127 571	6 161 031	6 587 534	40.11 %	38.52 %	39.30 %
0.3	3 676 520	2 819 521	7 038 585	56.59 %	34.31 %	**42.72 %**
0.35	3 572 710	2 441 656	7 142 395	59.40 %	33.34 %	42.71 %

recall of 15 %. Some example results in relation to the acceptance threshold can be seen in Table 2.

Finally the two stage classifier was tested on the same dataset. Based on the results from small data tests, the amount of neighbours was set to 30 % of categories. The results of this test can be seen in Table 3. The F-score is considerably better than for the kNN classifier and slightly lower than for a one–vs–all solution (but with much better performance). Furthermore, it is worth noting that the two stage classifier achieved F-score comparable to classifiers that took part in the Pascal Large Scale Hierarchical Text Classification Challenge (LSHTC3) [21]. Comparing results form this paper to the ones from Wikipedia based tasks in LSHTC3, we can see that these values are very similar. For example, the best F-score for Wikipedia datasets in LSHTC3 was 49 % for the medium and 45 % for the large dataset.

5.3 Performance for Big Data

Another important experiment was to test the performance of the training and the prediction phase for all three classifiers. The results presented in Fig. 2 are for the same dataset as before, with 18 335 categories and 2 992 212 articles. The tests were conducted with 62 compute nodes, each with 8 logical processors, giving 496 processors in total. The results are averaged in 10-fold cross validation. We can clearly see that in the training phase the two stage approach is considerably faster, than the traditional one–vs–all classifier. Although the centroid kNN approach presents poor precision, the training time is shorter by an order of magnitude. This means that this approach could still be useful in certain cases where lower precision is not an issue. All in all, the multi stage approach presents itself as a good way to increase performance of SVM classification, while maintaining high F-score.

As mentioned in the previous sections all presented classifiers are comparable when it comes to computational complexity of the prediction phase. The

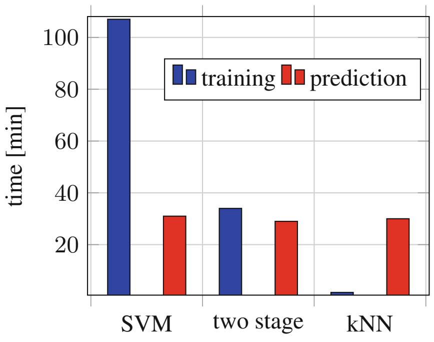

Fig. 2. Performance comparison of different solutions for big data

prediction times for the entire dataset are therefore very similar for the three approaches. The calculated times are as follows: 29 min for the two stage, 31 min for the SVM and 30 min for the kNN classifier.

6 Conclusions and Future Work

The aim of the research presented in this paper was to develop and evaluate a parallel multi stage approach to classification of text documents with SVM. Our solution was designed to be used with large scale text document repositories in mind, such as Wikipedia. The results of the experiments show that our approach scales up well and gives good F-score. The two stage approach based on *one–vs–near* scheme was tested on big datasets created from the entire Wikipedia. Precision and recall of our solution proved to be comparable to the typical *one–vs–all* scheme, while significantly improving the time needed for classifiers construction. Additionally it was proven that the simple centroid kNN classifier can also produce useful quality results, with classifier creation time shorter by an order of magnitude. Although the problem of text documents classification was extensively tested in many works (eg. [2,4,22]), there is still some room for further research and improvements in this area.

There are many yet untested approaches to this problem that are worth pursuing. It would be interesting to verify how substituting the initial kNN classifier with different approaches would impact the quality and performance of the classifier as well as allow to mining the relations between categories [23] . It would be also interesting to test the two stage approach with different kinds of SVM solvers and their parameters. The performance of the classifier could also be improved using an array data DBMS (such as SciDB) to store the feature vectors, instead of plain text files. This would further improve the performance of the classifier and possibly decrease the memory requirements. Also improvement of managing the threads distribution using BeesyCluster [24] can lead to results improvement.

Acknowledgments. The work was performed within grant "Modeling efficiency, reliability and power consumption of multilevel parallel HPC systems using CPUs and GPUs" sponsored by and covered by funds from the National Science Center in Poland based on decision no DEC-2012/07/B/ST6/01516.

References

1. de Kunder, M: The size of the world wide web (2014). http://www.worldwidewebsize.com/. Accessed 22 May 2014
2. Gantner, Z., Lars, S.-T.: Automatic content-based categorization of wikipedia articles. In: Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources, People's Web 2009, pp. 32–37. Association for Computational Linguistics, Stroudsburg (2009)
3. Han, E.-H.S., Karypis, G.: Centroid-based document classification: analysis and experimental results. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 424–431. Springer, Heidelberg (2000)
4. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: Liblinear: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
5. Szymański, J.: Wikipedia articles representation with matrix'u. In: Hota, C., Srimani, P.K. (eds.) ICDCIT 2013. LNCS, vol. 7753, pp. 500–510. Springer, Heidelberg (2013)
6. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**, 21–27 (1967)
7. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances in Neural Information Processing Systems 18, pp. 1473–1480. MIT Press, Cambridge (2005)
8. Draszawka, K., Szymanski, J.: Thresholding strategies for large scale multi-label text classifier. In: IEEE 2013 the 6th International Conference on Human System Interaction (HSI), pp. 350–355 (2013)
9. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998)
10. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. J. Mach. Learn. Res. **2**, 265–292 (2002)
11. Duan, K.-B., Keerthi, S.S.: Which is the best multiclass SVM method? An empirical study. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 278–285. Springer, Heidelberg (2005)
12. Vinoth, R., Jayachandran, A., Balaji, M., Srinivasan, R.: A hybrid text classification approach using KNN and SVM. Int. J. Adv. Found. Res. Comput. (IJAFRC) **1**(3), 20–26 (2014)
13. Zhang, H., Berg, A., Maire, M., Malik, J.: SVM-KNN: discriminative nearest neighbor classification for visual category recognition. In: Proceedinngs of 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2126–2136 (2006)
14. Shih, Y., Wei, D.: Machine learning final project: Handwritten sanskrit recognition using a multi-class SVM with K-NN guidance (2011)
15. Hsu, C.-C., Yang, C.-Y., Yang, J.-S.: Associating *k*NN and SVM for higher classification accuracy. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 550–555. Springer, Heidelberg (2005)

16. Balicki, J., Szymanski, J., Kępa, M., Draszawka, K., Korlub, W.: Improving effectiveness of svm classifier for large scale data. In: Proceeedings of the 14th International Conference on Artificial Intelligence and Soft Computing (in print). Springer (2015)
17. Gabriel, E., et al.: Open MPI: goals, concept, and design of a next generation mpi implementation. In: Kranzlmüller, D., Kacsuk, P., Dongarra, J. (eds.) EuroPVM/MPI 2004. LNCS, vol. 3241, pp. 97–104. Springer, Heidelberg (2004)
18. Wikipedia: Wikipedia database dump (2014). http://dumps.wikimedia.org/enwiki/20140102/. Accessed 25 January 2014
19. Szymanski, J.: Comparative analysis of text representation methods using classification. Cybern. Syst. **45**, 180–199 (2014)
20. Shanahan, J.G., Roma, N.: Improving SVM text classification performance through threshold adjustment. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 361–372. Springer, Heidelberg (2003)
21. Institute of Informatics and Telecommunications - NCSR Demokritos in Greece: Large scale hierarchical text classification challenge (2015). http://lshtc.iit.demokritos.gr/. Accessed 18 January 2015
22. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University (2003)
23. Szymański, J.: Mining relations between wikipedia categories. In: Zavoral, F., Yaghob, J., Pichappan, P., El-Qawasmeh, E. (eds.) NDT 2010. CCIS, vol. 88, pp. 248–255. Springer, Heidelberg (2010)
24. Czarnul, P.: Modeling, run-time optimization and execution of distributed workflow applications in the JEE-based beesycluster environment. J. Supercomput. **63**(1), 1–26 (2010)