Ensemble Approaches for Large-Scale Multi-Label Classification and Question Answering in Biomedicine

Yannis Papanikolaou[1], Dimitrios Dimitriadis[1], Grigorios Tsoumakas[1], Manos Laliotis[2], Nikos Markantonatos[3], and Ioannis Vlahavas[1]

[1] Aristotle University of Thessaloniki, Thessaloniki 54124, Greece
yannis.papanik@gmail.com,{dndimitr,greg,vlahavas}@csd.auth.gr
[2] Atypon, 5201 Great America Parkway Suite 510, Santa Clara, CA 95054, USA
elalio@atypon.com
[3] Atypon Hellas, Dimitrakopoulou 7, Agia Paraskevi 15341, Athens, Greece
nikos@atypon.com

Keywords: ensemble learning, multilabel learning, SVM, LDA, Metalabeler, BioASQ

Abstract. This paper documents the systems that we developed for our participation in the BioASQ 2014 large-scale biomedical semantic indexing and question answering challenge. For the large-scale semantic indexing task, we employed ensembles consisting of support vector machines, both vanilla and tuned to handle class imbalance, labeled latent Dirichlet allocation models and meta-models predicting the number of relevant labels. For the question answering task we ensembled different scorings of candidate answers based on work in recent literature.

1 Introduction

This paper discusses our approaches to the large-scale semantic indexing and question-answering tasks of the 2nd (2014) version of the BioASQ challenge. We mainly worked on the semantic indexing task. Section 2 offers background knowledge on the models and algorithms we employed. Section 3 presents our classifier selection approaches for multi-label data. Section 4 describes the actual systems we used for the challenge and the experiments we performed. Section 5 presents our results. Section 6 presents our work on the question answering task. Finally, Section 7 concludes this paper.

2 Background

This section provides a brief description of the models/algorithms used in our participation in Task 2A of the BioAsQ challenge along with the necessary theory.

2.1 Support Vector Machines

Support Vector Machines [1] have been extensively used in the literature for classification and regression tasks. Being a non-probabilistic binary classification algorithm in its essence, it has managed to achieve state-of-the art performance in numerous tasks and has been applied in multiple domains for solving learning problems. In our experiments we used the Liblinear package [2], along with some minor modifications, which fitted perfectly our needs for a very fast and scalable implementation.

2.2 MetaLabeler

The MetaLabeler [3] is essentially a meta-model employed in multilabel tasks that serves to automatically determine the cardinality of the label set for a given instance. The idea is to train a linear regression model (e.g. with an SVM) with input from some feature space (an easy option could be simply the word tokens of each instance) and output the number of labels associated to the particular instance.

The need for the above meta-model arises in multi-label problems where, given an instance, the model's output for each label is a score or a probability. In this case, every instance is associated with a ranking of labels and we need to properly set a threshold so that we get a hard-assignment of labels. It should be noted here, that apart from the metalabeler a great deal of work exists in literature to address that particular problem [4] [5] but alternative solutions usually require a crossvalidation procedure which proves to be too time-consuming for large-scale datasets. We also experimented with an approach similar to the metalabeler [6]. In this case, the output of the regression training problem is not the actual number of labels but the one that maximizes some evaluation measure (the F-measure in our case). Thus, given a trained model, we employ it on a validation set to determine the number of labels that would maximize the F-measure for every instance. Even if unintuitively this approach would do better as it captures also the misclassification errors of the classifiers, in practice results were inferior compared to the metalabeler.

2.3 Topic Models

Latent Dirichlet Allocation (LDA) is a powerful probabilistic model first introduced by [7] [8] in an unsupervised learning context. The key idea is that a corpus of documents hides a number of topics; this model, given the corpus, attempts to learn the distibution of topics to documents (namely the Θ distribution) and the distribution of topics to word tokens (Φ distribution respectively). After learning these distributions, the trained model can be used either in a generative task (e.g. given some topics, produce a new document(s)) or in an inference task (given some new documents, determine the topics they belong to). It is rather obvious to note that this model seems naturally fitted to deal with multi-label problems,

apart from the fact that, being totally unsupervised, its resulting topics may be hard to interpret.

[9] and [10] incorporated the LDA theory into a supervised learning context where each topic corresponds to a label of the corpus in a one-to-one correspondence. We implemented the LLDA and the prior LLDA variant of [10]. The only difference between the two is that the prior LLDA model takes into account the relative frequencies of labels in the corpus, a crucial fact in case of a problem with power-law statistics[4] like the one we address. In experiments, the prior LLDA model was performing significantly better than the simple LLDA so we used that one for our systems. Eventhough this model's performance didn't match that of the SVMs, we opted to use it with the motivation that it could do better for some labels and therefore used it in two ensembles (see section 4.2).

3 A classifier selection multilabel ensemble

3.1 Previous work

The main idea behind ensembles is to exploit the fact that different classifiers may do well in different aspects of the learning task so combining them could improve overall performance. Ensembles have been extensively used in literature [12] with stacking [13], bagging [14] and boosting [15] being the main methods employed. In the context of multilabel problems, [16] proposes a fusion method where the probabilistic outputs of heterogeneous classifiers are averaged and the labels above a threshold are chosen. [17] propose a classifier selection scheme based on the F-measure. For each label and for each of the classifiers the F-measure is computed and the best performing is chosen to predict that particular label. We tried the last approach and even for large validation datasets we found a systematic decline on the micro-F measure.

In this work, we propose a different method, a mostly simple and fast ensemble technique oriented towards a classifier selection (rather than fusion) scheme. Essentially, we treat the problem as having L different classification tasks and requiring to be able to tell which of the models used is more suitable for each of them. In the description below, we suppose that there is a baseline model (i.e. a model that has a better overall performance than the others) but our idea can be applied with minor modifications without this assumption.

Formally, suppose we have a baseline model A and q different models B_i and we want to combine them in a multilabel task with input feature vectors $\boldsymbol{x}$ and output $\boldsymbol{y}, \boldsymbol{y} \in L, L$ being the set of labels. Instead of choosing a voting system for all labels, we could see for which labels each B_i performs better than A on some validation set and according to some evaluation metric $eval$. Let's denote

$$L_{Bi} = \{l : eval(B_i) > eval(A), eval(B_i) > eval(B_j)\},\ with\ l \in L\ and\ j \neq i$$

[4] by referring to a dataset with power-law statistics we mean that the vast majority of labels have a very low frequency and only very few have a high frequency, for a more elaborate explanation refer to [11]

and

$$L_A = L - \sum L_{Bi}$$

respectively. Then, when predicting on unseen data, we could predict labels that belong to L_A from model A and labels belonging to each L_{Bi} from the respective model B_i.

There are two remaining issues to be solved; a) choose a valid evaluation metric *eval* and b) assure that results pointed by *eval* on a validation set can be generalised to new, unseen data. As the contest's main evaluation metric was the micro-F measure we opted for it. As mentioned, we also tried to use the F-measure (per label) but it was not improving overall performance, even on the validation dataset.

Concerning the second isssue, initially we tried to address it by just relying on using a large validation dataset. However, after obtaining unfavorable results on the competition, we relied on a siginficance test, namely a McNemar test with a confidence level of 95%. To sum up, we first predict with A (our baseline model) on a validation dataset and then for each label and for each model B_i we check if choosing B_i to predict for that label improves the overall micro-F measure. If yes, that label is candidate to belong to L_{Bi}. Then, for all labels that belong to the candidate sets, we run a McNemar test, or multiple McNemar tests accordingly, to check if the difference in performance is statistically significant. and if there is a B_i significanlty better than A on that label then we add that label to L_{Bi}. Below we show the pseudocode for this technique. This approach worked quite well, even for smaller validation datasets.

1. For all $documents \in ValidationDataset$ assign the relevant $labels \in L$ predicting with model A
2. For each model B_i
   - For all $documents \in ValidationDataset$ assign the relevant $labels \in L$ predicting with B_i
3. For each label $l \in L$ calculate the true positives tp_{Al}, false positives fp_{Al} and false negatives fn_{Al} for A
4. For each model B_i
   - For each label $\in L$ calculate tp_{Bil}, fp_{Bil} and fn_{Bil}
5. Set $tp_A = \sum tp_{Al}$ and fp_A, fn_A respectively
6. Set the micro-F measure as $mf_A = \frac{2tp_A}{2tp_A+fp_A+fn_A}$
7. For each label $l \in L$
   - For each model B_i
     - substract tp_Al, fp_Al and fn_Al from tp_A and fp_A, fn_A respectively
     - add tp_Bil, fp_Bil and fn_Bil to tp_A and fp_A, fn_A respectively
     - If the new mf_A is better than the previous add l in $candidateList_i$
8. For each label l
   (a) If l belongs to just one $candidateList_i$
   - perform a McNemar test between models A and B_i with significance level 0.95
   - if B_i is significantly better than A add l to L_{Bi}

(b) If l belongs to more than one $candidateList_i$
- perform a McNemar test between models A and each B_i with significance level 0.95 applying a FWER correction with the Bonferoni-Holms step method
- If just one B_i is significantly better than A add l to L_{Bi}
- Else if many B_i's are significantly better than A choose the model B_i that has the highest score in the McNemar test with A [5]

9. Compute L_A as $L_A = L - \sum L_{Bi}$
10. For all $documents \in TestDataset$ assign the relevant $labels \in L_A$ predicting with model A
11. For each model B_i
    - For all $documents \in TestDataset$ assign the relevant $labels \in L_{Bi}$ predicting with model B_i

A final note is that when performing multiple statistical comparisons (that is for more than two models) we need to keep control of the family-wise error rate (FWER) in order for the statistical comparisons to be valid. [18] refers to many techniques of controlling that error. In our case, as the tests we performed were parametrical, we used the Bonferroni-Holms step method.

4 Description of Systems and experiments

This section provides the description of our systems, the training procedure and the experiments. We present all results for the systems in the following section, so whenever speaking about e.g. a model being better than another or about performances, we refer the reader to section 5.

4.1 Description of the experiments

In our experiments we used a subset of the corpus, keeping only the documents belonging to the journals from which the new, unseen data would be taken. Thus we ended up with about 4.3 million documents. For all systems, we extracted a dictionary from the corpus keeping words and bigrams (pairs of words) with more than 6 ocurrences and less than half of the size of the corpus, removing stopwords (e.g. "and", "the", etc) and non-arithmetic symbols. In case of the SVMs' training, each feature was represented by its tf-idf value [6], where tf stands for term frequency and idf, inverse document frequency. In that case we also applied zoning for features belonging in the title and features that were a label (e.g. features such as "humans", "female", etc). In the context of the BioAsq competition we used the last 50 thousand documents for validation and the preceding 1.5 million documents for training.

[5] It is needless at this point to apply again McNemar tests among the B_i models because we are not interested on determining if their differences in performance are significant; we just need to choose one among them as we know they are all doing better than A

[6] apart from the BNS SVMs in which case we used the BNS value

4.2 Systems used in the competition

We used five systems in the competition, opting to name them as Asclepios, Hippocrates, Sisyphus, Galen and Panacea.

The first two systems are identical but trained in different size datasets. We trained L binary SVMs in a one-vs-all approach (one for each label) and a second-level model, the Metalabeler (for predicting an instance's label cardinality). During prediction we slightly changed the Liblinear code to output a score instead of a binary decision for the SVMs. This way, for each instance we obtain a descending ranking of labels, from the ones with the highest scores to the ones with the lowest. Then, by using the Metalabeler we predict a label cardinality c for that instance and thus choose the top c labels from the ranking. Asclepios was trained on the last 950 thousand documents while Hippocrates was trained on the last 1.5 million documents.

The rest of the systems are ensembles implemented just as described in section 3. They all have Hippocrates as a component, which was the best performing system, so from now and forth we will refer to it as the baseline model.

The third system, Sisyphus, consists of an ensemble of two models, the baseline and a model of simple binary SVMs. We initially used vanilla (not tuned) SVMs for the second model but then proceeded in trying also to tune them. Feature scaling with BNS [19] was our first effort, but the trained models performed worse and training required very long times. The reason for the last observation is that if performing scaling or feature selection in a multilabel problem, the features' scaling factors for training will be different for each label. This means that we need to vectorize the training corpus L times, a non-trivial task in our case where L is of the order of 10^4. If using common scaling factors for all labels instead (e.g. by tf-idf as we did) vectorizing needs to be done only once for all labels. Another effort for tuning the SVMs was to experiment with different values for the C parameter (other than the default 1) which did not really yield significant improvements. We then used the idea of [20] to change the weight parameter for positive instances (w1). When training a classifier with very few positive instances we can choose to penalize a false negative (a positive instance being misclassified) more than a false positive (a negative instance being misclassified). We followed this approach unfortunately just before the end of the third batch, but nonetheless it yielded very good results for the binary models.

The fourth model, Galen, is an ensemble of the baseline model and a prior LLDA model and the fifth, Panacea, combines in an ensemble the baseline model (SVMs with score ranking and Metalabeler), the tuned binary SVMs, the prior LLDA model (all trained on the last 1.5×10^6 documents) and a baseline model trained on the whole corpus (about 4.3m documents, except the last 50k documents). Even if from at first glance it seems redundant to combine two identical models, the reason we did this is the following: the corpus contains articles from 1974 to 2014. During this period a lot of things have changed concerning the semantics of some entities, the semantics of some labels and most importantly the distribution of labels to words. This leads to the effect of the first model, trained in 1.5 million documents (papers from 2007-2012) having a better performance

than the second one, trained on the whole corpus (papers between 1974-2012), in terms of the micro-f measure. Nonetheless, the second model learns more labels and is expected to do better in some very rare labels, having more training instances. Driven by this observation we added this model in the ensemble, combining four models in total.

5 Results

5.1 Parameter setup

All SVM-based models were trained with default parameters (C=1, e=0.01). For the LLDA model, we used 10 Markov chains and averaged them, taking a total of 600 samples (one sample every 5 iterations), after a burn-in period of 300 iterations. Alpha and beta parameters were equal for all labels during training with $alpha = 50/L$ and $beta = 0.001$. As noted in [10], the prior LLDA model reduces during prediction to an LDA model with the alpha parameter proportional to the frequency of each label. We set

$$alpha(l) = \frac{50 \times frequency(l)}{totalNumberOfLabels} + \frac{30}{L}$$

and took 200 samples (one every 5 iterations) after a burn-in of 300 iterations, from a single Markov chain. We note here that there was a lot of room for improving the LLDA variant (e.g. average from many Markov Chains or take more samples) but unfortunately we didn't have the time to do so.

Experiments were conducted on a machine with 40 processors and 1Tb of RAM. For the SVM models (apart from those with BNS scaling) the whole training procedure (dictionary extraction, vectorizing and training) for 1.5×10^6 documents, a vocabulary of 1.5×10^6 features and 26281 labels takes around 32 hours. The SVMs trained with BNS scaling, require a lot longer, about 106 hours while the LLDA model needs around 72 hours. Predicting for the 35×10^4 documents of Table 1 needs around 20 minutes for the SVMs and around 3 hours for the BNS SVMs. The prior LLDA model needs a very long time for predicting, around 33 hours. The reason for this is that the time needed for the Gibbs sampling algorithm is proportional to the number of documents and the number of labels, which in our case, are of the order of tens of thousands. In case of the size of the BioAsq datasets ($\sim$ 5000 documents) predicting for the LLDA needed around 4 hours.

5.2 Results

In this section we present the results of our experiments. Table 1 shows the performance of our component models in terms of the micro-F and macro-F measures. We can see that the Metalabeler on 1.5m documents is performing better overally, with the tuned SVMs following. Also, we can easily observe that the Metalabeler on 4.2 million documents is worse compared to the one on

Table 1. Results for the models with which we experimented trained on the last 1.5 million documents of the corpus and tested on 35k documents already annotated documents from the competition batches

Classifier	no. of labels	Micro-F	Macro-F
Vanilla SVMs	26281	0.56192	0.33190
Metalabeler(1.5m documents)	26281	0.59461	0.43622
SVMs with BNS scaling	26281	0.51024	0.27980
tuned SVMs(-w1 parameter)	26281	0.58330	0.37729
Metalabeler(4.2m documents)	26509	0.58508	0.42929
Prior labeled LDA	26281	0.38321	0.29563

Table 2. Results for the component models of our systems trained on the last 1.5 million documents of the corpus and tested on 12.3k documents already annotated documents from the competition batches

Classifier	no. of labels	Micro-F	Macro-F
Metalabeler(1.5m documents)	26281	0.60921	0.44745
tuned SVMs(-w1 parameter)	26281	0.60296	0.40705
Metalabeler(4.2m documents)	26509	0.55350	0.39926
Prior labeled LDA	26281	0.37662	0.40125

1.5m documents, learning though 228 more labels. The prior LLDA model is not performing not as near well as the SVM variants.

Tables 2 and 3 show respectively the performance of the models and the four systems described in section 4.2. Asclepios is ommited as it is identical to Hippocrates. Results are shown for 12.3k documents, having used 35k documents for validation. We can see that the ensemble systems perform better than the baseline (Hippocrates), with Panacea reaching the best performance even though the validation dataset is relatively small.

Table 3. Results for the systems that participated in the BioAsq challenge

Systems	Micro-F	Macro-F
Hippocrates	0.60921	0.44745
Sisyphus	0.61323	0.44816
Galen	0.60949	0.44880
Panacea	0.61368	0.44893

6 Question Answering

Being newcomers in the area of question answering, our modest goal was to replicate work already existing in the literature. We decided to focus on [21], an approach presented in the 2013 BioASQ Workshop for extracting answers to factoid questions. Furthermore, we only focused on phase B of the question answering task, taking the gold (correct) relevant concepts, articles, snippets, and RDF triples from the benchmark datasets as input.

For each factoid question, our system firsts extracts the lexical answer type (LAT). This is achieved by splitting the question into words, extracting the part-of-speech for each word and finally extracting the first consecutive nouns or adjectives in the word list of the question. Then, each of the relevant snippets is split into sentences and each of these sentences are processed with the 2013 Release of MetaMap [22] in order to extract candidate answers.

For each candidate answer c, we calculated five scores similarly to [21]. Let I denote an indicator function, returning 1 if each input is true and 0 otherwise. The first score is *prominence*, which considers the frequency of each candidate answer c within the set of sentences S of the relevant snippets:

$$\text{Prominence}(c) = \frac{\sum_{s \in S} I(c \in s)}{|S|} \tag{1}$$

The second score is a version of prominence that further takes into account the cosine similarity of the question q with each sentence:

$$\text{WeightedProminence}(c) = \frac{\sum_{s \in S} \text{similarity}(q, s) I(c \in s)}{\sum_{s \in S} \text{similarity}(q, s)} \tag{2}$$

The third score, *specificity*, considers the (in)frequency of each candidate answer in the corpus of PubMed abstracts A released by BioASQ:

$$\text{Specificity}(c) = \log \left(\frac{|A|}{\sum_{a \in A} I(c \in a)} \right) / \log(|A|) \tag{3}$$

The fourth and fifth scores consider the semantic type(s) of the candidate answers as detected by MetaMap. In particular they examine whether these types intersect with the semantic types(s) of the questions's LAT (fourth score) and the whole question (fifth score):

$$\text{TypeCoercionLAT}(c) = \begin{cases} 1 & \text{if} \quad \text{SemType}(c) \cap \text{SemType}(\text{LAT}) \neq \emptyset \\ 0 & \textit{otherwise} \end{cases} \tag{4}$$

$$\text{TypeCoercionQuestion}(c) = \begin{cases} 0.5 & \text{if} \quad \text{SemType}(c) \cap \text{SemType}(q) \neq \emptyset \\ 0 & \textit{otherwise} \end{cases} \tag{5}$$

Table 4 presents the results of the above scores as well as their ensembling on the 42 factoid questions out of the 100 questions provided by BioASQ as training set. Results are presented in terms of the three metrics of the BioASQ competition: Strict accuracy (SAcc), which compares the correct answer with the top candidate, lenient accuracy (LAcc), which compares the correct answer with the top 5 candidates and mean reciprocal rank (MRR), which takes into account the position of the correct answer within the ranking of candidates.

Table 4. Results of the different scores and their ensembling

Scoring	SAcc	LAcc	MRR
Prominence (P)	9%	31%	16%
WeightedProminence (WP)	23%	31%	25%
Specificity (S)	4%	23%	11%
P + WP + S	31%	43%	35%
P + WP + S + TypeCoercionLAT (TCLAT)	26%	40%	31%
P + WP + S + TCLAT × 0.5	29%	45%	35%
P + WP + S + TypeCoercionQuestion (TCQ)	24%	45%	33%
P + WP + S + TCQ × 0.5	29%	48%	36%
P + WP + S + TCQ × 0.5 + TCLAT	24%	43%	32%
P + WP + S + TCQ + TCLAT × 0.5	24%	48%	35%

Interestingly, we notice that in terms of SAcc, the best results are obtained by combining the first three non-semantic scorings. In terms of LAcc, the best results are obtained when combining the first three scorings with TCLAT weighted by 0.5 or with TCQ weighted by 1 and TCLAT weighted by 0.5. The best results in terms of MRR are obtained when combining the first three scorings with TCQ weighted by 0.5.

7 Conclusions and future work

While experimenting with different datasets, we noticed a significant change in the performance of models with time. It would be really interesting to study in a systematic way this concept drift along time, as it could yield interesting observations about trends in the literature, changes of meaning of terms and, from a machine learning view, changes in the hidden distribution. Concerning the algorithms we put into practice we think that, despite its poor performance, the LLDA model has a lot to offer in a multilabel classification task as the one we dealt with, and that there is a lot of room for improvements as well. For instance, a possible parallelization or some variant of a faster Gibbs sampling implementation scheme during the prediction phase could improve performance by allowing to draw more samples. Either way, a hybrid approach to exploit both the SVM and the LDA theory could bring significant improvements over the multilabel classification problem.

References

1. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297
2. 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** (June 2008) 1871–1874
3. Tang, L., Rajan, S., Narayanan, V.K.: Large scale multi-label classification via metalabeler. In: WWW '09: Proceedings of the 18th international conference on World wide web, New York, NY, USA, ACM (2009) 211–220
4. Yang, Y.: A study of thresholding strategies for text categorization. In: SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2001) 137–145
5. Fan, R.E., Lin, C.J.: A study on threshold selection for multi-label classification. Technical report, National Taiwan University (2007)
6. Nam, J., Kim, J., Gurevych, I., Fürnkranz, J.: Large-scale multi-label text classification - revisiting neural networks. CoRR **abs/1312.5419** (2013)
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3** (March 2003) 993–1022
8. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences **101**(Suppl. 1) (April 2004) 5228–5235
9. Ramage, D., Hall, D., Nallapati, R., Manning, C.D.: Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. EMNLP '09, Stroudsburg, PA, USA, Association for Computational Linguistics (2009) 248–256
10. Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M.: Statistical topic models for multi-label document classification. Mach. Learn. **88**(1-2) (July 2012) 157–208
11. 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. SIGIR '03, New York, NY, USA, ACM (2003) 96–103
12. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Proceedings of the 1st International Workshop in Multiple Classifier Systems. (2000) 1–15
13. Wolpert, D.H.: Original contribution: Stacked generalization. Neural Netw. **5**(2) (February 1992) 241–259
14. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2) (August 1996) 123–140
15. Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**(2) (July 1990) 197–227
16. Tahir, M.A., Kittler, J., Bouridane, A.: Multilabel classification using heterogeneous ensemble of multi-label classifiers. Pattern Recogn. Lett. **33**(5) (2012) 513–523
17. Jimeno-Yepes, A., Mork, J.G., Demner-Fushman, D., Aronson, A.R.: A one-size-fits-all indexing method does not exist: Automatic selection based on meta-learning. JCSE **6**(2) (2012) 151–160
18. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30
19. Forman, G.: BNS feature scaling: an improved representation over tf-idf for svm text classification. In: Proceedings of the 17th ACM conference on Information and knowledge management. CIKM '08, New York, NY, USA, ACM (2008) 263–270

20. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. **5** (2004) 361–397
21. Weissenborn, D., Tsatsaronis, G., Schroeder, M.: Answering factoid questions in the biomedical domain. In Ngomo, A.C.N., Paliouras, G., eds.: BioASQ@CLEF. Volume 1094 of CEUR Workshop Proceedings., CEUR-WS.org (2013)
22. Aronson, A.R., Lang, F.M.: An overview of metamap: historical perspective and recent advances. JAMIA **17**(3) (2010) 229–236