Scalable Preference Learning from Data Streams

Fabon Dzogang
Department of Computer Science,
University of Bristol,
Bristol, United Kingdom.
fabon.dzogang@bris.ac.uk

Thomas Lansdall-Welfare
Department of Computer Science,
University of Bristol,
Bristol, United Kingdom.
thomas.lansdall-welfare@bris.ac.uk

Saatviga Sudhahar
Department of Computer Science,
University of Bristol,
Bristol, United Kingdom.
saatviga.sudhahar@bris.ac.uk

Nello Cristianini
Department of Computer Science,
University of Bristol,
Bristol, United Kingdom.
nello.cristianini@bris.ac.uk

ABSTRACT

We study the task of learning the preferences of online readers of news, based on their past choices. Previous work has shown that it is possible to model this situation as a competition between articles, where the most appealing articles of the day are those selected by the most users. The appeal of an article can be computed from its textual content, and the evaluation function can be learned from training data. In this paper, we show how this task can benefit from an efficient algorithm, based on hashing representations, which enables it to be deployed on high intensity data streams. We demonstrate the effectiveness of this approach on four real world news streams, compare it with standard approaches, and describe a new online demonstration based on this technology.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*; I.7.4 [**Document and Text Processing**]: Electronic Publishing

General Terms

Algorithms, Experimentation

Keywords

News popularity, News appeal, Learning to rank, Hashing trick, Online learning

1. INTRODUCTION

The simultaneous online activities of millions of users generate a stream of clicks that contains valuable information about their preferences.

We consider the scenario - typical for online newspapers but also for social media sites such as Facebook - where users are presented with a selection of news items to choose from. In order to engage with the content, users need to click on it, thereby revealing their choice, and in the process disclosing some information about their preferences.

We are interested in using these choices in order to predict which news items are more likely to be chosen in the future by a specific class of users, based on information about their past choices.

Any practical algorithm working in this setting needs to not only be very scalable, due to the large amount of information generated in websites with high traffic, but also online: that is it should be capable of constantly adapting its behaviour based on recent data, in this way tracking any potential drifts in users' preferences.

For practical reasons, we also assume that we have only information about which news items in a given news outlet have become "popular", or "trending": in other words, who are the winners in the competition for user attention, on a given day. This data is publicly released by many websites.

From previous work, we know that the probability of a news article becoming popular does depend on its content, and that it is possible to identify which words increase the probability of a news article being clicked [9]. However, while the content of an article influences its appeal, the actual set of popular articles is defined by the ones with the most appeal: in other words, an article needs to be more appealing than its competitors in order to be chosen. The same article could be a winner or a loser, in different days. Previous work also shows that the appeal of an article can be modeled as a linear function of its textual content, with the parameters being estimated from a large number of training pairs, each showing a popular and a non-popular article from the same day and page. In other words, this can be turned into a learning-to-rank task, and solved with machine learning algorithms [11].

The scalability of this approach is very important, so in this study we explore the use of an efficient representation and an efficient learning algorithm for the task described above. We have used a representation of text based on random projections, implemented by means of a hashing function [16]. This representation, that efficiently reduces the articles to a 4096 dimensional vector, is then combined with an algorithm recently introduced: the Clasher [13] to provide classification of data streams. The entire construction requires constant time and constant space to incorporate a new training point or to label a new test point, and therefore is ideally suitable for data streams.

We demonstrate that this approach works on a number of real world data streams (those generated by BBC, New York Times, Seattle Times, and NPR websites) and that its predictive power is comparable with that of more classical methods, such as the Lasso and Ridge Regression, while still retaining its important computational efficiency.

Finally, we describe an online demonstration that showcases this technology, providing news recommendations using the methods reported in this paper.

2. LEARNING TO RANK PREFERENCES

Previous work has shown that the *readers' preferences problem* can not be solved if approached as a simple classification task (i.e. separating popular from non-popular articles) [9] . This is due to the popularity of an article being a relative quantity that is affected by the other articles that are published on the same day. For example, an article that was popular one day can easily be non-popular if it appears in the following days news where more popular articles are published.

To solve this problem, our prior results indicate that one needs to solve a *preference learning task* that consists of classifying ordered pairs of articles [9]. We call this classifier a *ranker*. For the ranking procedure we learn an appeal function that separates pairs of more-appealing (i.e. popular) and less-appealing (i.e. non-popular) articles. Therefore the ranking procedure is based on information about the preference relationship between pairs of data points (x_+, x_-). Where x_+ refers to an example drawn from the set of popular articles for an outlet and x_- is drawn from the set of non-popular articles, we want to learn the relationship between the two items x_+ and x_-, expressed as a binary classification problem on the vector of their difference $x = x_+ - x_-$.

3. EFFICIENT CLASSIFICATION ON DATA STREAMS

We wish to perform our preference learning on data streams of news articles that are continuously providing new articles to learn from and to classify. This situation arises naturally in many settings where one wishes to learn a user's preferences in an efficient manner, such as selecting items to display to social media users. To tackle this challenge, the preference learning classifier used must be online, in that it learns from each new example as it arrives then discards it, while also using limited computational resources, i.e. bounded memory and processing times. This challenging settings is obviously therefore unsuitable for traditional offline classifiers [1, 5] and dimensionality reduction techniques [3] which become intractable at large scale.

Here we build on our previous work to learn efficiently from high dimensional data streams, to learn user preferences within the same setting. Our proposed method, first introduced in [13], is an online classification algorithm that benefits from constant time and memory complexity in the dimensionality of the feature set size by using the *Hashing trick* to perform *Stochastic Gradient* updates on the stream, as summarised in the following sections.

The Hashing trick

Traditionally, a bag-of-words approach is adopted for text classification, describing each term by its Term Frequency-Inverse Document Frequency (TF-IDF) score [14]. However, due to the Zipfian nature of text [12], a significant drawback of this approach on data streams is keeping track of the ever-growing vocabulary of words that have appeared in the stream. For a real world, deployed system that needs to run in a robust manner for long periods, such a drawback can be problematic.

One solution is the *Hashing trick*, an efficient feature compression technique that keeps sufficient information for learning while extracting scalable vector representations from text on the fly. This can be summarised as follows:

Data samples are mapped from a v-dimensional feature space into a low-dimensional random feature space by compressing each feature (terms in this case) to its hash, where each hash index is in the interval $[1..m]$, and $m \ll v$ is the desired size of the compressed representation. In the compressed feature space, terms collide uniformly around each of the m preallocated hashes. It has been proved that under mild conditions this transform roughly preserves inner product evaluations [6] and unlike other competitive compression techniques this trick does not rely on the full input space representation and can therefore be applied very efficiently on data streams at very large scale.

The new embedding is maintained on the stream by updating the relevant statistics about the hashes processed so far. In the compressed features, TF counts and TF-IDF scores for the hashed features are approximated based on the *count-min sketch* [4]. An analysis shows that this compressed embedding approximates the geometry of the full corpus-wise TF and TF-IDF input space. Extensive experiments conducted on real world datasets have shown that a compressed space composed of $m = 2^{12}$ hashes does not hurt the performance of a topic classifier too much [13]. We refer to this representation as the 12 byte hash representation of an article or simply *h12* for short. The reader is invited to refer to [13] for full results and discussion.

Online approximation of the mean

In a classical setting, offline learning algorithms typically process training data in batches, iterating over the data several times until it converges on the solution to its convex optimisation problem. To solve this problem, where we have a continuous stream of data, this poses an intractable problem for offline methods. In a streaming setting, we use an online stochastic gradient descent method known as clashing [13], where labelled data is represented as a set of hashes (using the Hashing trick) or words (using a TF-IDF representation) and used to maintain a labelled partition of the feature space. A classifier that uses clashing in this way is therefore known as a Clasher, it can be summarised as follows. Given a stream of examples $x^1, \ldots, x^n$ drawn from a class c, the mean point satisfies:

$$p^* = \arg\min_p \sum_{i=1}^{n} \frac{1}{2} \|p - x^i\|_2^2. \tag{1}$$

Since the size of a stream is unbounded, in a realistic scenario p^* cannot be computed exactly. The Clasher estimates this quantity online, the contribution of example x to the total cost is $\frac{1}{2}\|p-x\|_2^2$, where the gradient is $\nabla(p) = p-x$. Therefore, after one gradient step a new estimate for the mean is given by: $p - \alpha\nabla(p) = (1-\alpha)p + \alpha x$, where the learning rate $\alpha \in [0,1]$ controls the deviation from the current mean. As shown in the above derivation, at time t the Clasher approximates the mean prototype p_c^{t+1} for each class c with a smooth exponential average of $x^1, \ldots, x^t$:

$$p_c^{t+1} = (1-\alpha)p_c^t + \alpha x^t \tag{2}$$

When receiving data from a data stream, unlabelled data z can be classified at any time by computing the class c^* of their closest prototypes in Euclidean distance:

$$c^* = \arg\max_{p_c} \|p_c - z\|. \tag{3}$$

As shown in Eqn 2, unlike other online classifiers that are trained discriminatively, the Clasher's learning procedure possesses inherent parallel capabilities: computations can be easily distributed over different processors, one for each class.

Previous work has shown that on real world text data, the Clasher obtains competitive F-scores with respect to the most accurate learning algorithms for text classification, using significantly fewer computational resources. In addition, an analysis about the predictive performance of the Clasher shows that it behaves well within the hashed space representation detailed in the previous section. The reader is referred to [13] for full results and discussion.

4. EXPERIMENTS

We have trained rankers on data streams within a global interval of six years between 2008 and 2014 using both TF-IDF representations (words) and scalable hash representations (h12). In this section we assess the accuracy of the Clasher with respect to other competitive text classification methods for the preference learning task. We first describe the data streams used for learning user preferences, then we introduce other text classification methods tailored for preference learning and detail the protocol used for evaluation. Finally, we present the results obtained and report the performance for all evaluated methods.

Data description

Using our modular architecture for news media analysis [7, 8], we gathered data from four news outlet sources (BBC, New York Times, Seattle Times, NPR) between 2nd May 2008 and 14th July 2014. For each outlet, we collect two streams of training data, the Top Stories and the Most Popular[1] news feeds. The Top Stories feeds feature the articles which appeared on the main page of their news outlet, while the Most Popular feeds feature those articles which received the most attention from the public. From these feeds, we denote any article which appears in both the Top Stories and Most Popular feeds of an outlet as popular, while non-popular articles are those featured in a Top Stories feed that do not appear in any Most Popular feed.

Following this, every article was mapped into both its TF-IDF representation following standard preprocessing techniques [10], using the New York Times corpus [15] as a fixed size vocabulary ($v \approx 2 \times 10^5$), and its hash space representation where $m = 2^{12}$. As described in our ranking procedure, we could then form training examples from pairs of popular and non-popular articles by taking the vector of their difference. When forming these training examples, each pair is randomly assigned to either form a positive or negative pair by simply inverting the sign of the difference. This allows us to maintain balanced training data when operating in a streaming setting.

The number of pairs between 2nd May 2008 and 14th July 2014 for each outlet is detailed in Table 1, along with the total number of days we had data available, the number of popular articles and the number of non-popular articles.

Online evaluation of methods

We evaluated the performance of the Clasher for the preference learning task using both the TF-IDF representation, and the hash space representation on each outlet. We compare this with two other classifiers using the full TF-IDF representation, namely online Ridge and online Lasso [2]. Each of the methods can be summarised as follows:

- **Clasher(h12)** is the proposed scalable method that builds a noisy representation of the stream and approximates the mean of each class. It has inherent parallel capabilities and learns in constant time and memory complexity.

- **Clasher(words)** is the proposed method where the compression step is not performed. It learns in linear time and memory complexity.

- **Ridge(words)** seeks an online estimate of the least-squares solution with small l2-norm. It deals well in settings where the data is over represented in feature space. It learns in linear time and memory complexity.

- **Lasso(words)** seeks a sparse online estimate of the least-squares solution (with small l1-norm). It deals well in settings where some features are correlated or uninformative. It learns in linear time and memory complexity.

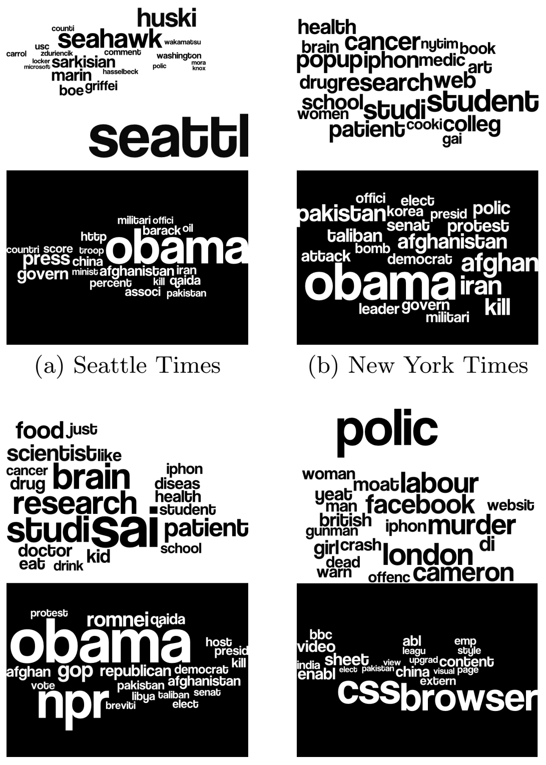
For all methods we use decaying learning rates of the form $\alpha(t) = (\lambda t + 1)^{-1}$, where λ is the regularization parameter. It is set to 1 for the Clasher and to $(\sqrt{p})^{-1}$ for the Ridge and the Lasso.

Since the pairing procedure can cause dependencies in the data published within the same day, the error of a classifier designed as a ranker will be biased within the same day. Indeed, every popular article is paired with every non-popular articles published that day, leading to many redundant pairs. Therefore, to measure the performance of a ranking model that was last updated on day d we compute its error rate based on unseen data published on day $d+1$.

[1]For NPR and New York Times, we use the Most Emailed news feed as the Most Popular stream.

News outlet	Pairs (x_+, x_-)	Published days	Popular set	Non-popular set
BBC	2,300,000	1,027	21,400	185,000
Seattle Times	1,480,000	1,341	7,400	266,000
New York Times	483,000	1,082	11,100	75,800
NPR	463,000	1,109	16,100	53,500

Table 1: Summary of the streams used for learning user preferences, collected between 2nd May 2008 and 14th July 2014.



(a) Seattle Times (b) New York Times (c) NPR (d) BBC

Figure 1: Top 20 (black on white) and least 20 (white on black) stemmed features from the Clasher on words model, showing the user preferences for four outlets.

Results

Table 2 gives the average cumulative error rates obtained by the models on each of the four outlets. The signal for appeal is clearly the strongest for the Seattle Times audience with all methods reaching an average error of 20% or below. The signal appears to be the weakest for the BBC audience where performance varies from 34% to 38% error.

The Clasher on the h12 representation performs remarkably well with respect to the other methods that use the full TF-IDF space. It is particularly distinctive since the size of the h12 space is only 2% of the size of the original input space. Because we have used a fixed size TF-IDF space to describe the articles on words it is possible that the h12 space carries newer informative features for learning user preferences. For example, the word *iPhone* was not known to the public until 2007, and now features as a very appealing word for the BBC audience as illustrated in Figure 1d.

On average, the Lasso gives slightly poorer performance than the other methods. Even though it is possible to improve the Lasso's accuracy by giving less emphasis to the l1-norm constraint, it would be at the cost of poorer interpretability, since more words will be assigned non-zero values.

Figure 2 illustrates the learning curve of the models for the Seattle Times and the BBC audiences. While the error remains quite high for the BBC, as more data is processed the model becomes better at predicting the user preferences. This suggests that more data would be of benefit to improve the performance of the model. On Seattle Times data, all models quickly reach a stable regime where they make few errors while predicting users preferences.

On both the BBC and Seattle Times data, the Clasher(h12) converges as fast as the other methods on words: this can be explained by the low distortion property of the compressed hashed space. Although not described in detail in this paper, the h12 representation data streams are processed much faster than their TF-IDF counterparts, due to the large amount of processing time spent by these methods on inner-product evaluations. In contrast, the processing time for the h12 space is reduced by 98%.

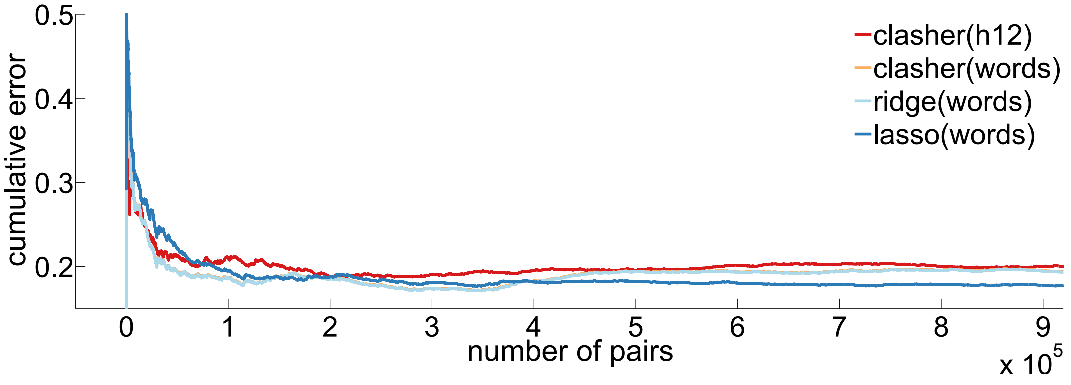
Finally, in Figure 1 we illustrate the word clouds representing the top 20 positive and negative features of the Clasher trained on words for each of the four outlets. The word clouds are extracted by ranking the average coordinates of the vectors $w = p_+ - p_-$ normal to the hyperplanes separating positive from negative pairs at the end of each day. From Figure 1, we can observe that Seattle Times readers have a strong interest in local sports news, with Seahawks, Huskies and Mariners all being present as positive features. We can also see that New York Times and NPR readers share very similar interests, favouring content related to science, technology, health and education research. We can see a general trend, also reported in [9], that readers avoid news content about public affairs, as represented here by terms such as Obama, Afghanistan, Iran and Pakistan appearing prominently in the negative features.

5. NEWS RECOMMENDATION APPLICATION

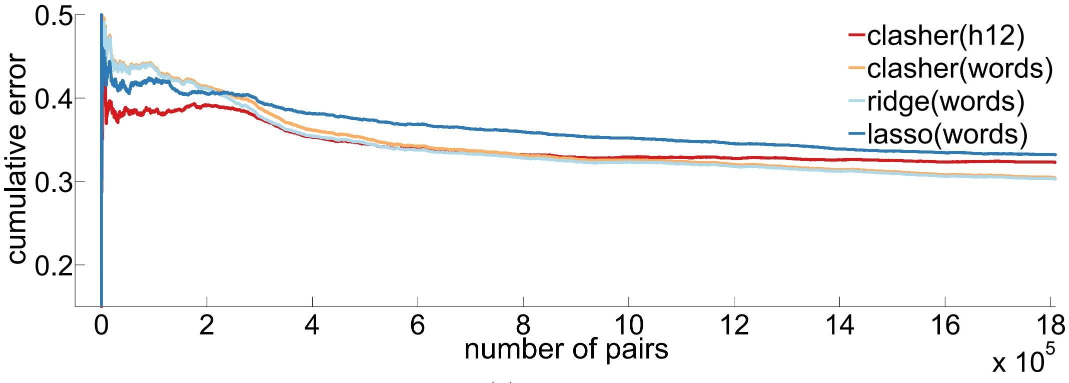
In order to recommend news and demonstrate the feasibility of the Clasher for preference learning, we developed a news recommendation application called Clickable. This application, available at `http://clickable.enm.bris.ac.uk` uses the Clasher(h12) to recommend the most appealing articles each day from a large number of news sources as part of our news analysis infrastructure [7, 8].

News outlet	Clasher(h12)	Clasher(words)	Ridge(words)	Lasso(words)
Seattle Times	0.20	0.19	0.19	0.19
New York Times	0.26	0.25	0.25	0.29
NPR	0.27	0.26	0.26	0.29
BBC	0.34	0.34	0.35	0.38

Table 2: Comparison between the performance of the proposed Clasher(h12) method for learning user preferences with three other competitive methods for text classification on words representation, including the Clasher. Performance is measured by the average cumulative error rate of the ranker computed on the next day's data and reported for a global training interval of six years between 2nd May 2008 and 14th July 2014.



(a) Seattle Times

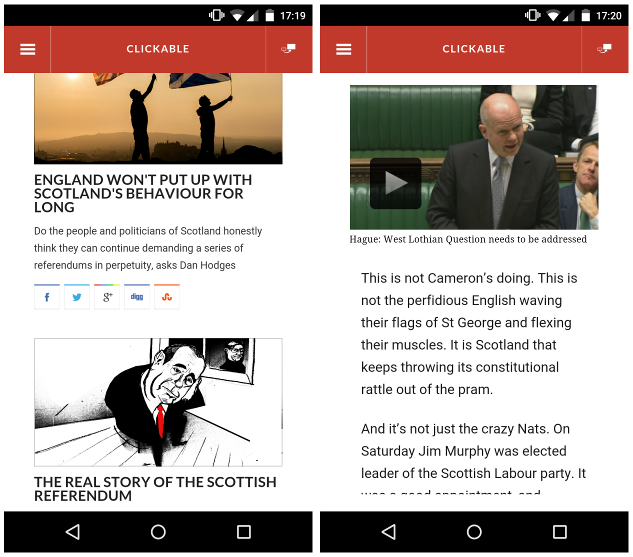


(b) BBC

Figure 2: Learning curves for the four methods evaluated over a six year interval between 2nd May 2008 and 14th July 2014. The learning curves show the cumulative error rate computed on the next days data.

Our aim is that based upon the users' interaction with the application, we will be able to train a user preference model specific to our own users, as we capture a growing number of interactions. We have bootstrapped the model for user preferences with the BBC model, since we believe this model is closest to our own users in the United Kingdom. Over time, the model will learn what makes our users click on specific articles that we recommend, and diverge from the BBC preference model.

While our application does not currently experience such large numbers of users that we would require such efficient methods for assessing the appeal of the daily news, it does allow the application to be very scalable, demonstrating how one could use this approach in a large scale application with millions of users, where one may wish to learn a new model for each user. In such instances, the advantages of using constant processing time and memory becomes crucial to the tractability of the approach.



ENGLAND WON'T PUT UP WITH SCOTLAND'S BEHAVIOUR FOR LONG

Do the people and politicians of Scotland honestly think they can continue demanding a series of referendums in perpetuity, asks Dan Hodges

THE REAL STORY OF THE SCOTTISH REFERENDUM

Hague: West Lothian Question needs to be addressed

This is not Cameron's doing. This is not the perfidious English waving their flags of St George and flexing their muscles. It is Scotland that keeps throwing its constitutional rattle out of the pram.

And it's not just the crazy Nats. On Saturday Jim Murphy was elected leader of the Scottish Labour party. It

Figure 3: Screenshots of the Clickable application on a Nexus 5 Android phone.

6. CONCLUSIONS

In this paper, we study the problem of learning user preferences from a stream of data, focussed in the news media domain. We combine two previous works, one on learning user preferences, the other on efficient classification for data streams, and performed experiments showing that the proposed combination of methods works well for our given task.

Further to this, we present an online application for news recommendation (Clickable: `http://clickable.enm.bris.ac.uk` based upon the evaluated models, demonstrating the feasibility of such an approach and outlining how this can be generalised easily to the case of multiple sets of users, as would be found in a real world news aggregator, news portal or social media site. It efficiently discovers which words make a user-class click on articles, and can track changes, due to its online nature.

7. ACKNOWLEDGMENTS

We want to thank Elena Hensinger and Ricardo Nanculef whose earlier publications and discussions have indirectly inspired the present work. The authors are supported by the EU projects Complacs and ThinkBIG.

8. REFERENCES

[1] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 1. Springer New York, 2006.

[2] Léon Bottou. Stochastic Gradient Descent Tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[3] Christopher JC Burges. *Dimension Reduction*. Now Publishers Inc, 2010.

[4] Graham Cormode and S Muthukrishnan. An Improved Data Stream Summary: The Count-Min Sketch and its Applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[6] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A Sparse Johnson-Lindenstrauss Transform. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, pages 341–350. ACM, 2010.

[7] I. Flaounas, O. Ali, M. Turchi, T. Snowsill, F. Nicart, T. De Bie, and N. Cristianini. NOAM: News Outlets Analysis and Monitoring System. In *SIGMOD 2011*, pages 1275–1278. ACM, 2011.

[8] Ilias Flaounas, Thomas Lansdall-Welfare, Panagiota Antonakaki, and Nello Cristianini. The Anatomy of a Modular System for Media Content Analysis. *CoRR*, abs/1402.6208, 2014.

[9] Elena Hensinger, Ilias Flaounas, and Nello Cristianini. Modelling and Predicting News Popularity. *Pattern Analysis and Applications*, 16(4):623–635, 2013.

[10] Thorsten Joachims. *Learning to Classify Text using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, 2002.

[11] Thorsten Joachims. Optimizing Search Engines using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002.

[12] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[13] Ricardo Ñanculef, Ilias Flaounas, and Nello Cristianini. Efficient Classification of Multi-labelled Text Streams by Clashing. *Expert Systems with Applications*, 2014.

[14] Stephen Robertson. Understanding Inverse Document Frequency: on Theoretical Arguments for IDF. *Journal of documentation*, 60(5):503–520, 2004.

[15] Evan Sandhaus. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12), 2008.

[16] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature Hashing for Large Scale Multitask Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009.