A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News

Kelwin Fernandes[1], Pedro Vinagre[2], and Paulo Cortez[2]

[1] INESC TEC Porto/Universidade do Porto, Portugal
[2] ALGORITMI Research Centre, Universidade do Minho, Portugal

Abstract. Due to the Web expansion, the prediction of online news popularity is becoming a trendy research topic. In this paper, we propose a novel and proactive Intelligent Decision Support System (IDSS) that analyzes articles prior to their publication. Using a broad set of extracted features (e.g., keywords, digital media content, earlier popularity of news referenced in the article) the IDSS first predicts if an article will become popular. Then, it optimizes a subset of the articles features that can more easily be changed by authors, searching for an enhancement of the predicted popularity probability. Using a large and recently collected dataset, with 39,000 articles from the Mashable website, we performed a robust rolling windows evaluation of five state of the art models. The best result was provided by a Random Forest with a discrimination power of 73%. Moreover, several stochastic hill climbing local searches were explored. When optimizing 1000 articles, the best optimization method obtained a mean gain improvement of 15 percentage points in terms of the estimated popularity probability. These results attest the proposed IDSS as a valuable tool for online news authors.

Keywords: Popularity Prediction, Online News, Text Mining, Classification, Stochastic Local Search

1 Introduction

Decision Support Systems (DSS) were proposed in the mid-1960s and involve the use of Information Technology to support decision-making. Due to advances in this field (e.g., Data Mining, Metaheuristics), there has been a growing interest in the development of Intelligent DSS (IDSS), which adopt Artificial Intelligence techniques to decision support [2]. The concept of Adaptive Business Intelligence (ABI) is a particular IDSS that was proposed in 2006 [10]. ABI systems combine prediction and optimization, which are often treated separately by IDSS, in order to support decisions more efficiently. The goal is to first use data-driven models for predicting what is more likely to happen in the future, and then use modern optimization methods to search for the best possible solution given what can be currently known and predicted.

Within the expansion of the Internet and Web 2.0, there has also been a growing interest in online news, which allow an easy and fast spread of information

around the globe. Thus, predicting the popularity of online news is becoming a recent research trend (e.g., [1,3,8,13,15]). Popularity is often measured by considering the number of interactions in the Web and social networks (e.g., number of shares, likes and comments). Predicting such popularity is valuable for authors, content providers, advertisers and even activists/politicians (e.g., to understand or influence public opinion) [3]. According to Tatar et al. [16], there are two main popularity prediction approaches: those that use features only known after publication and those that do not use such features. The first approach is more common (e.g., [1,8,9,13,15]). Since the prediction task is easier, higher prediction accuracies are often achieved. The latter approach is more scarce and, while a lower prediction performance might be expected, the predictions are more useful, allowing (as performed in this work) to improve content prior to publication.

Using the second approach, Petrovic et al. [12] predicted the number of retweets using features related with the tweet content (e.g., number of hashtags, mentions, URLs, length, words) and social features related to the author (e.g., number of followers, friends, is the user verified). A total of 21 million tweets were retrieved during October 2010. Using a binary task to discriminate retweeted from not retweeted posts, a top F-1 score of 47% was achieved when both tweet content and social features were used. Similarly, Bandari et al. [3] focused on four types of features (news source, category of the article, subjectivity language used and names mentioned in the article) to predict the number of tweets that mention an article. The dataset was retrieved from Feedzilla and related with one week of data. Four classification methods were tested to predict three popularity classes (1 to 20 tweets, 20 to 100 tweets, more than 100; articles with no tweets were discarded) and results ranged from 77% to 84% accuracy, for Naïve Bayes and Bagging, respectively. Finally, Hensinger et al. [7] tested two prediction binary classification tasks: popular/unpopular and appealing/non appealing, when compared with other articles published in the same day. The data was related with ten English news outlets related with one year. Using text features (e.g., bag of words of the title and description, keywords) and other characteristics (e.g., date of publishing), combined with a Support Vector Machine (SVM), the authors obtained better results for the appealing task when compared with popular/unpopular task, achieving results ranging from 62% to 86% of accuracy for the former, and 51% to 62% for the latter.

In this paper, we propose a novel proactive IDSS that analyzes online news *prior* to their publication. Assuming an ABI approach, the popularity of a candidate article is first estimated using a prediction module and then an optimization module suggests changes in the article content and structure, in order to maximize its expected popularity. Within our knowledge, there are no previous works that have addressed such proactive ABI approach, combining prediction and optimization for improving the news content. The prediction module uses a large list of inputs that includes purely new features (when compared with the literature [3,7,12]): digital media content (e.g., images, video); earlier popularity of news referenced in the article; average number of shares of keywords prior to publication; and natural language features (e.g., title polarity, Latent Dirich-

let Allocation topics). We adopt the common binary (popular/unpopular) task and test five state of the art methods (e.g., Random Forest, Adaptive Boosting, SVM), under a realistic rolling windows. Moreover, we use the trendy Mashable (`mashable.com/`) news content, which was not previously studied when predicting popularity, and collect a recent and large dataset related with the last two years (a much larger time period when compared with the literature). Furthermore, we also optimize news content using a local search method (stochastic hill climbing) that searches for enhancements in a partial set of features that can be more easily changed by the user.

2 Materials and Methods

2.1 Data Acquisition and Preparation

We retrieved the content of all the articles published in the last two years from Mashable, which is one of the largest news websites. All data collection and processing procedures described in this work (including the prediction and optimization modules) were implemented in Python by the authors. The data was collected during a two year period, from January 7 2013 to January 7 2015. We discarded a small portion of special occasion articles that did not follow the general HTML structure, since processing each occasion type would require a specific parser. We also discarded very recent articles (less than 3 weeks), since the number of Mashable shares did not reach convergence for some of these articles (e.g., with less than 4 days) and we also wanted to keep a constant number of articles per test set in our rolling windows assessment strategy (see Section 2.3). After such preprocessing, we ended with a total of 39,000 articles, as shown in Table 1. The collected data was donated to the UCI Machine Learning repository (`http://archive.ics.uci.edu/ml/`).

Table 1: Statistical measures of the Mashable dataset.

		Articles per day			
Number of articles	**Total days**	**Average**	**Standard Deviation**	**Min**	**Max**
39,000	709	55.00	22.65	12	105

We extracted an extensive set (total of 47) features from the HTML code in order to turn this data suitable for learning models, as shown in Table 2. In the table, the attribute types were classified into: number – integer value; ratio – within $[0, 1]$; bool – $\in \{0, 1\}$; and nominal. Column **Type** shows within brackets (#) the number of variables related with the attribute. Similarly to what is executed in [13,15], we performed a logarithmic transformation to scale the unbounded numeric features (e.g., number of words in article), while the nominal attributes were transformed with the common *1-of-C* encoding.

We selected a large list of characteristics that describe different aspects of the article and that were considered possibly relevant to influence the number

of shares. Some of the features are dependent of particularities of the Mashable service: articles often reference other articles published in the same service; and articles have meta-data, such as keywords, data channel type and total number of shares (when considering Facebook, Twitter, Google+, LinkedIn, StumbleUpon and Pinterest). Thus, we extracted the minimum, average and maximum number of shares (known before publication) of all Mashable links cited in the article. Similarly, we rank all article keyword average shares (known before publication), in order to get the worst, average and best keywords. For each of these keywords, we extract the minimum, average and maximum number of shares. The data channel categories are: "lifestyle","bus","entertainment","socmed", "tech","viral" and "world".

We also extracted several natural language processing features. The Latent Dirichlet Allocation (LDA) [4] algorithm was applied to all Mashable texts (known before publication) in order to first identify the five top relevant topics and then measure the closeness of current article to such topics. To compute the subjectivity and polarity sentiment analysis, we adopted the Pattern web mining module (`http://www.clips.ua.ac.be/pattern`) [5], allowing the computation of sentiment polarity and subjectivity scores.

Table 2: List of attributes by category.

Feature	Type (#)
Words	
Number of words in the title	number (1)
Number of words in the article	number (1)
Average word length	number (1)
Rate of non-stop words	ratio (1)
Rate of unique words	ratio (1)
Rate of unique non-stop words	ratio (1)
Links	
Number of links	number (1)
Number of Mashable article links	number (1)
Minimum, average and maximum number of shares of Mashable links	number (3)
Digital Media	
Number of images	number (1)
Number of videos	number (1)
Time	
Day of the week	nominal (1)
Published on a weekend?	bool (1)

Feature	Type (#)
Keywords	
Number of keywords	number (1)
Worst keyword (min./avg./max. shares)	number (3)
Average keyword (min./avg./max. shares)	number (3)
Best keyword (min./avg./max. shares)	number (3)
Article category (Mashable data channel)	nominal (1)
Natural Language Processing	
Closeness to top 5 LDA topics	ratio (5)
Title subjectivity	ratio (1)
Article text subjectivity score and its absolute difference to 0.5	ratio (2)
Title sentiment polarity	ratio (1)
Rate of positive and negative words	ratio (2)
Pos. words rate among non-neutral words	ratio (1)
Neg. words rate among non-neutral words	ratio (1)
Polarity of positive words (min./avg./max.)	ratio (3)
Polarity of negative words (min./avg./max.)	ratio (3)
Article text polarity score and its absolute difference to 0.5	ratio (2)

Target	Type (#)
Number of article Mashable shares	number (1)

2.2 Intelligent Decision Support System

Following the ABI concept, the proposed IDSS contains three main modules (Figure 1): data extraction and processing, prediction and optimization. The first module executes the steps described in Section 2.1 and it is responsible for collecting the online articles and computing their respective features. The

prediction module first receives the processed data and splits it into training, validation and test sets (data separation). Then, it tunes and fits the classification models (model training and selection). Next, the best classification model is stored and used to provide article success predictions (popularity estimation). Finally, the optimization module searches for better combinations of a subset of the current article content characteristics. During this search, there is an heavy use of the classification model (the oracle). Also, some of the new searched feature combinations may require a recomputing of the respective features (e.g., average keyword minimum number of shares). In the figure, such dependency is represented by the arrow between the feature extraction and optimization. Once the optimization is finished, a list of article change suggestions is provided to the user, allowing her/him to make a decision.

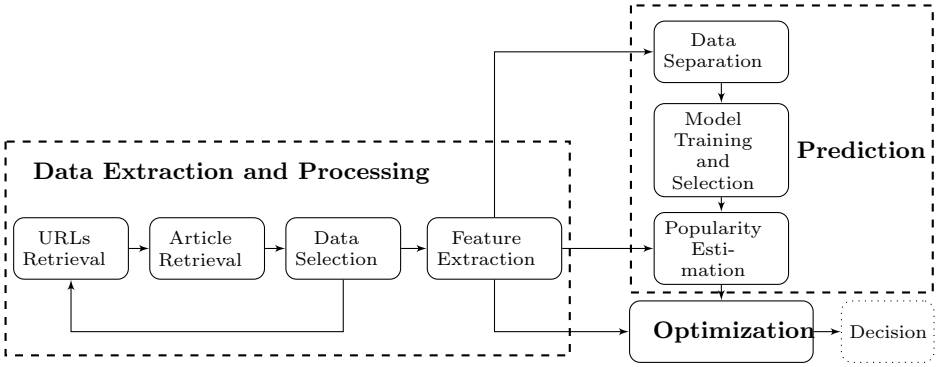

Fig. 1: Flow diagram describing the IDSS behavior.

2.3 Prediction Module

We adopted the Scikit learn [11] library for fitting the prediction models. Similarly to what is executed in [12,3,7], we assume a binary classification task, where an article is considered "popular" if the number of shares is higher than a fixed decision threshold (D_1), else it is considered "unpopular".

In this paper, we tested five classification models: Random Forest (RF); Adaptive Boosting (AdaBoost); SVM with a Radial Basis Function (RBF) kernel; K-Nearest Neighbors (KNN) and Naïve Bayes (NB). A grid search was used to search for the best hyperparameters of: RF and AdaBoost (number of trees); SVM (C trade-off parameter); and KNN (number of neighbors). During this grid search, the training data was internally split into training (70%) and validation sets (30%) by using a random holdout split. Once the best hyperparameter is selected, then the model is fit to all training data.

The receiver operating characteristic (ROC) curve shows the performance of a two class classifier across the range of possible threshold ($D_2 \in [0, 1]$) values, plotting one minus the specificity (x-axis) versus the sensitivity (y-axis) [6]. In this work, the classification methods assume a probabilistic modeling, where a class is considered positive if its predicted probability is $p > D_2$. We computed several classification metrics: Accuracy, Precision, Recall, F1 score (all using a fixed $D_2 = 0.5$); and the Area Under the ROC (AUC, which considers all D_2 values). The AUC metric is the most relevant metric, since it measures the classifier's discrimination power and it is independent of the selected D_2 value [6]. The ideal method should present an AUC of 1.0, while an AUC of 0.5 denotes a random classifier. For achieving a robust evaluation, we adopt a rolling windows analysis [14]. Under this evaluation, a training window of W consecutive samples is used to fit the model and then L predictions are performed. Next, the training window is updated by replacing the L oldest samples with L more recent ones, in order to fit a new model and perform a new set of L predictions, and so on.

2.4 Optimization

Local search optimizes a goal by searching within the neighborhood of an initial solution. This type of search suits our IDSS optimization module, since it receives an article (the initial solution) and then tries to increase its predicted popularity probability by searching for possible article changes (within the neighborhood of the initial solution). An example of a simple local search method is the hill climbing, which iteratively searches within the neighborhood of the current solution and updates such solution when a better one is found, until a local optimum is reached or the method is stopped. In this paper, we used a stochastic hill climbing [10], which works as the pure hill climbing except that worst solutions can be selected with a probability of P. We tested several values of P, ranging from $P = 0$ (hill climbing) to $P = 1$ (Monte-Carlo random search).

For evaluating the quality of the solutions, the local search maximizes the probability for the "popular" class, as provided by the best classification model. Moreover, the search is only performed over a subset of features that are more suitable to be changed by the author (adaptation of content or change in day of publication), as detailed in Table 3. In each iteration, the neighborhood search space assumes small perturbations (increase or decrease) in the feature original values. For instance, if the current number of words in the title is $n = 5$, then a search is executed for a shorter ($n' = 4$) or longer ($n' = 6$) title. Since the day of the week was represented as a nominal variable, a random selection for a different day is assumed in the perturbation. Similarly, given that the set of keywords (K) is not numeric, a different perturbation strategy is proposed. For a particular article, we compute a list of suggested keywords K' that includes words that appear more than once in the text and that were used as keywords in previous articles. To keep the problem computationally tractable, we only considered the best five keywords in terms of their previous average shares. Then, we generate perturbations by adding one of the suggested keywords or by removing one of the original keywords. The average performance when optimizing N articles (i.e.,

N local searches), is evaluated using the Mean Gain (MG) and Conversion Rate (CR):

$$MG = \frac{1}{N}\sum_{i=1}^{N}(Q'_i - Q_i)$$
$$CR = \overline{U'}/U \quad (1)$$

where Q_i denotes the quality (estimated popularity probability) for the original article (i), Q'_i is the quality obtained using the local search, U is the number of unpopular articles (estimated probabilitity $\leq D_2$, for all N original articles) and $\overline{U'}$ is the number of converted articles (original estimated probability was $\leq D_2$ but after optimization changed to $> D_2$).

Table 3: Optimizable Features.

Feature	**Perturbations**
Number of words in the title (n)	$n' \in \{n-1, n+1\}, n \geq 0 \wedge n' \neq n$
Number of words in the content (n)	$n' \in \{n-1, n+1\}, n \geq 0 \wedge n' \neq n$
Number of images (n)	$n' \in \{n-1, n+1\}, n \geq 0 \wedge n' \neq n$
Number of videos (n)	$n' \in \{n-1, n+1\}, n \geq 0 \wedge n' \neq n$
Day of week (w)	$w' \in [0..7), w' \neq w$
Keywords (K)	$k' \in \{K \cup i\} \cup \{K - j\},\ i\ \in K' \wedge j \in K$

3 Experiments and Results

3.1 Prediction

For the prediction experiments, we adopted the rolling windows scheme with a training window size of $W = 10,000$ and performing $L = 1,000$ predictions at each iteration. Under this setup, each classification model is trained 29 times (iterations), producing 29 prediction sets (each of size L). For defining a popular class, we used a fixed value of $D_1 = 1,400$ shares, which resulted in a balanced "popular"/"unpopular" class distribution in the first training set (first $10,000$ articles). The selected grid search ranges for the hyperparameters were: RF and AdaBoost – number of trees $\in \{10, 20, 50, 100, 200, 400\}$; SVM – $C \in \{2^0, 2^1, ..., 2^6\}$; and KNN – number of neighbors $\in \{1, 3, 5, 10, 20\}$.

Table 4 shows the obtained classification metrics, as computed over the union of all 29 test sets. In the table, the models were ranked according to their performance in terms of the AUC metric. The left of Figure 2 plots the ROC curves of the best (RF), worst (NB) and baseline (diagonal line, corresponds to random predictions) models. The plot confirms the RF superiority over the NB model for all D_2 thresholds, including more sensitive (x-axis values near zero, $D_2 >> 0.5$) or specific (x-axis near one, $D_2 << 0.5$) trade-offs. For the best model (RF), the right panel of Figure 2 shows the evolution of the AUC metric over the rolling windows iterations, revealing an interesting steady predictive

performance over time. The best obtained result (AUC=0.73) is 23 percentage points higher than the random classifier. While not perfect, an interesting discrimination level, higher than 70%, was achieved.

Table 4: Comparison of models for the rolling window evaluation (best values in **bold**).

Model	Accuracy	Precision	Recall	F1	AUC
Random Forest (RF)	**0.67**	0.67	**0.71**	**0.69**	**0.73**
Adaptive Boosting (AdaBoost)	0.66	0.68	0.67	0.67	0.72
Support Vector Machine (SVM)	0.66	0.67	0.68	0.68	0.71
K-Nearest Neighbors (KNN)	0.62	0.66	0.55	0.60	0.67
Naïve Bayes (NB)	0.62	**0.68**	0.49	0.57	0.65

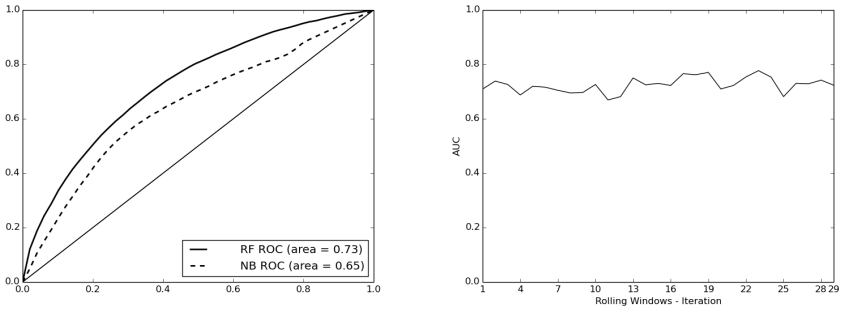

Fig. 2: ROC curves (left) and AUC metric distribution over time for RF (right).

Table 5 shows the relative importance (column **Rank** shows ratio values, # denotes the ranking of the feature), as measured by the RF algorithm when trained with all data (39,000 articles). Due to space limitations, the table shows the best 15 features and also the features that are used by the optimization module. The keyword related features have a stronger importance, followed by LDA based features and shares of Mashable links. In particular, the features that are optimized in the next section (**with keywords** subset) have a strong importance (33%) in the RF model.

3.2 Optimization

For the optimization experiments, we used the best classification model (RF), as trained during the last iteration of the rolling windows scheme. Then, we selected all articles from the last test set ($N = 1,000$) to evaluate the local search methods. We tested six stochastic hill climbing probabilities ($P \in$

Table 5: Ranking of features according to their importance in the RF model.

Feature	**Rank (#)**	**Feature**	**Rank (#)**
Avg. keyword (avg. shares)	0.0456 (1)	Closeness to top 1 LDA topic	0.0287 (11)
Avg. keyword (max. shares)	0.0389 (2)	Rate of unique non-stop words	0.0274 (12)
Closeness to top 3 LDA topic	0.0323 (3)	Article text subjectivity	0.0271 (13)
Article category (Mashable data channel)	0.0304 (4)	Rate of unique tokens words	0.0271 (14)
Min. shares of Mashable links	0.0297 (5)	Average token length	0.0271 (15)
Best keyword (avg. shares)	0.0294 (6)	Number of words	0.0263 (16)
Avg. shares of Mashable links	0.0294 (7)	Day of the week	0.0260 (18)
Closeness to top 2 LDA topic	0.0293 (8)	Number of words in the title	0.0161 (31)
Worst keyword (avg. shares)	0.0292 (9)	Number of images	0.0142 (34)
Closeness to top 5 LDA topic	0.0288 (10)	Number of videos	0.0082 (44)

$\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$). We also tested two feature optimization subsets related with Table 3: using all features except the keywords (*without keywords*) and using all features (*with keywords*). Each local search is stopped after 100 iterations. During the search, we store the best results associated with the iterations $I \in \{0, 1, 2, 4, 8, 10, 20, 40, 60, 80, 100\}$.

Figure 3 shows the final optimization performance (after 100 iterations) for variations of the stochastic probability parameter P and when considering the two feature perturbation subsets. The convergence of the local search (for different values of P) is also shown in Figure 3. The extreme values of P (0 – pure hill climbing; 1 – random search) produce lower performances when compared with their neighbor values. In particular, Figure 4 shows that the pure hill climbing is too greedy, performing a fast initial convergence that quickly gets flat. When using the *without keywords* subset, the best value of P is 0.2 for MG and 0.4 for CR metric. For the *with keywords* subset, the best value of P is 0.8 for both optimization metrics. Furthermore, the inclusion of keywords-related suggestions produces a substantial impact in the optimization, increasing the performance in both metrics. For instance, the MG metric increases from 0.05 to 0.16 in the best case ($P = 0.8$). Moreover, Figure 3 shows that the *without keywords* subset optimization is an easier task when compared with the *with keywords* search. As argued by Zhang and Dimitroff [17], metadata can play an important role on webpage visibility and this might explain the importance of the keywords in terms of its influence when predicting (Table 5) and when optimizing popularity (Figure 3).

For demonstration purposes, Figure 5 shows an example of the interface of the implemented IDSS prototype. A more recent article (from January 16 2015) was selected for this demonstration. The IDSS, in this case using the *without keywords* subset, estimated an increase in the popularity probability of 13 percentage points if several changes are executed, such as decreasing the number of title words from 11 to 10. In another example (not shown in the figure), using the *with keywords* subset, the IDSS advised a change from the keywords $K \in \{$"television", "showtime", "uncategorized", "entertainment", "film", "homeland", "recaps"$\}$ to the set $K' \in \{$"film","relationship","family", and "night"$\}$ for an article about the end of the "Homeland" TV show.

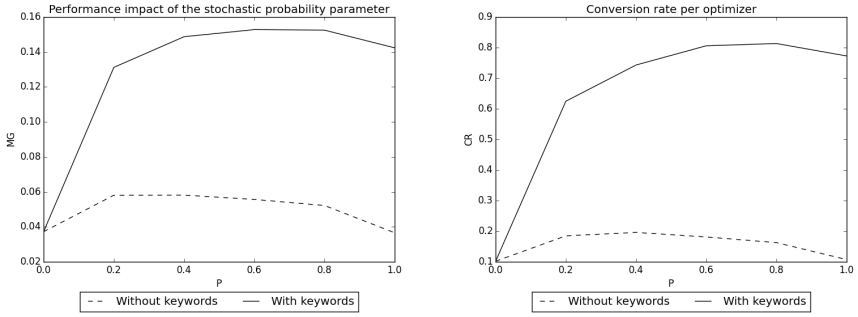

Fig. 3: Stochastic probability (P) impact on the Mean Gain (left) and in the Conversion Rate (right).

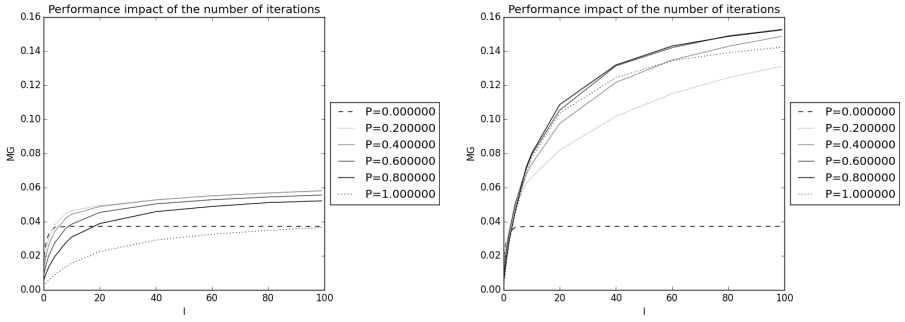

Fig. 4: Convergence of the local search under the *without keywords* (left) and *with keywords* (right) feature subsets (y-axis denotes the Mean Gain and x-axis the number of iterations).

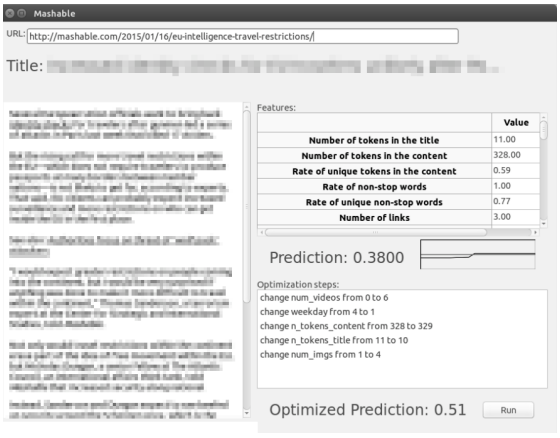

Fig. 5: Example of the interface of the IDSS prototype.

4 Conclusions

With the expansion of the Web, there is a growing interest in predicting online news popularity. In this work, we propose an Intelligent Decision Support System (IDSS) that first extracts a broad set of features that are known prior to an article publication, in order to predict its future popularity, under a binary classification task. Then, it optimizes a subset of the article features (that are more suitable to be changed by the author), in order to enhance its expected popularity.

Using large and recent dataset, with 39,000 articles collected during a 2 year period from the popular Mashable news service, we performed a rolling windows evaluation, testing five state of the art classification models under distinct metrics. Overall, the best result was achieved by a Random Forest (RF), with an overall area under the Receiver Operating Characteristic (ROC) curve of 73%, which corresponds to an acceptable discrimination. We also analyzed the importance of the RF inputs, revealing the keyword based features as one of the most important, followed by Natural Language Processing features and previous shares of Mashable links. Using the best prediction model as an oracle, we explored several stochastic hill climbing search variants aiming at the increase in the estimated article probability when changing two subsets of the article features (e.g., number of words in title). When optimizing 1,000 articles (from the last rolling windows test set), we achieved 15 percentage points in terms of the mean gain for the best local search setup. Considering the obtained results, we believe that the proposed IDSS is quite valuable for Mashable authors.

In future work, we intend to explore more advanced features related to content, such as trends analysis. Also, we plan to perform tracking of articles over time, allowing the usage of more sophisticated forecasting approaches.

Acknowledgements

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope UID/CEC/00319/2013. The authors would like to thank Pedro Sernadela for his contributions in previous work.

References

1. Mohamed Ahmed, Stella Spagna, Felipe Huici, and Saverio Niccolini. A peek into the future: predicting the evolution of popularity in user generated content. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 607–616. ACM, 2013.
2. David Arnott and Graham Pervan. Eight key issues for the decision support systems discipline. *Decision Support Systems*, 44(3):657–672, 2008.
3. Roja Bandari, Sitaram Asur, and Bernardo A Huberman. The pulse of news in social media: Forecasting popularity. In *ICWSM*, 2012.
4. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

5. Tom De Smedt, Lucas Nijs, and Walter Daelemans. Creative web services with pattern. In *Proceedings of the Fifth International Conference on Computational Creativity*, 2014.
6. Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
7. Elena Hensinger, Ilias Flaounas, and Nello Cristianini. Modelling and predicting news popularity. *Pattern Analysis and Applications*, 16(4):623–635, 2013.
8. Andreas Kaltenbrunner, Vicenc Gomez, and Vicente Lopez. Description and prediction of slashdot activity. In *Web Conference, 2007. LA-WEB 2007. Latin American*, pages 57–66. IEEE, 2007.
9. Jong Gun Lee, Sue Moon, and Kavé Salamatian. Modeling and predicting the popularity of online contents with cox proportional hazard regression model. *Neurocomputing*, 76(1):134–145, 2012.
10. Zbigniew Michalewicz, Martin Schmidt, Matthew Michalewicz, and Constantin Chiriac. *Adaptive business intelligence*. Springer, 2006.
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. Sasa Petrovic, Miles Osborne, and Victor Lavrenko. RT to Win! Predicting Message Propagation in Twitter. In *Fifth International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 586–589, 2011.
13. Gabor Szabo and Bernardo A Huberman. Predicting the popularity of online content. *Communications of the ACM*, 53(8):80–88, 2010.
14. Leonard J Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, 16(4):437–450, 2000.
15. Alexandru Tatar, Panayotis Antoniadis, Marcelo Dias De Amorim, and Serge Fdida. From popularity prediction to ranking online news. *Social Network Analysis and Mining*, 4(1):1–12, 2014.
16. Alexandru Tatar, Marcelo Dias de Amorim, Serge Fdida, and Panayotis Antoniadis. A survey on predicting the popularity of web content. *Journal of Internet Services and Applications*, 5(1):1–20, 2014.
17. Jin Zhang and Alexandra Dimitroff. The impact of metadata implementation on webpage visibility in search engine results (part ii). *Information processing & management*, 41(3):691–715, 2005.