LlamaFur: Learning Latent Category Matrix to Find Unexpected Relations in Wikipedia

Paolo Boldi[*]
Università degli Studi di Milano
paolo.boldi@unimi.it

Corrado Monti
Università degli Studi di Milano
corrado.monti@unimi.it

ABSTRACT

Besides finding trends and unveiling typical patterns, modern information retrieval is increasingly interested in the discovery of serendipity and surprising information. In this work[1] we focus on finding *unexpected links* in hyperlinked corpora when documents are assigned to categories. To achieve our goal, we determine a latent category matrix that explains common links using a highly scalable margin-based online learning algorithm, which makes us able to process graphs with 10^8 links in less than 10 minutes. We show that our method provides better accuracy than all existing text-based techniques, with higher efficiency and relying on a much smaller amount of information. It also provides higher precision than standard link prediction, especially at low recall levels; the two methods are in fact shown to be orthogonal to each other and can therefore be fruitfully combined.

CCS Concepts

•**Information systems → Data mining;** •**Computing methodologies → Anomaly detection;** *Knowledge representation and reasoning;* •**Human-centered computing →** *Wikis;*

1. INTRODUCTION

In general, data mining aims at extracting potentially useful information from some (typically unstructured) dataset. Albeit the basic and foremost aim of data mining is discovering frequent patterns, a quite important and somehow dual problem is that of finding unexpected (surprising, unusual...) information; it is striking that this line of investigation did not receive the same amount of attention. There is some research on the determination of surprising information in textual corpora, but there is essentially no work dealing with *unexpected links*. Even if some of the previous proposals exploiting the textual content of documents can be used also for links, a probably more effective way to approach such a scenario is by using link prediction algorithms [16], stipulating that a link that is difficult to predict is unexpected. In this paper, we prove that the availability of some form of categorization of documents can significantly improve the techniques described, leading to algorithms that are extremely efficient, use much less information than text-based methods, and offer better precision/recall trade-offs. Compared to link prediction, our technique also provides higher precision at low recall levels; moreover, the two methods have orthogonal outputs, and therefore their combination improves over both.

Our idea is that if the documents within a linked corpora are tagged with categorical information, one can learn how category/category pairs influence the presence of links, and as a consequence determine which links are unusual (in the sense that they are not "typical"). For example, documents of the category "Actor" often contain links to documents of the category "Movie" (simply because almost all actor pages contain links to the movies they acted in). The fact that George Clooney used to own Max, a 300-pound pig, for 18 years presents itself as a link from an "Actor" page to a page belonging to the category "Pigs"/"Coprophagous animals", which is atypical in the sense above.

Our basic algorithm – henceforth called *LlamaFur*, "Learning LAtent MAtrix to Find Unexpected Relations" – tries to learn a matrix describing the latent relations between categories, and then reconstructs the links that are explainable according to the matrix, and determine those that cannot be justified by the categories alone. It is worth noting that the discovery of unexpected links offers a chance to find unknown information: given a certain document, we can highlight text snippets containing unexpected links. Meaningful text is often characterized, in web documents, by the presence of links that enrich its semantic; this is especially true in the case of Wikipedia, often used as a knowledge base for ontologies. Its link structure has proven to be a powerful resource for many tasks [21]. For this reason, finding unexpected links seems a valuable way to detect meaningful text with information unknown to the reader. We wish to remark that our results could in principle be applied to a plethora of different kinds of objects. The only assumption on the input is a (possibly directed) graph, and a meaningful (possibly overlapping) categorization of its nodes.

[*]Partially supported by the EU-FET grant NADINE (GA 288956).

[1]Due to the limited amout of space available, we omitted many details; the interested reader may wish to read the extended version avilable at `http://arxiv.org/abs/1603.09540`.

WebSci '16, May 22-25, 2016, Hannover, Germany
© 2016 ACM. ISBN 978-1-4503-4208-7/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2908131.2908153

2. RELATED WORK

One of the first papers trying to consider the problem in the context of text mining was [14]. In that work, two supposedly similar web sites are compared (ideally, two web sites of two competitors). The authors first try to find a match between the pages of the two web sites, and then propose a measure of unexpectedness of a term when comparing two otherwise similar pages. All measures are based on term (or document) frequencies; the same approach is improved in [10]. Note that finding unexpected information is crucial in a number of contexts, because of the fundamental role played by serendipity in data mining (see, e.g., [19, 18]).

An alternative way to approach the problem of finding unexpected links is by using *link prediction* [16]: the expectedness of a link e in a network G is the likelihood of the creation of e in $G - \{e\}$. In fact, we will later show that state-of-the-art link prediction algorithms like [1] are very good at evaluating the (un)expectedness of links. Nonetheless, it turns out that the signal obtained from the latent category matrix is even better and partly orthogonal to the one that comes from the graph alone, and combining the two techniques greatly improves the accuracy of both.

Our basic idea – explaining a graph according to some feature its nodes exhibit – is not new. It has been proposed as a graph model by some authors, and have been successfully applied to social networks [13, 12].

This problem has often been casted (e.g. [7]) as a Latent Dirichlet Allocation (LDA) problem, but in almost all cases the number of nodes does not exceed 10^3; in [9] (explicitly tailored for "large graphs"), there are 10^4 nodes and 10^5 links (processed in $45 - 60$ minutes). The algorithm we propose is simpler and requires 9 minutes to run on a graph three orders of magnitude larger (about 10^6 nodes and 10^8 links).

Interpreting links in a network as a result of features of each node has a solid empirical background. Complex behavior, where nodes with certain features tend to connect to another type of nodes, has also proven to be greatly beneficial in analyzing real social networks. Tendencies of such kind are called *mixing patterns* and are often described by a category-category matrix [5]. We will present a model based on a single latent category-category matrix, for the particular scope of mining unexpected links in the graph.

3. LEARNING THE CATEGORY MATRIX

Consider a directed graph $G = (D, L)$ (the "document graph"), whose nodes $d \in D$ represent *documents* and whose arcs $(d, d') \in L$ represent *(hypertextual) links* between documents. Further assume that we have a set C of *categories* and that each document $d \in D$ is assigned a set of categories $C_d \subseteq C$.

Our first goal is to reconstruct the most plausible latent "category matrix" that explains the observed document graph; more precisely, we wish to find a $C \times C$ real-valued matrix W such that

$$\sum_{c \in C_d} \sum_{c' \in C_{d'}} w_{c,c'} \tag{1}$$

is positive iff $(d, d') \in L$. This relation implicitly defines a graph model: given each category set C_d and the latent category matrix W, one can obtain a graph. We are going to assume that in most cases a relation is *unexpected* – that is, surprising to the reader – if it is poorly explained by a plausible category matrix. We will put this assumption under test in the experimental section.

To find such the matrix W described above, we recast our goal in the framework of binary classification; we focused on online algorithms, since they require a constant amount of memory and are therefore more scalable. As we will explain later on, the idea here is that by learning how to separate links from non-links, the classifier must infer W as its internal state. To this aim, let us represent each document d with the indicator vector of C_d, i.e., with the binary vector $\mathbf{d}$ such that $d_c = 1$ *iff* $c \in C_d$. Now, an *example* will be a pair of documents (d, d'), represented as the outer product kernel $\mathbf{d} \otimes \mathbf{d}'$: this is a matrix where the element $[\mathbf{d} \otimes \mathbf{d}']_{c,c'}$ is 1 iff the first document belongs to c and the second to c'. This $(|C| \times |C|)$-matrix[2] can be alternatively thought of as a vector of size $|C|^2$, allowing us to use them as training examples for a perceptron-like classifier, where the label is $y = 1$ iff $(d, d') \in L$ (there is a link), and $y = -1$ otherwise. The learned vector $\mathbf{w}$ will be, if seen as a $|C| \times |C|$ matrix, the desired W appearing in (1). In other words, we are using $|C|^2$ features, in fact a kernel projection of a space of dimension $2|C|$ onto the larger space of size $|C|^2$. Similarly the weight vector to be learned has size $|C|^2$. Positive examples are those that correspond to existing links.

Among the existing perceptron-like online classification frameworks, we chose the well-known Passive-Aggressive classifier, characterized by being extremely fast, simple to implement, and shown by many experiments [17] to perform well on similar datasets. The algorithm is fed with a sequence of pairs of documents $(d_1, d'_1), \ldots, (d_T, d'_T) \in D^2$ (whose construction is described below) and tries to learn a latent matrix W so that $\sum_{c \in C_{d_t}} \sum_{c' \in C_{d'_t}} w_{c,c'}$ is $\geq 1 - \xi$ or $\leq -1 + \xi$, depending on whether (d_t, d'_t) is a positive example (in our case, a link); ξ is an error slack variable that is learnt along with W. The cost function tries to keep as much memory of the previous optimization steps as possible, and at the same time to minimize the error embodied by the slack variable. By merging the Passive-Aggressive solution to this problem with our aforementioned framework, we obtain the algorithm described in Alg. 1.

In our case, W is built through a single-pass online learning process, where we have all positive examples at our disposal (and they are in fact all included in the training sequence), but where negative examples cannot be all included, because they are too many and they would produce overfitting. Since it is critical [11] to have a balanced number of positive and negative examples, we proceed as follows: nodes are enumerated (in arbitrary order), and for each node $d \in D$, all arcs of the form $(d, -) \in E$ are put in the sequence, followed by an equal number of pairs of the form $(d, -) \notin E$ (for those pairs, the destination nodes are chosen uniformly at random).

[2]In practice, we normalize this matrix so that it has unit $L1$-norm. Normalization often gives better results in practice [8]; in this case, documents belonging to few categories provide stronger signals than those that belong to many categories.

Algorithm 1 Passive-Aggressive algorithm to build the latent category matrix.

```
INPUTS: The graph $(D, L)$, a set of categories $C_d \subseteq C$ for each
document $d \in D$, a parameter $K > 0$
OUTPUT: The latent category matrix $W$

1. $W \leftarrow \mathbf{0}$
2. Let $(d_1, d'_1), \ldots, (d_T, d'_T)$ be a sequence of elements of $D \times D$.
3. For $i = 1, \ldots, T$
   (a) $\rho \leftarrow \frac{1}{|C_{d_i}| \cdot |C_{d'_i}|}$
   (b) $\mu \leftarrow \sum_{c \in C_{d_i}} \sum_{c' \in C_{d'_i}} w_{c,c'}$
   (c) if $(d_i, d'_i) \in L$ then   $\delta \leftarrow \rho \cdot \min(K, 1 - \mu\rho)$
       else   $\delta \leftarrow -\rho \cdot \min(K, 1 + \mu\rho)$
   (d) For each $c \in C_{d_i}$, $c' \in C_{d'_i}$:   $w_{c,c'} \leftarrow w_{c,c'} + \delta$
```

4. A NAIVE WAY TO BUILD THE CATEGORY MATRIX

Let us describe an alternative, naive variant of how the latent category matrix W could be obtained. Recall that the purpose is to use equation (1) to compute the expectedness of a link (d, d'). With this purpose, we may use a naive Bayes technique [2], estimating the probability of existence of a link through maximum likelihood and assuming independence between category memberships.

For a given category c, let D_c be the set of documents that have the category c; let also $\mathcal{E}_{c,d}$ represent the event that d belongs to the category c (i.e., $c \in C_d$ or, equivalently, $d \in D_c$). Now for any two categories c and c' one can compute the probability that there is a link between two documents that belong to those categories as $p_{c,c'} = P[(d, d') \in L \mid \mathcal{E}_{c,d} \text{ and } \mathcal{E}_{c',d'}]$. This quantity can be naively estimated as

$$p_{c,c'} = \frac{|\{(D_c \times D_{c'}) \cap L\}|}{|D_c| \cdot |D_{c'}|}.$$

For a pair of documents (d, d'), and under some indipendence assumptions[3] the presence of a link has probability

$$\prod_{c \in C_d} \prod_{c' \in C_{d'}} P\left[(d, d') \in L \middle| \mathcal{E}_{c,d} \text{ and } \mathcal{E}_{c',d'}\right] = \prod_{c \in C_d} \prod_{c' \in C_{d'}} p_{c,c'}.$$

Applying a logarithm and add-one smoothing [20], this is rank-equivalent to $\sum_{c \in C_d} \sum_{c' \in C_{d'}} w_{c,c'}$ where

$$w_{c,c'} = \log \frac{|\{(D_c \times D_{c'}) \cap L\}| + 1}{(|D_c| + 1) \cdot (|D_{c'}| + 1)}$$

This is yet another way to define the matrix W used in the LlamaFur algorithm; the resulting expectedness score for link (d, d') is given by (1), and will be referred to as Naive-LlamaFur.

5. USING THE CATEGORY MATRIX

Let us now call W the category matrix obtained at the end of the learning process (that is, the matrix output by Alg. 1), or equivalently the matrix built using the naive approach of Section 4. This matrix allows one to sort the links $(d, d') \in L$ in increasing order of $\sum_{c \in C_{d_t}} \sum_{c' \in C_{d'_t}} w_{c,c'}$ (i.e., by increasing explainability): the first links are the most unexpected.

[3]More precisely, we are assuming that $\mathcal{E}_{c,d}$ and $\mathcal{E}_{c',d'}$ are independent, whenever $c \neq c'$ or $d \neq d'$, and also that they are independent even under the knowledge that $(d, d') \in L$.

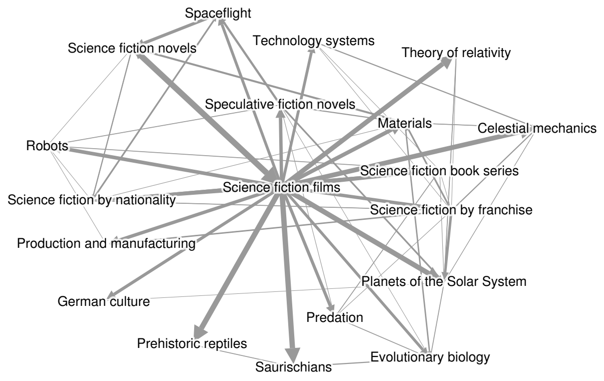

Figure 1: A fragments of the latent category graph induced by LlamaFur matrix W, representing the 18 closest neighbors of category "Science Fiction Films". The width of the arc from c to c' is proportional to $w_{c,c'}$, and arcs with $w_{c,c'} \leq 1$ are not shown.

In particular, in the case of the learning approach of Section 3, one can build a graph $G^* = (D, L^*)$ whose links are the set L^* of pairs (d, d') such that $\sum_{c \in C_{d_t}} \sum_{c' \in C_{d'_t}} w_{c,c'}$ is positive. In a standard binary-classification scenario, G^* would be the graph G that our classifier learned. In particular, the elements of the set $L \setminus L^*$ ($L^* \setminus L$, resp.) are the false negative (false positive, resp.) instances.

But ours is *not* a link-prediction task, and we do not expect in any sense that L and L^* are similar. In particular, we shall certainly observe a phenomenon that we can call *generalization effect*: suppose that it frequently happens that a document assigned to a category c (e.g., an actor) contains links to documents assigned to another category c' (e.g., a movie). This will probably make $w_{c,c'}$ very large, and so we may falsely deduce that *every* document assigned to c (every actor) contains a link to *every* document assigned to c' (every movie). In other words, LlamaFur cannot be used as a reliable link-prediction algorithm.

The generalization effect will, by itself, make L^* much larger than L (i.e., it will produce many false positive instances), but we do not care much about this aspect. Our focus is *not* on trying to reconstruct L, but rather in understanding which elements of L are difficult to explain based on the categories of the involved documents. We say that a link $(d, d') \in L$ is *explainable* iff $(d, d') \in L^*$; the set of explainable links is therefore $L \cap L^*$. On the contrary, the elements of $L \setminus L^*$ are called unexplainable, and these are the links we want to focus on. In Figure 1 we show an example of how the learned W looks like, when considering the Wikipedia dataset (see Section 6).

From this representations we can catch a glimpse of how this method is able to build a model for the graph, capturing meaningful relations between categories.

6. EXPERIMENTS

Given its increasing importance in knowledge representation [21], we used the English edition of Wikipedia as our testbed. In particular, we employed the `enwiki` snapshot[4] of February 3, 2014 to obtain a document graph with 4 514 662 pages, 110 699 703 links and 1 134 715 categories.

Wikipedia categories are organized in a pseudo-tree (where each category is assigned to a "parent" category), but they turn out to be quite noisy; for this reason, we adopted the cleansing procedure described in [4] to reduce the number of categories to 20 000: this number was chosen to offer a good trade-off between the F-measure of Alg. 1 (see below) and the space needed to store W.

To assess how well Alg. 1 is generalizing, we first used a 10-fold cross-validation technique—specifically, we divided in 10 folds the space of node pairs $D \times D$. The average F-Measure on unknown node pairs is 86.9% (precision and recall are 90% and 84%, respectively), showing that the model learnt by LlamaFur is robust and not threatened by overfitting. These results suggest that we can consider the unexplained links as atypical: learning on the whole graph, the ratio $|L \cap L^*|/|L|$ – that is, how many existing links are explained by W – is equal to 86%. Please note that running our algorithm on the whole Wikipedia graph (110 699 703 arcs) with the categorization we chose (20 000 categories) required only 9 minutes on an Intel Xeon CPU with 2.40GHz. Finally, we proceeded to assign our unexpectedness score to each link (see Table 1 and 2 for some annecdotal example).

George W. Bush	Kim's regime argued the secret *[nuclear]* production was necessary for security purposes – citing the presence of United States-owned nuclear weapons in South Korea and the new tensions with the United States under President George W. Bush.
Kim Il-sung	He succeeded his father and founder of the DPRK, Kim Il-sung.

Table 1: Most expected links of Kim Jong-il[4], according to LlamaFur.

Elvis Presley	In a 2011 news story, The Sun reported Kim Jong-il was obsessed with Elvis Presley. His mansion was crammed with his idol's records and his collection of 20,000 Hollywood movies included Presley's titles – along with Rambo and Godzilla. He even copied the King's Vegas-era look of giant shades, jumpsuits and bouffant hairstyle.
Michael Jordan	Kim reportedly enjoyed basketball. Former United States Secretary of State Madeleine Albright ended her summit with Kim by presenting him with a basketball signed by NBA legend Michael Jordan.

Table 2: Most unexpected links of Kim Jong-il[4], according to LlamaFur.

Evaluation methodology. We want to evaluate the effectiveness of LlamaFur in mining unexpected links using a standard approach commonly adopted in Information Retrieval. In our context, a *query* is a document, the possible *results* are the hyperlinks that the document contains, and a result is *relevant* for our problem if it represents an unexpected link. The scenario we have in mind is that of a user wishing to find surprising links in a certain Wikipedia page. What we are trying to assess is how well LlamaFur can identify an unknown set of unexpected links, having full knowledge of graph and categorization of nodes.

In order to compare the results obtained by LlamaFur with the existing state-of-the-art for similar problems, we performed a user study based on the same pooling method adopted for many standard collections such as TREC (trec.nist.gov): we considered a random sample of 237 queries (i.e., Wikipedia documents); for each query we took, among its t possible results (i.e., links), the top-$\lfloor \alpha \cdot t \rfloor$ most unexpected ones according to each system under comparison (see below); all the resulting links were evaluated by human beings. We set $\alpha = 0.1$, and obtained 3 698 links.

The human evaluators were asked to categorize each link into one of four classes ("totally expected", "slightly expected", "slightly unexpected" and "totally unexpected"). They were provided with the first paragraph of the two Wikipedia pages, and a link to the whole article if needed. The resulting dataset of 3 698 evaluated links is available for download[5] and inspection. (To obtain a sufficiently large number of evaluated links in a short amount of time, there was virtually no overlap between the links that the evaluators worked on; however, we manually inspected the dataset to find that the labels produced are quite robust—we invite the reader to do the same.) After the human evaluation, we only considered the queries that have at least one irrelevant ("totally/slightly expected") and one relevant ("totally/slightly unexpected") result according to the evaluation, obtaining a dataset with 117 queries. In this dataset, on average each query has 3.48 relevant results over 20.9 evaluated links. About 58.1% of the links were labelled as "totally expected", only 2.2% were "totally unexpected" and about 8.8% were labelled as "unexpected".

It is evident how unexpected links are very sparse (many pages did not present any unexpected link at all); this motivated us to employ bpref [6], a well-suited information retrieval measure. To compute it, we followed TREC specifications[6]; in a nutshell, bpref computes an index in the range [0, 1] of how many judged relevant documents (in our case, surprising links) are retrieved ahead of a judged non-relevant document (unsurprising links).

Baselines and competitors. In our comparison, LlamaFur is tested in combination and against a number of baselines and competitors. In particular, we considered LlamaFur and its naive variant, Naive-LlamaFur, along with some of the other (un)expectedness measures proposed in the literature.

Albeit there are, at the best of our knowledge, no algorithms specifically devoted to determining unexpected links, we can adapt some techniques used for unexpected documents to our case. All of those methods try to measure the unexpectedness of a document d among a set of retrieved documents R. In our application, we are considering a link (d', d) and taking R to be the set of all documents towards which d' has a hyperlink. (a) *Text-based methods.* In the

[4]This dataset is commonly referred to as `enwiki-20140203-pages-articles` according to Wikipedia naming scheme.

[4]Kim Jong-il was the supreme leader of North Korea from 1994 to 2011.

[5]Evaluations are available at http://git.io/vmChm; all the code and data needed to replicate our findings are available at http://git.io/vmzjm.

[6]http://trec.nist.gov/pubs/trec16/appendices/measures.pdf

literature, all of the measures of unexpectedness are based on the textual content of the document under consideration. Here, we chose to compare our technique with M2 and M4 of [10] (b) *Link-prediction methods*. A completely different, alternative approach to the problem is based on *link prediction*: how likely is it that the link (d', d) is created, if we assume that it is not there? Among the many techniques for link prediction [16], we tested the well-known *Adamic-Adar index* [1] (AA, in the following). (c) *Combinations*. Besides testing all the described techniques in isolation, we tried to combine them linearly (normalizing through their *studentized residual*).

Results. LlamaFur obtains an average bpref of 0.343, much better than the textual baselines (M2 and M4 get 0.179 and 0.293, respectively) and than AA (0.286). The naive version has a bpref of only 0.251, while combining AA with LlamaFur improves over both, obtaining a bpref of 0.350.

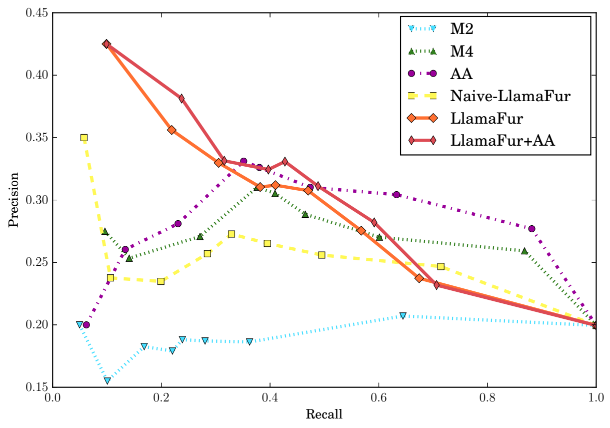

Figure 2: Precision-recall curves.

Some complementary information about the behaviour is provided by the precision-recall graph of Figure 2: first of all, LlamaFur, AA, M4 and their combinations have larger precision than the remaining ones at almost all the recall levels; on the other hand LlamaFur +AA is the best method for recall values up to 50%, and LlamaFur has definitely better precision than AA until 30% of recall.

In fact, M4, AA and LlamaFur seem to be complementary to one another; in some sense, this is not surprising given that they stem from completely different sources of information: one is based on the textual content, another on the pure link graph and the latter on the category data.

7. CONCLUSIONS AND FUTURE WORK

In this work we presented a graph model based on the interplay between categories, able to catch the notion of *expected links* on a graph; we showed that this model can be employed to find unexpected links in hyperlinked document corpora, through the determination of a latent category matrix; the latter is built using a perceptron-like technique. We demonstrated that our method provides better accuracy than most existing text-based techniques, with higher efficiency and relying on a much smaller amount of information. Moreover, we showed that LlamaFur can process graphs with 10^8 links or more without effort. We carried out experiments on the categorized Wikipedia graph – a widely employed source of information for knowledge representation. It would be useful to try our unexpected link mining approach on different datasets.

8. REFERENCES

[1] L.A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2001.

[2] C.M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[3] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[4] P. Boldi and C. Monti. Cleansing wikipedia categories using centrality. In *Proc. 24th Int.Conf. on WWW*, WWW '16 Companion, 2016 (To appear).

[5] Paolo Boldi, Irene Crimaldi, and Corrado Monti. A network model characterized by a latent attribute structure with competition. *Information Sciences*, to appear.

[6] C. Buckley and E.M. Voorhees. Retrieval evaluation with incomplete information. In *Proc. of the 27th ACM SIGIR*, pages 25–32. ACM, 2004.

[7] J. Chang and D.M. Blei. Relational topic models for document networks. In *Int. Conf. on AI and statistics*, pages 81–88, 2009.

[8] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[9] K. Henderson and T. Eliassi-Rad. Applying latent dirichlet allocation to group discovery in large graphs. In *Proc. 2009 ACM Symposium on Applied Computing*, pages 1456–1461. ACM, 2009.

[10] F. Jacquenet and C. Largeron. Discovering unexpected documents in corpora. *Knowledge-Based Systems*, 22(6):421 – 429, 2009.

[11] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[12] M. Kim and J. Leskovec. Multiplicative attribute graph model of real-world networks. *Internet Mathematics*, 8(1-2):113–160, 2012.

[13] S. Lattanzi and D. Sivakumar. Affiliation networks. In *Proc. of ACM STOC '09*, pages 427–434, 2009.

[14] B. Liu, Y. Ma, and Philip S. Yu. Discovering unexpected information from your competitors' web sites. In *Proc. KDD 2001*, pages 144–153. ACM, 2001.

[15] Y. Liu, A. Niculescu-Mizil, and W. Gryc. Topic-link lda: joint models of topic and author community. In *Proc. 26th Annual Int. Conf. on Machine Learning*, pages 665–672. ACM, 2009.

[16] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[17] C. Monti, A. Rozza, G. Zappella, M. Zignani, A. Arvidsson, and E. Colleoni. Modelling political disaffection from twitter data. In *Proc. of the 2nd Int. WISDOM*, page 3. ACM, 2013.

[18] T. Murakami, K. Mori, and R. Orihara. Metrics for evaluating the serendipity of recommendation lists. In *Proc. of the 2007 Conf. on New Frontiers in AI*, JSAI'07, pages 40–46. Springer-Verlag, 2008.

[19] N. Ramakrishnan and A.Y. Grama. Data mining-guest editors' introduction: From serendipity to science. *Computer*, 32(8):34–37, 1999.

[20] Stuart Russell and Peter Norvig. Artificial intelligence: A modern approach. 2010.

[21] F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the WWW*, 6(3):203 – 217, 2008.