PRE-PRINT version of the paper accepted for publication at SIGIR 2016.

A Visual Analytics Approach for What-If Analysis of Information Retrieval Systems

Marco Angelini
University of Rome
Rome, Italy

Nicola Ferro
University of Padua
Padua, Italy

Giuseppe Santucci
University of Rome
Rome, Italy

Gianmaria Silvello
University of Padua
Padua, Italy

ABSTRACT

We present the innovative visual analytics approach of the VATE^2 system, which eases and makes more effective the experimental evaluation process by introducing the what-if analysis. The what-if analysis is aimed at estimating the possible effects of a modification to an IR system to select the most promising fixes before implementing them, thus saving a considerable amount of effort.

VATE^2 builds on an analytical framework which models the behavior of the systems in order to make estimations, and integrates this analytical framework into a visual part which, via proper interaction and animations, receives input and provides feedback to the user.

1. INTRODUCTION

Understanding and interpreting the results produced by experimental evaluation is not a trivial task, which requires a lot of manual effort due to the complex interactions among the components of an *Information Retrieval (IR)* system. Moreover, after such activity, the researcher needs to come back to design and then implement the modifications that the previous analysis suggested as possible solutions to the identified problems. Afterwards, a new experimentation cycle needs to be started to verify whether the introduced modifications actually give the expected improvement. Therefore, the overall process of improving an IR system is extremely time and resource demanding and proceeds through cycles where each new feature needs to be implemented and experimented.

The goal of this paper is to introduce a new phase in this cycle: we call it *what-if analysis* and it falls between the experimental evaluation and the design and implementation of the identified modifications. What-if analysis aims at estimating what the effects of a modification to the IR system under examination could be before actually being implemented.

What-if analysis exploits *Visual Analytics (VA)* techniques to make researchers and developers: (i) interact with and explore the ranked result list produced by an IR system and the achieved performances; (ii) hypothesize possible causes of failure and their fixes; (iii) estimate the possible impact of such fixes through a powerful analytical model of the system behavior. This paper introduces the *Visual Analytics Tool for Experimental Evaluation (VATE^2)* prototype, which provides a proof-of-concept of how *what-if- analysis* works.

The paper is organized as follows: Section 2 summarizes our previous work on VA for performance and failure analysis, which constitutes the starting point for developing the what-if analysis; Section 3 gives an overview of the analytics framework defining the what-if analysis. Section 4 illustrates how the what-if analysis has been realized in the VATE^2 prototype and describes its main features. Finally, Section 5 draws some conclusions.

2. PERFORMANCE AND FAILURE ANALYSIS

In order to quantify the performances of an IR system, we realized *Visual Information Retrieval Tool for Upfront Evaluation (VIRTUE)* [1, 2] which adopts the *Discounted Cumulated Gain (DCG)* family of measures [4] which allow for graded relevance judgments and embed a model of the user behavior while he scrolls down the result list which also gives an account of its overall satisfaction.

We compare the result list produced by an experiment with respect to an *ideal* ranking created starting from the relevant documents in the ground-truth. In addition to what is typically done, we compare the result list with respect to an *optimal* one created with the same documents retrieved by the IR system but with an optimal ranking, i.e. a permutation of the results retrieved by the experiment aimed at maximizing its performances by sorting the retrieved documents in decreasing order of relevance.

Looking at a performance curve, as the DCG curve is, it is not always easy to spot what the critical regions in a ranking are. Indeed, DCG is a not-decreasing monotonic function which increases only when a relevant document is found in the ranking. However, when DCG stays constant, it is not immediately clear to understand whether this is due to a failure of the system which is not retrieving relevant documents while it would still be expected to do so, or whether the system is performing properly since there would be nothing to gain at that rank position.

To overcome this and similar issues, we introduce the *Relative Position (RP)* indicator, which allows us to quantify and explain what happens at each rank position and its paired with a visual counterpart which eases the explo-

ACM ISBN 123-XXX.
DOI: 10.XXX

ration of the performances across the ranking, immediately grasping the most critical areas. RP quantifies the effect of misplacing relevant documents with respect to the ideal case, i.e. it accounts for how far a document is from its ideal position. RP eases the interpretation of the DCG curve since, for example, a constant value of DCG implies a negative value of RP, if this is due to a failure of the system which is not retrieving relevant documents while it would still be expected to do so, or a zero value of RP, if the system is performing properly since there would be nothing to gain at that rank position.

The RP indicator is paired with a visual counterpart that makes it even easier to quickly spot and inspect critical areas of the ranking. A bar is added on the left of the visualization where each rank position is represented with a box and, by using appropriate color coding to distinguish between zero, positive and negative values and shading to represent the intensity, each box represents the values of RP.

For example, in this way, looking at the bar and its colors, the developer can immediately identify not relevant documents which have been ranked in the positions of relevant ones. Then, the visualization allows them to inspect those documents and compare them with the topic at hand in order to make hypotheses about the causes of a failure.

3. WHAT-IF ANALYSIS

Suppose now that a query is about `personal computation devices` while a document talks about `personal computing devices`. If a system suffers from under-stemming, `computation` and `computing` may not be properly stemmed to `comput`. In this case, the previous document may be ranked lower because it matches the query only partially.

By performing failure analysis, the developer hypothesizes that the problem is the stemmer which does not conflate `computing` to `comput`. At the same time, the developer hypothesizes that if s/he fixes the failure, a given relevant document would be ranked higher than in the current system. What the visualization of Figure 2 offers to the developer is: (i) the possibility of dragging and dropping the target document in the estimated position of the rank; (ii) the estimation of which other documents would be affected by the movement of the target document and how the overall ranking would be modified; (iii) the computation of the system performances according to the new ranking. Indeed, if the stemmer is fixed, not only the target document identified by the developer would be affected by this modification but also other documents which, for example, contain the term `computable` and which were not examined by or known to the developer. Therefore, moving a single target document would actually cause the movement and repositioning of a whole set of documents that share features impacted by the same modification which will affect the target document selected by the developer. These complex interactions between documents may generate modifications on the ranking that go well beyond what the developer imagined when moving the single target document and which are definitely hard for her/him to guess. Thus, the contribution of the visualization and analytical engine of Figure 2 is to automatically point out to the developer all these complex interactions and how they affect the overall ranking.

In order to carry out the scenario just envisioned, VATE^2 needs: (i) to understand which documents would be affected by the movement of a target document indicated by the developer; and (ii) to adopt a strategy for simulating what the movement of the documents in the ranked list could be. Both of these items require a quite complex analytical model and computations. The complete description of the engine employed in VATE^2 is described in [3].

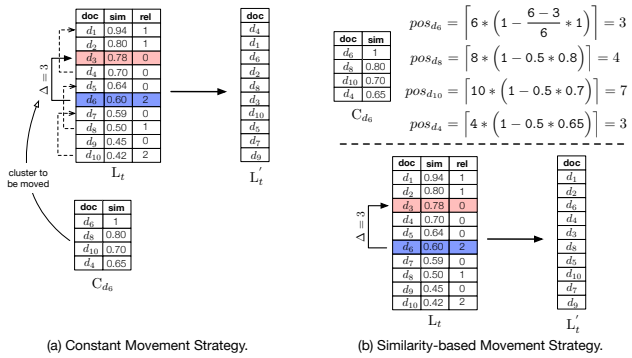

Figure 1: Document movement.

Document clustering is exploited in VATE^2 in order to understand which documents would be affected by the movement of a target document indicated by the developer, using a variation of the cluster hypothesis [5] that we could call the *failure hypothesis*: "closely associated documents tend to be affected by the same failures", stating the common intuition that a given failure will affect documents with common features (in our example all the documents where the term `computing` appears), and, consequently, that a fix for that failure will have an effect on the documents sharing those common features.

The movement of the document and the related document cluster happens according to two alternative strategies. The first is a straightforward algorithm that moves the documents in the cluster by the same amount of positions as the document dragged and dropped by the developer, as shown in Figure 1.(a). The second is a slightly more sophisticated algorithm that takes into consideration the similarity of the documents in the cluster to the document selected by the developer and then moves the documents in the cluster by an amount of positions proportional to their similarity to the document dragged and dropped by the developer, as shown in Figure 1.(b).

Once the new ranked list has been produced by using a clustering and movement strategy, the performances of this new ranked list are computed and the corresponding new line is shown to the developer so that s/he can assess whether the hypothesized modification may be beneficial or not. In the former case VATE^2 turns on a green light to indicate to the developer that s/he should go on with the fix of the system, otherwise it turns on a red light meaning that the fix may be useless or worsen the system.

4. THE VATE^2 PROTOTYPE

The prototype of VATE^2 is available online[1] and Figure 2 shows its main characteristics. The system is structured in three main parts:

(i) **Experimental collection information (A)**: this component is placed on the left side and it allows the devel-

[1]http://ims-ws.dei.unipd.it/vate_ui/

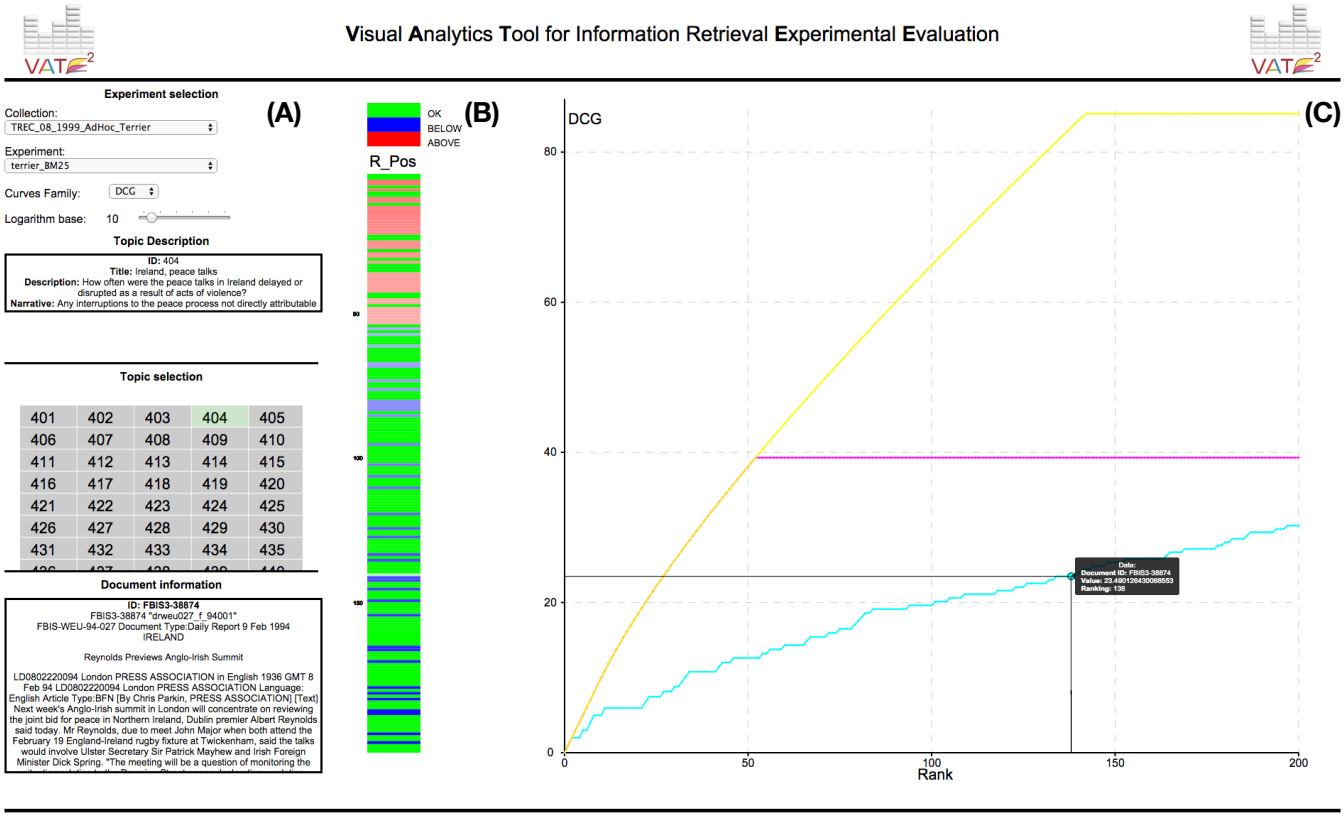

Figure 2: General overview of the VATE^2 system.

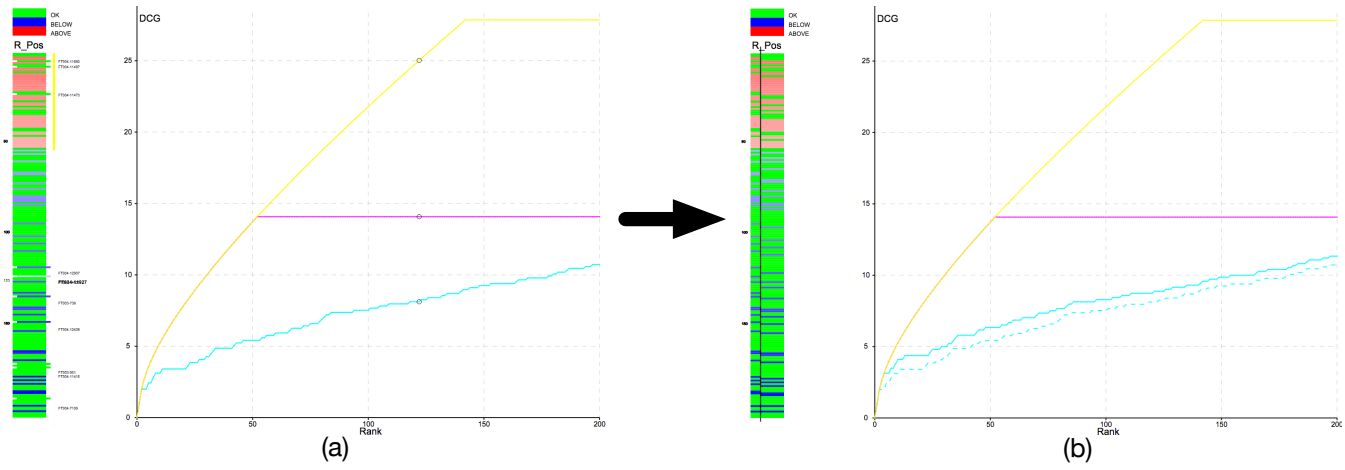

Figure 3: (a) selection of a document and highlight of its cluster; (b) the ranked list and the DCG curve after the movement.

oper to inspect and interact with the information regarding the experimental collection. More in detail, it is divided into three sub-components. The first is the "Experiment Selection" where the developer can select the experimental collection, the experiment to analyze and the evaluation measure and its parameters. The second sub-component is the "Topic Information" composed of the structured description of the topic and the topic selection grid. The third sub-component is the "Document Information" reporting the content of the document under analysis.

(ii) **Ranked list exploration (B)**: this component is placed on the center and shows a visual representation of the ranked list. More in detail, the documents are represented as rectangles ordered by rank from top to bottom where the color indicates the *RP* value; green rectangles indicate well-placed documents, blue rectangles indicate the documents placed below their ideal position and red rectangles indicate the documents placed above their ideal position. The intensity of the color encodes the severity of the misplacement, the more intense the worse the misplacement.

(iii) **Performance view (C)**: this component is placed on the right side and shows the performance curves of the selected experiment. The yellow curve is the ideal one, the magenta curve is the optimal one and the cyan curve is the experiment one. The developer can analyze the trend of the experiment by comparing the behavior of its curve with the ideal and optimal ranking by spotting the possible areas of improvement. Note that for simplicity we talk about "performances", even though we are considering the effectiveness of the evaluated systems.

The developer can interactively select the topic to be analyzed in the topic selection grid and the ranked list and the performance curves are updated accordingly to the selected topic for the given experiment. The user can select the effectiveness measure to be used among: *Cumulated Gain (CG)*, DCG, *Normalized Cumulated Gain (nCG)* and *Normalized Discounted Cumulated Gain (nDCG)*. The ranked list can be dynamically inspected by hovering the mouse over the documents: for each inspected document, the system shows its content on the left in the "Document information" area and marks with a circle the position of the document on the curves in the "Performance view". Moreover, the developer can interact with the "Performance view" by hovering the mouse over the curves which, by means of a tooltip, reports information about the document and the performance score. When a point in the DCG curve is selected, VATE^2 highlights the rectangle of the corresponding document in the ranked list giving a different perspective to the developer who can have a visual indication about the misplacement of the document thanks to the color of the rectangle.

Concerning what-if analysis, as shown in Figure 3.(a), once the developer selects a document, the system displaces on the right the rectangles corresponding to the documents in its similarity cluster and reports their identifiers also on the right. Moreover, a vertical yellow bar highlights the ideal interval in which the selected document should be placed. In Figure 3 Δ quantifies the movement of a document in terms of the number of positions it has been moved.

Once the developer selects a document, s/he can drag it to a new position in the ranked list; afterwards, the document along with its similarity cluster is moved in the new positions. This action is visually shown to the developer and it is represented with an animated movement of the corresponding rectangles to the new positions.

The effect of a movement is shown in Figure 3.(b). We can see an effect on the ranked list which is now split in two parts: on the left there is the old ranked list while on the right there is the new ranked list produced after the movement. In this way the developer can visually compare the effects of the movement and see what other documents have been affected by it. The effect of the movement can be also assessed in the "Performance view" where the new experiment curve (solid stroke) can be compared with the old one (dashed stroke). By analyzing both the ranked list and the performance curves the developer can understand whether their hypothesis about the fix leads to an improvement or deterioration in the performances.

All these operations can be iterated as many times as the developer needs to validate possible alternative fixes of the system.

From the design point-of-view, the adopted solutions reflect the way in which the system is typically used by experts. The "Performance view" part is as big as possible to allow experts a comfortable analysis of the system performances, which is the starting point of every investigation, and calls for handy tools for spotting each change in the DCG curve. Then, the interaction over the performance curve coordinated with the ranked list bar on the left facilitates developers in understanding the critical areas in the ranking, while the dynamic presentation of document content on the bottom left with respect to topic content on the top left supports the identification of possible failure causes.

5. FINAL REMARKS

In this paper we introduced the idea of what-if analysis as a new phase in experimental evaluation and we developed the VATE^2 prototype which demonstrates what what-if analysis is and how it works.

The identification of which documents are affected by a modification and the estimation of how the movement of one of them will cause the others to move is an extremely challenging problem. So, future work will concentrate on refining the techniques we have adopted to this end.

6. REFERENCES

[1] M. Angelini, N. Ferro, G. Santucci, and G. Silvello. A Visual Interactive Environment for Making Sense of Experimental Data. In *Proc. 36th European Conference on IR Research (ECIR 2014)*, pages 767–770. LNCS 8416, Springer, 2014.

[2] M. Angelini, N. Ferro, G. Santucci, and G. Silvello. VIRTUE: A visual tool for information retrieval performance evaluation and failure analysis. *Journal of Vis. Lang. & Comp. (JVLC)*, 25(4):394–413, 2014.

[3] N. Ferro and G. Silvello. What-If Analysis: A Visual Analytics Approach to Information Retrieval Evaluation. In *Proc. 7th Italian Information Retrieval Workshop (IIR 2016)*. CEUR Workshop Proc. (CEUR-WS.org), 2016.

[4] K. Järvelin and J. Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. on Inf. Sys. (TOIS)*, 20(4):422–446, October 2002.

[5] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, England, 2nd edition, 1979.