Salience-induced Term-driven Serendipitous Web Exploration

Yannis Haralambous[1] and Ehoussou Emmanuel N'zi[2]

[1] IMT Atlantique & UMR CNRS 6285 Lab-STICC,
CS 83818, 29238 Brest Cedex 3, France
yannis.haralambous@imt-atlantique.fr ORCID 0000-0003-1443-6115
[2] IMT Atlantique & UMR CNRS 6285 Lab-STICC,
CS 83818, 29238 Brest Cedex 3, France
La mètis, 22 rue d'Aumale, 75009 Paris, France ehoussou.nzi@imt-atlantique.fr

Abstract. The Web is a vast place and search engines are our sensory organs to perceive it. To be efficient, Web exploration ideally should have a high serendipity potential. We present a formalization of Web search as a linguistic transformation, evaluate its stability, and apply it to produce *serendipity lattices*, containing suggestions of term chains to be used as exploration paths. We show experimentally that these lattices conform to two of the three serendipity criteria: relatedness and unexpectedness.

According to www.statista.com, around 4.2 billion people are active Internet users. The Web is a vast place (it contains at least 5.4 billion pages according to www.worldwidewebsize.com) and therefore Web search engines are pivotal tools in daily human activities, all around the world. In the same way as our physical sensory organs process a huge amount of information but transmit only a relevant part of it to the brain, Web search engines provide us with a very limited number of pages, the criterion of choice being (or supposedly being) *relevance*.

Considering Web search as a sort of perception, we redefine *salience* as the appearance of some information object (information objects include URLs, text segmented and analyzed as terms, synsets, concepts, etc.[3]) in a given amount of top-ranked results of a Web search. When an information object I "appears" (we will define exactly what we mean by this) in a Web search based on a query q, we will say that I is *q-salient*.

One can explore the Web by following links, or by using terms (*term-driven* Web exploration) provided as queries to Web search engines. When these queries are built automatically out of terms included in the documents obtained by previous queries, we call this *Automatic Query Expansion* [5] based on *Pseudo-Relevance Feedback* [11, 2]. In the literature, this kind of exploration is usually oriented by semantic relatedness.

[3] Information objects obtained by the search engine or contained in the documents pointed to by the URLs can also include images, videos and other multimodal objects, in this paper we consider only textual objects.

On the other hand, in the last years more and more importance is given to *serendipity* as scientific progress vector. Information systems are studied with respect to their serendipitous potential (called *serendipity factor* by [9]), and serendipity plays also an important role in recommendation systems [10, 8].

When it comes to Web exploration, total randomness is not a viable strategy since it would distance the user more and more from eir centers of interest; on the other hand, keeping a semantic relatedness between terms used as queries would keep the user in a narrow domain but would not allow for any potential for surprise. This is where *serendipity-oriented exploration* proves useful [3]. According to [7] there are three aspects of serendipity:

1. *relatedness*: the new terms must be related somehow with the previous ones, for the exploration to make sense;
2. *unexpectedness*: they should be unexpected (where "expected" terms can be either those who are semantically related to the previous ones, or those that users commonly combine in queries);
3. *interestingness*: they should represent entities that are important to the user.

We present an algorithm that will, out of two terms, produce a *lattice* of terms (synsets, concepts, etc.), the paths of which can be used for serendipitous Web exploration: if t and t' are the initial two terms, we obtain terms t_i which are *related* to t and t' since they are $(t \wedge t')$-salient, and *unexpected* with respect to t since they are not t-salient. Every term that is included in the lattice in this way is automatically attached to one or more preexisting paths, and all paths end at t': we call them *serendipitous exploration paths*.

To choose candidate terms t_i inside information objects, our approach uses linguistic methods (POS tagging, dependency relations, ...), therefore we consider the mapping from a term pair to a term lattice as a linguistic transformation. But for that to make sense, we first need to prove that the process is stable enough to produce consistent results in a reasonable time window.

This paper is structured as follows: we first build a formal framework to study Web search and describe the information objects involved; we then experimentally prove stability of the Web search operation in our framework; and finally we present the algorithm of lattice construction and experimentally show that it meets the criteria of relatedness and unexpectedness.

1 The Framework

On an abstract level, Web search is an act of communication between a human and a machine (the server). The motivation of this act for the human is to find information or knowledge on the Web, the message sent by the human is encoded in a formal language (the language of *queries*), it is transmitted through HTTP protocol to the server of the search engine, and the latter replies by a Web page containing a ranked collection of information objects. A Web search session is a sequence of query/results pairs, where the human user refines queries until the

results fulfill eir needs. A Web exploration session is a search session where the user has a more global need: to explore a domain, where surprise can work as a trigger for serendipitous ideas, inventions or discoveries.

We use the term "information objects" to keep a certain level of generality: they can be URLs, page titles and snippets (containing selected excerpts from the pages pointed to by the URLs), but also contents of documents pointed to by URLs, various metadata provided by the search engine, etc. This amounts to a collection of heterogeneous data that can be analyzed in various ways. We will restrict ourselves to textual contents, in order to be able to use natural language processing methods.

Formalizing Web search as a linguistic transformation carries some inherent difficulties:

1. Web search results, which are ranked collections of information objects, contain *heterogeneous* and *engine-dependent* information; URLs are often accompanied by text snippets extracted from the URL target, as well as other information;
2. Web search results are partly *random*; indeed, even repeating the same query in a short lapse of time will *not* systematically produce the same result (depending of course on the measures used to compare information objects);
3. information object collections are *ranked*, and this rank—even though the algorithm the search engine has used to obtain it may not be totally impartial—has to be taken into account in the evaluation of results;
4. disambiguation of queries is a difficult process because the more terms one sends (to disambiguate by collocation) the less pages contain all terms of the query, and these pages are not necessarily the most interesting ones;
5. depending on the requirements of a specific task, the nature of the information objects needed by the user can vary;
6. information objects need to be compared, and for this we need specific measures which should ensure convergence of the searching process: despite their randomness, information objects need some stability properties so that the pseudo-relevance used to obtain a given exploration strategy remains valid in a reasonable time window.

In the following we will consider separately the different facets of the problem.

1.1 Queries

The Web search communication process is asymmetric: while the information objects received from the search engine have a rich multilevel structure, the *query* required to obtain them is a mostly short utterance in a very restricted formal language, the vocabulary of which belongs to a given natural language. We define the *query language* as:

Definition 1 (Query language)**.** *Let $\mathfrak{E}$ be a sufficiently large set of English language terms (in the sense of noun phrases having a high termhood value in a given knowledge domain, cf. [6]). We define the* ***query language*** $\mathfrak{Q}$ *as being*

the regular formal language based on the alphabet $\mathfrak{E} \cup \{$ "(", ")", "$\wedge$ ", "$\vee$ ", "$\neg$"$\}$, *and the following rules:*

$$\begin{aligned} S &\to W, \\ W &\to W \wedge W \mid W \vee W \mid \neg W \mid (W), \\ W &\to m, \end{aligned}$$

where $m \in \mathfrak{E}$ *and* S *is the start symbol.*

Note that we do not include the (obvious) rule of term concatenation $W \to W\ W$ because it is ambiguous and search engine-dependent.

Words of the formal language $\mathfrak{Q}$ have to be encoded before transmission: this involves surrounding elements of $\mathfrak{E}$ by ASCII double quotes `"..."`[4] and replacing symbols $\wedge, \vee, \neg$ by character strings `AND`, `OR`, `NOT`. Furthermore there are constraints on the character lengths of queries: in the search engine we will consider for experimentation, the maximal length of an encoded query is of 1,500 characters.

There are many possible enhancements to the definition of $\mathfrak{Q}$ which we will not consider in this paper: use of proximity operators, (upper or lower) casing, use of search engine user preferences, use of HTTP headers, memorization of previous queries by the search engine, etc. In our experiments we requested English text results and deactivated memorization of query history and adult content filtering.

2 Information objects

The definition of information object should be flexible enough to be adapted to various search engines and user needs. We proceed as follows. Let N (*repetition rate*) and M (*dimension*) be numbers and q a query in $\mathfrak{Q}$:

1. we systematically repeat the query q N times is a short time window and merge the results as a ranked weighted collection of information objects;
2. we restrict this collection to the M top-ranked results;
3. we define *salience* as the occurrence of an information object in this set of results;
4. information objects are annotated by their *weight vectors*: these M-dimensional vectors store a weight computed out of the their presence at the various ranks of the collection, and other characteristics (e.g., as dependency distance from some other term).

In this paper we will use $N = 10$ and $M = 50$.

There are two kinds of information objects, the distinction depending on the web search engine features:

[4] Double quotes are necessary for some search engines to avoid replacement of the English term due to spelling correction on the search engine side.

1. **local** information objects, which are returned directly by the search engine (URLs, page titles and snippets, metadata) or extracted from them: terms, synsets, concepts, and more;
2. **remote** information objects which are extracted from documents pointed to by the URLs.

The difference between the two is of practical nature: documents to download can be of very variable size, and analyzing them can be complex and error-prone. It should be noted that (local) text snippets are of relatively uniform size and have been chosen by the search engine as being representative of the document—normally they contain the individual query terms.

As an example of local information objects, here are the URL, Web page title and text snippet obtained with rank 21 when sending the query `misapplied` to a major search engine:

> URL: `https://eu.usatoday.com/story/opinion/policing/spotlight/2018/05/08/justice-system-false-imprisonment-policing-usa/587723002/`
>
> Title: 'Misapplied justice' causes innocent men and...
>
> Snippet: 08/05/2018 · '**Misapplied** justice' causes innocent to suffer. It's estimated that about 20,000 people in prison have been falsely convicted, with blacks wrongfully incarcerated at higher rate than whites

As we can see in the example, neither the title nor the snippet need to be grammatical sentences, and the snippet contains the query term.

In this paper we consider only local information objects, but it is always possible to switch to remote ones when search strategy requires it (formalization and algorithms are the same). More specifically we will consider only URLs, terms, synsets and concepts, even though the framework is valid for many more types.

Definition 2 (Information Object)**.** *We call* ***information object*** *of dimension M a pair (t, v) where t is an URL (including the empty URL ε), or a term, or a synset, or a concept, and $v \in \mathbb{R}^M$.*

We call t the ***content*** *and v the* ***weight vector*** *of the object. The order of elements of v represents rank in the collection of information objects returned by the search engine.*

Let us denote by $\mathbb{O}_U$, $\mathbb{O}_T$, $\mathbb{O}_S$, $\mathbb{O}_C$ the sets of information objects (where contents are respectively URLs, terms, synsets, concepts).

Definition 3 (Web Search)**.** *Let $q \in \mathfrak{Q}$ be a word of the query language, $*$ an information object type in $\{U, T, S, C\}$, and $t_{1\ldots N} \in \mathbb{R}_+^N$ an N-tuple of time values. Let $\phi_{*,t_{1\ldots N}}(q)$ be the result of a (N-repeated) Web search of q at times $t_{1\ldots N}$, where the i-th elements of weight vectors are defined as $v_i := \frac{1}{N}\sum_{j=1}^{N} \lambda_{i,j}(I)$, where $\lambda_{i,j}(I)$ can be either the presence (or the result of some linguistic computation) of some $I \in \mathbb{O}_*$ at the i-th rank of the collection of information objects returned by the search engine at the j-th iteration of the query.*

The value of $\phi_{*,t_{1\dots N}}(q)$ can be considered as an element of $2^{\mathbb{O}_*}$ (that is a set of arbitrary size of information objects of type $*$). In other words, we have, for each $*$:

$$\begin{aligned}\phi : \mathbb{R}_+ \times \mathfrak{Q} &\to 2^{\mathbb{O}_*}\\ (t_{1\dots N}, q) &\mapsto \varphi_{*,t_{1\dots N}}(q).\end{aligned}$$

Notice whenever we will consider $\{(t_1, v_1), \dots, (t_n, v_n)\}$ as being the image of $\phi_{*,t_{1\dots N}}(q)$, we will consider that (a) $t_i \neq t_j$ for all $i \neq j$, and (b) there is at least one nonzero v_i.

We will show in Section 3 that φ is statistically independent of $t_{1\dots N}$, in a reasonable time window, and hence can be considered as an application of queries into information object sets.

2.1 URLs

URLs [4, § 3] are the most fundamental information object contents. The URL of a Web page is divided in five parts: (a) the *scheme* (`http` or `https`); (b) the *authority* (machine name, domain, domain extension); (c) the *path* to the remote file (in Unix notation); (d) following a question mark, the *query*: one or more attributes in name-value form; (e) the *fragment*: whatever follows the #. For the sake of simplicity we will keep only parts (b) and (c). Let us call $\mathbb{U}$ the set of parts (b) and (c) of all valid URLs according to [4], to which we add the empty URL ε.

The number of URLs returned by search engines varies depending on the frequency of the terms queried in the Web corpus and on the search engine's resources. As a reasonable compromise between an insufficiently small number of URLs and an overwhelming amount of increasingly irrelevant URLs, we have chosen to keep only the 50 top-ranked URLs in each result provided by the search engine, and this has lead to the definition of information content with URL content of dimension $M = 50$.

2.2 Terms

If we restrict ourselves to local information objects, terms are obtained in the following way: we parse the Web page titles and text snippets provided by the search engine in order to obtain lexemes (standard forms), POS tags and syntactic dependencies. Then,

- in the simplest case we keep only nouns and named entities;
- if required, we keep also noun phrases with specific POS tag combinations (adjective + noun, etc.) that can be considered as terms;
- in specific knowledge domains we can also filter these noun phrases by calculating their *termhood* coefficient (cf. [6]).

In other words we have a spectrum of methods ranging from simple noun filtering to full terminological extraction.

The value of each coordinate of the weight vector of a term information object can be binary (present or not at a given rank) or a weight calculated as the termhood or some other characteristic of the occurrence of the given term in the given title+snippet (for example, if there is a dependency chain between the term and one of the terms of the query, a weight depending on the length of the chain and the nature of its dependencies). In the case of remote information objects, one can also consider tfidf and similar relevance metrics.

2.3 Synsets

A synset is a sense given (a) as the common sense of a group of words and (b) by a short glose. WordNet, for example, is a graph of synsets. Using word disambiguation methods we can attach synset IDs to terms (if there is ambiguity, we keep either the most probable synset or all probable synsets). The weight can then be the probability that the term belongs to the current synset.

2.4 Concepts

By semantically annotating terms, we can attach them to specific concepts in standard or ad hoc ontologies. As semantic annotation will be done on-the-fly, there can be many matchings for a given term, and hence we can either keep the most probable matching or keep all of them as different information objects. The weight will then be the relevance of the matching.

The difference between "synsets" and "concepts" is of practical nature in this framework: in the former case we consider WordNet or a similar resource as a single knowledge resource and weight information objects by the probability of belonging to a given synset (resulting from word disambiguation methods); the latter case corresponds to search strategies in specific knowledge domains where (standard or ad hoc) ontologies exist or are in the process of being built. The weight can correspond to the relation between the concept and the concept(s) represented by the term(s) of the query.

3 Stability of Web Search

To study the stability of $\varphi_{*,t_{1\ldots N}}$ we start by the most fundamental information objects, namely URL information objects: $\varphi_{U,t_{1\ldots N}}$. Let $q \in \mathfrak{Q}$. The result of $\varphi_{U,t_{1\ldots N}}(q)$ is an element I of $2^{\mathbb{O}_U}$, that is a set $\{(t_1, v_1), \ldots, (t_k, v_k)\}$, where t_i are URLs and v_i weight vectors.

If $I = \{(t_1, v_1), \ldots, (t_k, v_k)\}$, we will denote by $\mathcal{C}(I) = \{t_1, \ldots, t_k\}$ the set of content elements of I.

Definition 4 (Information Object Distances)**.** *Let* $I = \{(t_1, v_1), \ldots, (t_k, v_k)\}$ *and* $I' = \{(t'_1, v'_1), \ldots, (t'_{k'}, v'_{k'})\}$ *be URL information objects.*

Let $K = \#(\mathcal{C}(I)) + \#(\mathcal{C}(I') \setminus \mathcal{C}(I))$. Let us renumber I' by a function ℓ such that $\ell(i) = j$ if there is a t_j in I such that $t_j = t'_i$, and otherwise ℓ is an injective function into $k+1, \dots, K$.

We can consider I as a vector of $\mathbb{R}^K$ (dimensions $1, \dots, k$) and I' as a vector of the same $\mathbb{R}^K$ (some dimensions between 1 and k, and all dimensions between $k+1$ and K). The coefficients of these vectors cannot be the v_i (resp. $v'_{\ell(i)}$) because these are already vectors in $\mathbb{R}^M$—therefore we "flatten" them using a function f.

Let $f : \mathbb{R}^M \to \mathbb{R}_+$ a function such that if $f(v) = 0 \Rightarrow v = \mathbf{0}$, then we define

$$d_f(I, I') := \frac{\sum_{i=1}^{K} |f(v_i) - f(v'_{\ell(i)})|}{\sum_{i=1}^{K} \max(f(v_i), f(v'_{\ell(i)}))},$$

which is the Soergel distance [12, p. 987].

We will define two distances:

1. *d_{ranked} when $f(v_i) = \sum_{j=1}^{M} \frac{v_{i,j}}{j}$, where $v_{i,j}$ is the j-th weight of the weight vector v_i;*
2. *d_{unranked} when $f(v_i) = 1$ if there is at least one j such that $v_{i,j} > 0$ and $f(v_i) = 0$ if the weight vector v_i is a zero vector.*

Notice that $d_{\text{unranked}}(I, I')$ is the Jaccard distance of sets $\mathcal{C}(I)$ and $\mathcal{C}(I')$.

Definition 5 (d-Stability)**.** *Let d a 1-bounded distance on $\mathbb{O}_U$, that is an application $d : \mathbb{O}_U \times \mathbb{O}_U \to [0,1]$ such that $d(o,o) = 0$, $d(o,o') = d(o',o)$ and $d(o,o'') \leq d(o,o') + d(o',o'')$ for all $o, o', o'' \in \mathbb{O}_U$.*

*We call a querying application $\varphi_{U,t_1,\dots,t_N}$ d-**stable** if*

$$\max_{\substack{t_1<\dots<t_N \\ t'_1<\dots<t'_N}} d(\varphi_{U,t_1,\dots,t_N}(q), \varphi_{U,t'_1,\dots,t'_N}(q)) \lesssim 0.05,$$

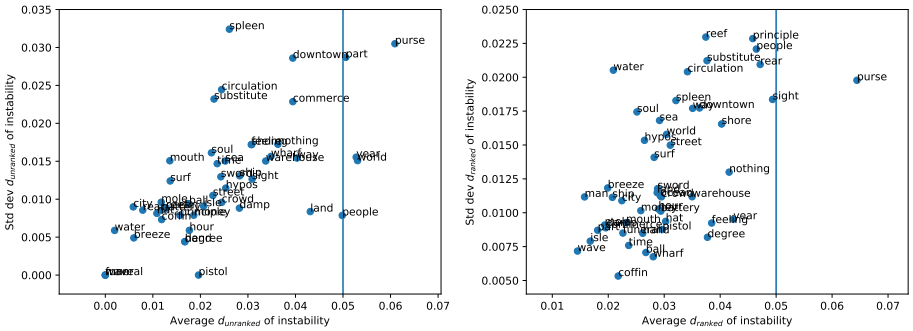
for any $q \in \mathfrak{Q}$. In that case we will write φ_U instead of $\varphi_{U,t_1,\dots,t_N}$.

Hypothesis 1. *The Web search application φ_U is both d_{ranked}-stable and d_{unranked}-stable.*

Experimental validation of Hypothesis 1 Our corpus will consist of the fifty first nouns of *Moby Dick*: *year, money, purse, nothing, shore, part, world, way, spleen, circulation, mouth, damp, soul, coffin, warehouse, rear, funeral, hypos, hand, principle, street, people, hat, time, sea, substitute, pistol, ball, sword, ship, man, degree, feeling, ocean, city, wharf, isle, reef, commerce, surf, downtown, battery, mole, wave, breeze, hour, sight, land, crowd, water.*

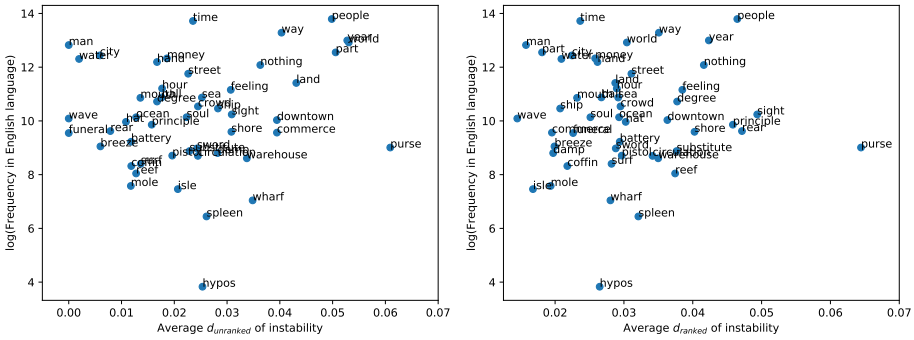
Let $N = 10$ and $M = 50$. For each w in this set we have calculated a first series of ten $\phi_U(w)$ and then a second series, that makes a total of 200 queries for each word. For each w we calculated the distances d_{unranked} and d_{ranked} of the i-th elements of the two series, and then measured the average and standard deviation of the distance measurements.

The results can be seen in the following two diagrams where the x axis represents average $d_{(\text{un})\text{ranked}}$ and the y axis standard deviation of $d_{(\text{un})\text{ranked}}$:



As we see, in the unranked case most distances are under the 0.05 barrier, with a few exceptions: *purse*, *year*, *world*, *part*; in the ranked case only the word *purse* has an instability higher than 0.05. The difference between the two distances is due to the fact that changes in results occur are rather located in lower ranks (near the bottom of the Web page). This is not the case for *purse* where the results seem to oscillate between two versions differing also in higher ranks. Note that the two distances are significantly different: their correlation coefficient is not higher than 0.476.

One could wonder whether instability is related to the frequency of words. Indeed, there is a small positive correlation ($\rho_{\text{unranked}} = 0.184$) between the frequency of words in English language[5] and unranked instability, but this correlation almost disappears when ranked distance is measured ($\rho_{\text{ranked}} = 0.037$):



Differences between the two diagrams depend on the word (e.g., they are important for *rear* and unimportant for *people*) and show whether instability lies in lower ranks or can be found at all ranks.

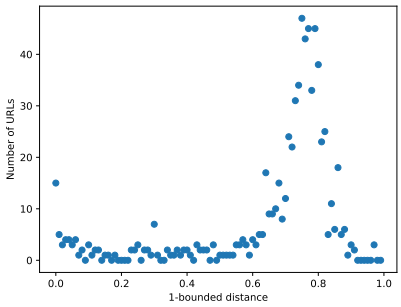
[5] Taken from the Corpus of Contemporary American English `https://corpus.byu.edu/coca/`.

The next step is to prove the stability of φ_T, φ_S, φ_C. As these rely entirely on the titles and text snippets accompanying URLs, it suffices to show that title and snippet for a given URL do not vary when queries are repeated.

Experimental validation of the invariance of title and snippet

On a corpus of 1,784 queries (108 unique queries) we obtained 86,493 URLs (7,211 unique ones). Among the 86,493 URLs, only 618 (that is 0.7%) had more than one different snippet (out of which 577 had two different snippets, 29 had three, 9 had four, and there were single URLs with 6, 7 and 10 different snippets respectively).

We calculated a 1-bounded distance (more precisely, the Levenshtein distance divided by the length of the longest snippet of the two) between the snippets for the 618 cases of URLs with more than one snippet. Here is the distribution of this distance:



There is a peak at approx. 0.8 and the most plausible hypothesis explaining it is that this is the distance between entirely different snippets having only the query terms in common.

4 Salience and serendipity lattices

In the following we deal with elements of $\mathbb{O}_T$ because their content elements (i.e., terms) can be used directly in queries, which is not the case of elements of $\mathbb{O}_S$ and $\mathbb{O}_C$ (see the "Future Works" section).

We first define a more restrictive version of φ_T:

Definition 6 (Valid Web Search)**.** *Let a query q be equal to $t \wedge t' \wedge \cdots \wedge t^{(n)}$, i.e., a conjunction of terms. Let $\varphi_T(q)$ be the Web search with query q as an application $\mathfrak{Q} \to 2^{\mathbb{O}_T}$.*

We call the search $\varphi_T(q)$ ***valid*** *if (a) there are elements $(t, v), \ldots, (t^{(n)}, v^{(n)})$ (corresponding to the terms $t, t', \ldots, t^{(n)}$ in the query) in $\varphi_T(q)$, and (b) these elements have weight vectors v such that if for some j_0, $v_{j_0} = 0$ then all information objects of $\varphi_T(q)$ have weight vectors with zero j_0-th dimension.*

In other words, among the results returned by the search engine, we consider only those the snippet of which contains *all* query terms. On a practical level,

to obtain a valid Web search one has to consider as many URLs as necessary to obtain M snippets containing the query terms. In theory, one has to examine *all* URL snippets until obtaining the M first ones fulfilling our need. But in reality search engines tend to display those URLs first, and it is highly probable that when a snippet missing some query term is displayed then all following snippets also miss at least one query term.

Every Web search can be transformed into a valid Web search if we remove the URLs the snippets of which do not contain *all* query terms. Therefore we can always assume that a given Web search is valid.

Let us now define the notion of salience:

Definition 7 (Salience)**.** *Let q be a query and $\varphi_T(q)$ a valid Web search. Let t be a term. We say that t is q-**salient** if there is a (nonzero) occurrence vector v such that $(t, v) \in \varphi_T(q)$.*

Obviously every t belonging to q is q-salient, since the search is valid. This constraint is very important because, once exhausted the results containing all query terms, search engines will provide "partial snippets," i.e., results in which *not all* terms of the query appear. It is this constraint that guarantees our algorithm stopping after a finite number of steps: by adding more and more terms to the query, the number of pages containing *all of them* will necessarily decrease until reaching zero.

Definition 8 (Serendipity lattice of t, t')**.** *Let t, t' be two terms. We will call **serendipity lattice** of t, t' a lattice (the partial order of which we denote by $\succ$) of terms with t as supremum and t' as infimum, with the following properties:*

1. *all paths $t = t_1 \succ \cdots \succ t_m \succ t'$ are of equal length m (the height of the lattice);*
2. *for any element t_i in such a path, t_i is not $(t_1 \wedge \cdots \wedge t_{i-1})$-salient;*
3. *t' is $(t_1 \wedge \cdots \wedge t_m)$-salient.*

Intuitively, every path $t_2, \ldots, t_m$ is a sequence of terms to add to t so that t' becomes salient in the resulting query, but not in any smaller query. The fact that no term is salient with respect to the terms preceding it in the path guarantees the minimality of the path: every intermediate term is necessary. All paths are of the same length because in our algorithm we stop when a path makes t' salient, after having finished examining all paths of the same length.

Algorithm 1 (p. 12) provides the serendipity lattice S of a pair of terms t, t'. Some explanations:

- we consider S as a graph whose edges (s, s') represent orderings $s \succ s'$. In the first part of the algorithm, edges are added to S, but S has not yet a lattice structure because some paths may not lead to t'. In the second part of the algorithm we remove the unnecessary edges by a bottom-up approach: we start from t' and keep in S only the edges we encounter while going upwards;
- queries q and q' are built by progressively adding conjunctive terms and the function "last" returns the rightmost conjunctive term. Note that conjunction is not commutative, in the sense that $\varphi_U(q \wedge q')$ is not necessarily equal to $\varphi_U(q' \wedge q)$;

Algorithm 1: Calculation of the serendipity lattice S of the pair of terms t, t'.

```
Data: Terms t, t', (valid) Web search application φ_T
Result: A serendipity lattice of t, t', that is a lattice S with t as supremum and
        t' as infimum
/* Calculating a superset of S                                              */
if t' is t-salient then
    S = {(t, t')}
else
    Q ← {t};
    Q' ← {t ∧ t'};
    MaxLength ← ∞;
    for q' ∈ Q' do
        if |q'| ≤ MaxLength then
            for x ∈ Filter(φ_T(q')) do
                for q ∈ Q do
                    if |q| ≤ MaxLength then
                        if x is not q-salient then
                            Q ← Q ∪ {q ∧ x};
                            /* last(q) means last conjunctive term in q  */
                            S ← S ∪ {(last(q), x)};
                            if t' is not (q ∧ x)-salient then
                                Q' ← Q' ∪ {q' ∧ x}
                            else
                                S ← S ∪ {(x, t')};
                                if MaxLength = ∞ then
                                    MaxLength ← |q|
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
/* Removing unnecessary edges from S                                        */
A ← {t'};
S' ← ∅;
while S ≠ ∅ do
    for (s, s') ∈ S do
        if s' ∈ A then
            S' ← S' ∪ {(s, s')};
            A ← A ∪ {s};
            S ← S \ {(s, s')};
        end
    end
end
S ← S';
```

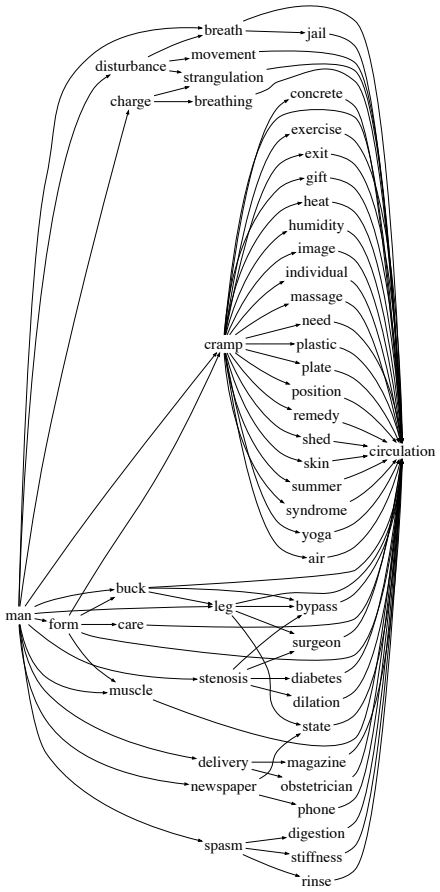


Fig. 1. Serendipity lattice for the pair *man – circulation*.

- $|q|$ denotes the number of conjunctive terms in q, in particular $|t| = 1$;
- by ∞ we represent a sufficiently large number;
- the "Filter" function will limit the amount of terms used to create further queries and avoid combinatorial explosion. We weight terms in $\varphi_T(q')$ as follows: $w(t_0) = \sum_{i=1}^{M} \frac{r_i}{i+\varepsilon}$ where r_i is the binary value of the i-th coordinate of the weight vector of t_0 and $0 < \varepsilon < 0.1$ is a random number, whose goal is to keep the weight values distinct, so that w is an injective function. After weighting the terms we filter their set by keeping only a limited number (1,000) of top-weight terms.

Note that the algorithm will necessarily stop because of the constraint of the snippet containing all conjunctive terms of the query: when the query becomes complex enough, either we will attain some topmost result of the query $t \wedge t'$ and hence t' will become salient, or the search engine will not be able to produce

any result with snippet containing all query terms and hence the algorithm will stop.

See Fig. 1 for an example of serendipity lattice with 50 vertices and 100 edges, for the pair *man – circulation* from the "Moby Dick" corpus.

4.1 Discussion

We call S a *serendipity lattice* because we claim that the paths between supremum and infimum of S have a serendipity potential as Web exploration paths, let us see why.

As we already mentioned in the introduction, according to [7] there are three aspects of serendipity: *relatedness*, *unexpectedness* and *interestingness*.

We will first develop our argumentation about the conformity of our method to the two first criteria, and then attempt to prove it experimentally on the corpus of 50 first Moby Dick nouns.

Argumentation Let t, t' be the pair of terms out of which we create a serendipity lattice S. Our method conforms to the criterion of *relatedness* since every x in the lattice is $(t \wedge t' \wedge \cdots)$-salient, i.e., appears in snippets together with t and t'. It conforms to the criterion of *unexpectedness* because every x in the lattice is *not* $(t \wedge \cdots)$-salient, i.e., does *not* appear in any of the snippets obtained by the previous query.

Experimental proof of relatedness To evaluate relatedness of serendipity lattices we will use the FrameNet v1.7 corpus (`https://framenet.icsi.berkeley.edu/`). In this context, a *frame* is "a schematic representation of a situation involving various participants, props, and other conceptual roles" (Wikipedia). Version 1.7 of FrameNet comes with 1,073 frames and 4,235 distinct nouns. We consider that *being in the same frame is a strong relatedness indicator between two words*.

Our "Moby Dick" corpus (the lattices obtained out of the 50 first nouns of *Moby Dick*) contains 2,331 lattices, which, in turn, contain 9,630 distinct words and 26,779 distinct lattice edges.

2,607 words of our corpus appear in FrameNet, and the extremities of 406 lattice edges belong to the same FrameNet frame, which means that they are strongly related. As a comparison basis, we took the same number of FrameNet words, but this time arbitrarily chosen, created random edges between them and counted how many of them happened to be in the same frame. We repeated the operation 100 times. Here are the results:

Method	# FN words	# pairs of FN words	# pairs in the same frame
In serendipity lattices	2,607	11,097	406
Randomly chosen	2,607	11,097	6.35 ± 2.35

In other words, if the extremities of our lattice edges were unrelated then we would expect to find an average of 6.35 of them (with standard deviation 2.35)

in the same frames. But we find 406, that is more than **sixty times as many** lattice edge extremities belonging to the same FrameNet frame. Hence, these extremities are *related*.

As for *unexpectedness*, we will use the *Word Associations Network* (`https://wordassociations.net`) by Yuriy A. Rotmistrov (see also [1]). We consider that this resource provides all "expected" terms that are mentally associated with a given term. We consider that, *the fact that a lattice edge does not appear in WAN is a strong indicator of unexpectedness*.

For the 9,630 words of the lattices of our "Moby Dick" corpus we retrieved 1,938,534 associations. From the 26,779 lattices edges of our corpus, only 1,782 appear among WAN associations, therefore we can conclude that **93,4%** of our lattice edges can be considered as "unexpected".

5 Conclusion and Future Works

We have presented a formalization of the Web search operation considered as a linguistic transformation. We have proven experimentally the stability of this operation. Then we have introduced two new notions: *q-salience* (whether an information object appears in the results of a Web search of query q) and *serendipity lattice*, which is the result of an algorithm we introduce, and aims at providing paths for serendipitous Web exploration. Finally, we have shown experimentally that our lattices conform to two of the three serendipity criteria: relatedness and unexpectedness.

Future work will consist into extending the approach to be concept-based instead of term-based and to explore evaluation of serendipity potential in more depth. Our goal is to build an interface for assisting the user in exploring the Web efficiently—serendipity lattices will be part of the building blocks of this interface.

Acknowledgments The authors would like to thank the staff of La Mètis for their precious help in implementing the algorithms and performing massive online searches.

References

1. Acar, S., Runco, M.A.: Assessing associative distance among ideas elicited by tests of divergent thinking. Creativity Research Journal **26**, 229–238 (2014)
2. Al-Masri, M., Berrut, C., Chevallet, J.P.: A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In: Proceedings of ECIR 2016. LNCS, vol. 9626, pp. 709–715 (2016)
3. Beale, R.: Supporting serendipity: Using ambient intelligence to augment user exploration for data mining and web browsing. Int. J. Human-Computer Studies **65**, 421–433 (2007)
4. Berners-Lee, T.: RFC 3986. Uniform Resource Identifier (URI): Generic Syntax (2005)

5. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Computing Surveys **44**, 1–49 (2012)
6. Frantzi, K., Ananiadou, S., Tsujii, J.: The C-value/NC-value Method of Automatic Recognition for Multi-word Terms. In: Proceedings of ECDL'98. LNCS, vol. 1513, pp. 585–604 (1998)
7. Huang, J., et al.: Learning to recommend related entities with serendipity for web search users. ACM TALLIP **17**, 25:1–25:22 (2018)
8. Kotkov, D., Wang, S., Veijalainen, J.: A survey of serendipity in recommender systems. Knowledge-Based Systems **111**, 180–192 (2016)
9. McCay-Peet, L., Toms, E.G.: The serendipity quotient. Proceedings of the American Society for Information Science and Technology **48**, 1–4 (2011)
10. Meng, Q., Hatano, K.: Visualizing basic words chosen by Latent Dirichlet Allocation for serendipitous recommendation. In: Proceedings of the 3rd Int. Conf. on Adv. Appl. Informatics. pp. 819–824 (2014)
11. Vaidyanathan, R., Das, S., Srivastava, N.: Query expansion strategy based on pseudo relevance feedback and term weight scheme for monolingual retrieval. International Journal of Computer Applications **105**, 1–6 (2015)
12. Willett, P., Barnard, J.M., Downs, G.M.: Chemical similarity searching. J. Chem. Inf. Comput. Sci. **38**, 983–996 (1998)