

Structure of the Web Searcher's Query Modifications: Sequential, Branching, Merging, Re-Merging and Non-Sequential Execution

Nikolai Buzikashvili

*Institute of System Analysis, Russian Academy of Science
Prospect 60-let Oktyabrya, 9, Moscow, Russia
buzik@cs.isa.ru*

Abstract—This A non-linear model of query modifications during a single session of user searching on the Web is evaluated. In addition to a sequential query modification, in which any query may be reformulation of its direct predecessor, the model allows non-linear interpretation of query reformulations: branching, merging, re-merging (merging of earlier branched chains) and non-sequential execution of logically linear chains of queries. To detect short-term (i.e. during a single time session of interaction with the Web search engine) dependencies between user's queries we use different query-to-query similarity metrics and consider query-on-query(ies) dependency as similarity of current query to queries submitted earlier in a time session. Usage of different metrics leads to different but complimentary results. The method is applied to query logs of the Yandex, the major Russian search engine. Regardless of difference of results of different metrics, the findings are coincide: about 20% of time sessions containing 3 and more distinct queries contain such combinations of queries which can be interpreted as non-linearities: branchings are occurred in 12% of sessions; mergings are occurred in 6%, and re-mergings in 3%. Number of branches are a little bigger than two; the first branch is smaller then the last. A merging search is frequently executed in the re-merging manner. The last merging chain is smaller then the first one.

Keywords— query modification, branching, merging, re-merging

I. INTRODUCTION

The paper considers a query modification process in terms of dependencies between queries submitted by the same user and is centered on non-linear dependencies. Early conceptual works [1], [2], [6], [12] on algorithmic-like description of information searching behavior don't exclude a possibility of a non-linear search: a search process may contain several branches [2] and a complex search task may be decomposed into different chains [12], which merge at the final step. However, following empirical studies in the field of the Web user searching behavior (e.g. [9], [10]) and works [7], [8] more centered on improvement of the search results use a linear search framework in which a query can be a modification only on its direct predecessor: $Q_T = f(Q_{T-1})$.

One of the reasons was an absence of a technical model describing non-linear dependencies and tools to detect them in query logs. Such models are introduced in [3], [4], [11] and [5], [11] investigate several structural properties of these modifications. In the paper, we follow to the conceptual

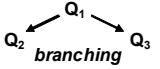
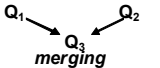
framework of [3], [4]. Working models [11] on the one hand and [4], [5] on the other hand differ in allowed query-to-queries dependencies and in used dependency metrics and query string analysis.

(1) a query may depend on non-direct predecessor: $Q_T = f(Q_{S < T})$ instead of $Q_T = f(Q_{S = T-1})$. As a result, several queries may depend on the same query (a *branching* search: queries $Q_{T1} = f(Q_S)$, ... $Q_{TN} = f(Q_S)$ depend on the same query Q_S). A model doesn't limit a number of chains $\{Q_S \Rightarrow Q_{T1} [= \dots]\}$ which have the same root Q_S

(2) a query may be a "combination" of a *pair* of earlier submitted queries $Q_T = f(Q_{S1 < T}, Q_{S2 < T})$ (a *merging* search) rather than a modification of a single query. A model restricts a number of determining predecessor by a pair as a maximum. This restriction is motivated by neurophysiological mechanisms of human information processing according to which parallel execution of more than two tasks (finally merging query modification chains) is practically impossible.

Interpretations of the same series of user queries vary depending on the query dependencies taken into account. Table I shows differences between linear and non-linear interpretations of two real life time serieses of user queries.

TABLE I. OBSERVED USER QUERIES AND THEIR INTERPRETATIONS

Observed sequence	Linear interpretation	Non-linear interpretation
$Q_1 = \text{Chinese nosh cook wanted}$ $Q_2 = \text{Chinese nosh}$ $Q_3 = \text{Chinese cook wanted}$	$Q_1 \xrightarrow{\text{broad}} Q_2 \xrightarrow{\text{vary}} Q_3$ or $Q_1 \xrightarrow{\text{broad}} Q_2 \quad Q_3$	
$Q_1 = \text{crown buy}$ $Q_2 = \text{toyota moscow}$ $Q_3 = \text{toyota crown moscow}$	$Q_1 \quad Q_2 \xrightarrow{\text{vary}} Q_3$	

However, are the non-linearity assumptions (a) psychologically grounded and (b) technically supported by search engine interfaces? A *branching search* is not a surprising manner for any living being. A human searching on the Web comes up against a common situation provoking a branching search: when an initial query perfectly expresses an information need but retrieves unsatisfactory results, a user modifies the query; if the results of the modified query are also unsatisfactory, a user refines the *initial* query rather than the current one. A *merging search* is poorly supported by a human

memory and is not supported by search environments. However, interviewing of ordinal searchers shows that both manners are *consciously* applied when a “direct search” gives unsatisfactory results. Besides, we can expect a lot of “unconscious” branching modifications of an initial query perfectly presenting an information need but retrieving poor results.

II. CONCEPTUAL FRAMEWORK

We use term-based query-on-queries dependencies as representatives of a query modification process. If there is a set of queries which may be determinants of a query (i.e. precede a query and have a non-zero influence on it according to a certain dependency measures), a query considered as modification of the recent among strongest determinants. All dependencies are detected for queries submitted during a short-time period (a time session). To determine dependency of a query only its first occurrence in a time session is used, and a query may depend on queries submitted before its first occurrence. The strongest dependency of a query is selected according to certain decision rules.

Let dependencies between queries be presented by a graph of a logical structure of a search, in which each query be presented by a unique node regardless of a number of occurrences of the query in a session. The connected components of this graph correspond to “task sessions”. The same logical structure may present different real sequences of user’s transactions. Different tasks or different branches of the same task may alternate in execution and even linear tasks may be executed non-linearly: one task is broken by another task and restored again (occasional non-linearity).

TABLE II. NON-LINEARITIES IN LOGICAL STRUCTURE AND IN OBSERVED SEQUENCE OF EXECUTION

Type of non-linearity	Sequence of execution	Dependencies in logical structure	Dependencies in sequence of execution
Occasional (non-linear execution of linearly dependent queries)	<i>kitten food</i> — 1 <i>cat food</i> — 2 <i>icit</i> — 3 <i>icit 2009</i> — 4 <i>hills cat diet</i> — 5	linear chains: 1 → 2 → 5 3 → 4	
Branching (one query determines several queries)	<i>cat food</i> — 1 <i>kitten food</i> — 2 <i>home for cats</i> — 3 <i>home for dogs</i> — 4		
Merging (a query depends on a pair)	<i>kitten food</i> — 1 <i>cat food</i> — 2 <i>hills</i> — 3 <i>hills cat diet</i> — 4		
Re-merging (mergence of branched chains)	<i>cat food</i> — 1 <i>kitten food</i> — 2 <i>icit cats</i> — 3 <i>cat kitten icit</i> — 4		

Let distinct queries submitted by the same user during a time session be presented by nodes and dependencies (the

strongest dependencies according to a certain dependency measure, which will be set in a form of 2-step decision rule) between queries be presented by arcs. We will refer to this directed acyclic graph as a query modification graph. According to (2) a maximum indegree is equal to 2; according to (1) outdegrees are not limited. If there are two paths from the node R to the node M crossing only in R and M we speak about re-merging (merging of earlier branched chains) and call H a root of re-merging.

Table II shows synthetic examples of basic (branching, merging) and derivative non-linearities (re-merging and occasional non-linearity of execution) in a search process and its logical structure.

An applicable non-linear model (1)–(2) presents a general framework to describe a query modification process. There are a lot of technical variants of the model implementation. Here, we use several rules of structuring query modification process based on lexical dependency measures. Different kinds of dependency measures are sound and the best way to detect and to analyze a non-linear search is to use a family of decision rules, each of which may operate with different measures (e.g., combine them).

III. DATASET

A week sample (March, 2005) of the query log of the major Russian search engine Yandex is used. The dataset was pre-processed to exclude users who are robots rather than humans. To do it a client discriminator threshold equal to 7 unique queries per 1-hour sliding window was applied. 30-min intersession cutoff was used to segment user transactions into time sessions. The preprocessed dataset (Table III) contains 117,097 users executed 644,901 transactions. Time sessions containing query language operators (in particular quotations) were excluded to simplify processing.

TABLE III. PREPROCESSED DATASET

	Any results	Non-empty results
time sessions	225,451	213,154
time sessions containing only one query	138,757(61.55%)	136,354(63.97%)
time sessions containing 2+ distinct queries	86,694 (38.45%)	76,800(36.03%)
time sessions containing 3+ distinct queries	38,571 (17.11%)	32,091 (15.06%)
non-first queries in time sessions	156,101	132,418
non-first-two(non-last-two) queries in time sessions	66,926	53,720

The study is centered on essential query modifications rather on spelling corrections. Misprinted queries may lead to the empty list of the retrieved results, and in this case a user corrects misprints. On the contrary, if a search engine varies terms of misprinted query and retrieves correct results, a user doesn’t correct misprints. To avoid accounting of such a behavior we consider only those time sessions which do not

include queries retrieving empty results. The right column of Table III presents the analyzed data after exclusion of time sessions containing the empty-result queries. 13,856 time sessions were excluded, in particular, 6,065 users executed 6,616 only “empty-result” sessions.

IV. METHOD IN BRIEF

Each non-first query submitted during a time session is considered as possible dependent query, which may be determined either by a single query or a pair of queries submitted earlier during the session. Dependency on a pair is fixed only if this dependency is bigger than dependency on any single query. Different measures are used to select 1- and 2-dependencies. When a dependent query Q is attributed accordingly to used dependency measures, decision rules are applied to select a “main” determinant of Q among all of its determinants. A decision rule may use one or several 1- and 2-dependency measures. While a decision rule may combine different measures and use simultaneously 1- and 2-measures, we apply 2-step decision rules: (1) at the first step the main 1-determinant Q^{Det} is chosen as the most recent among queries which have the most influence on Q according some measure; (2) if the chosen 1-determinant Q^{Det} belongs to a pair (Q^{Det} , Q^{Det*}) mostly determining Q according to the measure used at this step (this measure may differ from the measure used at the first step to choose 1-determinant), the pair (Q^{Det} , Q^{Det*}) is selected as a final determinant; otherwise Q is determined by a 1-determinant Q^{Det} .

V. METHOD IN DETAIL

Extensions of traditional term-based query-to-query similarity metrics are used.

A. Query Kernel and Query Image

A *query image* $Im(Q)$ is an unordered set of *all* terms of a query Q . A *query kernel* $Ker(Q)$ is an unordered set of terms of Q belonging to one or several *role classes* (Table IV). Different role classes include different parts of speech. Unambiguous parts of speech tagging is not a problem in inflecting languages (e.g. Russian); and non-dictionary words are effectively attributed heuristically.¹

A query image is used to distinct different queries. As a result, queries *<big cat in the box>* and *<the box in big cat>* are “identical” since they have identical images.

A query kernel is used to detect inter-query dependency. If the intersection of kernels $Ker(Q1)$ and $Ker(Q2)$ is not empty, one of these queries may depend on the other. Otherwise queries are independent. Let a query kernel intersects with kernels of several queries submitted earlier and considered as possible determinants. We should choose the determining

query (or a pair of queries in the case of merging) mostly influencing it according to a dependency measure used.

TABLE IV. FOUR ROLE CLASSES OF PARTS OF SPEECH

Role Class	Parts of speech included into Class
<i>Objects/Subjects</i>	nouns, names, acronyms, +unknown words which may be one of them
<i>Features</i>	adjectives, participles, [+morphologically similar unknown words] numerals and numbers
<i>Actions</i>	verbs, adverbs, adverbial participles [+morphologically similar unknown words]
<i>Others</i>	all other words (prepositions, articles, etc)

B. Query Dependency Measures

Kernels and images are unordered sets of terms. Such a representation simplifies processing. At the same time, since we consider only queries submitted during a limited period it is extremely unlikely that two queries presented by the same unordered set of terms express different information needs.

Dependency measures. Term-based measures of query dependency are used. There are two broad classes of term-based dependency measures:

(1) (originally indirected) similarity metrics (intersection, symmetric difference and Jaccard metrics) augmented by a direction from the determining query (a pair of queries) to the dependent query;

(2) originally directed dependency measures: difference, narrowing, broadening.

Query dependencies. Each query may depend on queries submitted previously during a considered time session. Each query may determine queries submitted later during the session. All dependency measures are applied only to images of those queries, kernels of which are intersected. At the first step we check intersection of kernels $Ker(Q)=\{t_1, \dots, t_n\}$ and $Ker(Q^{Det})=\{T_1, \dots, T_m\}$ of a possible dependent query Q and a possible 1- or 2-determinant Q^{Det} . If $Ker(Q) \cap Ker(Q^{Det}) \neq \emptyset$, Q depends on Q^{Det} . If several determinants maximize dependency, we choose the recent determinant (see next subchapter).

Two types of dependency are used: dependency on a single query and dependency on a pair of queries considered as an entity. All measures may be used both as 1-dependency and 2-dependency measures. Besides, there is a special form of 2-relation, an empty symmetric difference (non-strict narrowing/broadening).

1-dependency. A query is concerned as a modification of a single query submitted previously in the same time session. A single determinant is called *1-determinant*.

Any 1-dependency measure of the determining query Q_k is presented by a vector M , where $m_{j \leftarrow k}$ is a score (“rank”) of the influence of the determining query $Q_{i \leftarrow k}$ on Q_k among all predecessors of Q_k . If Q_k does not depend on Q_i , m_j is not defined.

¹ Distribution of POS-based classes in queries: 74.20% of query terms belong to the *Subjects*, 14.45% to the *Features*, 3.55% to the *Actions* (“download” is the absolute leader) and 7.80% to the *Others* class among 928,731 terms in 340,615 queries retrieved non-empty results.

Example 1 (1-dependencies). Let $Ker(Q1)=\{A,B\}$, $Ker(Q2)=\{A\}$, $Ker(Q3)=\{B\}$ and $Ker(Q4)=\{B,C\}$ be kernels of the queries in Fig. 1. (To reduce illustrations we suppose that all queries in the example hereinafter include only kernel terms). Row dependency vectors for *overlap* (V^\cap), *Jaccard* ($V^{Jaccard}$), *symmetric difference* (V^Δ) measures are:

$$\begin{array}{lcl} & Q1 & Q2 & Q3 \\ V^\cap(Q2) = & \langle 1 \rangle & & \\ V^\cap(Q3) = & \langle 1 & . \rangle & \\ V^\cap(Q4) = & \langle 1 & . & 1 \rangle \\ V^{Jaccard}(Q2) = V^\Delta(Q2) = & \langle 1 \rangle & & \\ V^{Jaccard}(Q3) = V^\Delta(Q3) = & \langle 1 & . \rangle & \\ V^{Jaccard}(Q4) = V^\Delta(Q4) = & \langle 2 & . & 1 \rangle \end{array}$$

2-dependency (merging). A query is considered as a merge of a pair of previously submitted queries. The determinative pair is called *2-determinant*. While a model does not limit a number of branches, it prohibits merging of more than two queries. This restriction is psychologically motivated. Merging chains are executed in parallel but even execution of two chains in parallel is a hard job.

2-dependency of a query Q on a pair (Q^{Det1}, Q^{Det2}) is considered *only if* Q depends on *each* of these queries, i.e. *both* intersections $Ker(Q) \cap Ker(Q^{Det1})$ and $Ker(Q) \cap Ker(Q^{Det2})$ are not empty.

2-dependency of Q on a pair (Q^{Det1}, Q^{Det2}) is fixed if $m(Q, (Q^{Det1} \cup Q^{Det2}))$ is stronger than each of cognominal 1-dependencies $m(Q, Q^{Det1})$, $m(Q, Q^{Det2})$. As a result, we do detect $Q3$ as merging of $Q1$ and $Q2$ in series $Q1=\langle \text{big cat} \rangle$, $Q2=\langle \text{fat cat} \rangle$, $Q3=\langle \text{big fat cat} \rangle$ if kernels contain not only the *Subjects* class, and do not detect merging if kernels contain only terms belonging to the *Subjects* class.

It is possible that none of the elements of the pair is a top 1-determinant (see the next example). However, is possible only for very long queries and is uncommon.

Example 2 (2-dependencies). Now let's consider 2-dependency for session in Fig. 1. Let $Ker(Q1)=\{A,B,C\}$, $Ker(Q2)=\{D,E,F\}$, $Ker(Q3)=\{A,B,D,E\}$ and $Ker(Q4)=\{A,B,C,D,E,F\}$ (due to unusual length of $Q4$ this is an atypical example). The matrixes describing 2-dependency of $Q4$ are:

$$M^\cap(Q4) = M^{Jaccard}(Q4) = M^\Delta(Q4) = \begin{bmatrix} & Q1 & Q2 \\ Q2 & & 1 \\ Q3 & 2 & 2 \end{bmatrix}$$

(here the top 1-determinant $Q3$ is not part of the top 2-determinant $(Q1, Q2)$). The image of the pair determinant $(Q1, Q2)$ is identical to the image of the dependent query $Q4$.

C. Precedence of Determining Queries

We take into account human memory limitations and consider only queries submitted *during a time session* as a possible determinant of a query submitted later *in this session*. Only in-session dependencies are extracted. Let several determinants have equal maximum influence on the dependent query. Which of them should be chosen as the "major" determinant? We consider dependency of a query on queries submitted earlier. While dependencies between *queries* should be extracted, a search engine query log contains not queries but *transactions* which correspond either to *query submissions* or to *paging* the retrieved results (Fig. 1, where (Q, p) denotes a transaction with p -th page of the results retrieved by query Q).

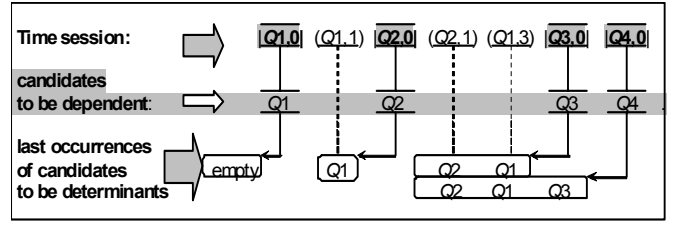


Fig. 1. Precedence vectors generation.

When a query Q is considered as a *dependent query* we consider dependencies only for the *first occurrence* of this query. When a query Q^{Det} is considered as a *determinant* of a query Q we take into account the *most recent* occurrence of Q^{Det} before the first occurrence of Q .

Dependencies of the same queries in different sessions may be different. For example, let $Q1=\langle \text{cat} \rangle$, $Q2=\langle \text{dog} \rangle$, $Q3=\langle \text{cat and dog} \rangle$ occur in 3 time sessions: $\langle Q1, Q3, Q2 \rangle$, $\langle Q1, Q2, Q3 \rangle$, $\langle Q3, Q1, Q2 \rangle$. $Q3$ is an element of a linear chain in the first session, a merge in the second and a root of branching in third.

The same query may occur several times during the session. We suppose that 1) when a query is considered as a possible depending query we should consider only *first* occurrence of the query; 2) when a query Q^{Det} is considered as a possible determinant of a query Q we should consider the recent occurrence of Q^{Det} before first occurrence of Q .

Precedence in 2-dependencies. When 1-dependency is considered there is no problem to detect the recent occurrence of a determinant (preceding the first occurrence of a dependent query) in the time-ordered series of transactions. To select the recent pair (preceding the first occurrence of a dependent query) among several determining pairs we use the following tricks:

Let (t_1, t_2) be the recent occurrences of elements of the pair, where t_i is presented either by absolute time of the recent (before a dependent query) occurrence of Q_i or by a rank of this occurrence in time-ordered scale of transactions (we use ranks.) A pair (Q_1, Q_2) is considered as more recent than (Q'_1, Q'_2) if $t_1 + t_2 > t'_1 + t'_2$. If several determining pairs $\{(Q'_1, Q'_2)\}$ maximize a sum $t'_1 + t'_2$ we choose a pair $i | t'_{1or2} = \max(\min(t'_1, t'_2))$, i.e. a pair the earlier element of which is the latest among earlier elements.

Example of precedence accounting. A precedence row vectors for (possibly) 1-dependent queries $Q2$, $Q3$ and $Q4$ (calculated for their first occurrences in a time session) in

$$\begin{array}{l} \text{Fig. 1 are} \\ \text{PrecV}(Q2) = \langle 1 \rangle \\ \text{PrecV}(Q3) = \langle 1 \quad 2 \rangle \\ \text{PrecV}(Q4) = \langle 2 \quad 3 \quad 1 \rangle \end{array}$$

and the precedence triangle matrixes (constricted on the precedence vectors) for 2-dependency are

$$\text{PrecMatr}(Q3) = \begin{bmatrix} & Q1 \\ Q2 & 1 \end{bmatrix} \quad \text{PrecMatr}(Q4) = \begin{bmatrix} & Q1 & Q2 \\ Q2 & 3 & 2 \\ Q3 & 1 & 2 \end{bmatrix}$$

D. Final Dependency Extraction

Decision rules select a “main” determinant of each dependent query among all of its determinants. A decision rule may use one or several 1- and 2-dependency measures. A decision rule operates with 1-dependency vectors, 2-dependency matrixes and a precedence vector.

Just one 1- or 2-determinant should be selected by the decision rule. However, 1-determinant may be not a part of 2-determinant even for the same measure (*Example 2*). To avoid a (prohibited) selection of 3 determining queries the decision rule should coordinate extraction of final 1- and 2-dependencies, which may be done in a variety of ways. Decision rules may simultaneously use different measures. While decision rules applied below are constructed as a sequential two-step procedure (1-dependency filter is used at the first step and 2-dependency filter is applied to the results of the first step) decision rules may have another form.

Example of the final determinant selection. Let's return to Example 1 and consider a family of decision rules each of which uses one of 1-dependency measures and selects for each dependent query the most recent among queries maximally influencing it according to the measure. The dependency structures extracted by these rules are presented by:

$$R^{\cap} = R^{\text{Jaccard}} = R^{\Delta} = \begin{bmatrix} Q1 & Q2 & Q3 \\ Q2 & 1 & \\ Q3 & 1 & \\ Q4 & & 1 \end{bmatrix} \Rightarrow Q1 <_{Q3 \rightarrow Q4}^{Q2}$$

VI. AUTOMATIC PROCESSING AND MANUAL CHECK

Let's consider the results yielded by three identically constructed 2-step decision rules. Each of rules is “monometric”, i.e. uses the same measure for 1- and 2-dependency determinants detection:

— **maximal overlap ($\max\cap$)** rule: the recent determinant *Det* (a single query Q^{Det} for 1-dependency or a pair (Q^{Det1} , Q^{Det2}) for 2-dependency) is chosen among queries which maximize *overlap* measures $Q \cap \text{Det}$;

— **minimal symmetric difference ($\min\Delta$)** rule: the recent among determinants which minimize *symmetric difference* measure is chosen.

— **Jaccard** rule: the recent among determinants which minimize *normalized symmetric difference* is chosen

Two types of query kernels are used:

— a kernel contains $\{\text{Subjects}+\text{Features}+\text{Actions}\}$ simulating commonly used procedures of similarity detection. All query terms are used to distinct queries and all non-*Others* terms are used to detect similarity

— a kernel contains only the *Subjects* role class.

It should be noted that the results of the same rules on so different kernel are very similar. The reason is not only an enormous fraction of *Subjects* terms in queries but collocation of other terms, i.e. if two queries belonging to the same time session contain the same subject these queries probably intersect in other terms.

When a decision procedure detects non-linear dependencies, two kinds of errors take place: false acceptance (Fig. 2) and false rejection. A brief manual check of the non-linearity detection shows that *all* non-linear query modifications correctly detected by $\min\Delta$ -based rules are a subset of non-linearities detected by $\max\cap$ -based rules. That is we should commit to the results of $\max\cap$ -based rules.

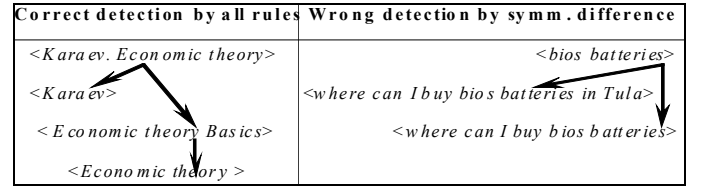


Fig. 2. Real examples of correct and incorrect detection.

Two-step decision rule is used. The results of 1-dependency extraction are used to extract 2-dependency. Namely, when the top 1-dependency determinant is detected the rule extracts the most recent and influencing determining pair (if exists), which contains the selected 1-determinant as an element. At the first step, the decision rules used in 2-step procedure are identical to the decision rules used in 1-dependency extraction. At the second step (selection of the pair) all 3 decision rules use the same *overlap* (2-dependency) measure, i.e. the determining pair is selected among the pairs which are most influencing according to this measure and contain the top 1-determinant extracted by the 1-dependency decision rule. Table V reports the results on the structure of query modification process detected by 2-step decisions rules.

A *merging search*: it is hardly probable that a merging results from the conscious task disaggregation (cf. [12]). The most mergings are *re-mergings*, which typically have the very special form of a merging a branch and the *initial* query. It is likely that the initial query perfectly presents the user *need* but the retrieved results are bad. A user modifies the query and when the results become good enough he “partly returns” to the initial query. A user does remember the need, not initial query.

TABLE V. CHARACTERISTICS OF SEARCH STRUCTURES EXTRACTED BY 1-&2-DEPENDENCY RULES

Kernel = Subjects			
	$\max\cap$	$\min\Delta$	Jaccard
Time sessions (% **) containing:			
only linearly executed chains	79.40%	70.17%	71.43%
non-linearly executed linear chains	6.54%	5.92%	6.12%
branchings	11.30%	20.73%	20.54%
mergings	5.87%	5.85%	4.39%
branchings & mergings	3.63%	3.88%	3.04%
re-mergings	3.21%	3.18%	2.45%

Queries:			
independent queries (% of non-first queries)	44.50%		
dependent on non-direct predecessor †	13.66%	21.00%	20.55%
roots of branching ‡	7.08%	13.04%	12.91%
direct sons of branching †	14.65%	27.68%	27.44%
direct sons per root of branching	2.07	2.12	2.13
mergings ‡	3.69%	3.67%	2.70%
Length of linear chains in:			
linear chains in LS	1.33	1.24	1.24
linearly executed linear chains in NLELS	1.16	1.12	1.14
non-linearly executed linear chains in NLELS	1.45	1.32	1.33
linearly executed linear chains in NLS	1.14	1.12	1.12
non-linearly executed linear chains in NLS	1.21	1.19	1.19
non-empty prebranching lin.chains in branching and rate of empty pre-branchings	1.26 68.1%	1.21 80.6%	1.20 80.3%
non-empty pre-branch. lin.chains in re-merging and rate of empty pre-branchings in remerging	1.28 67.3%	1.13 78.8%	1.13 80.9%
non-empty postmerging lin.chain (non-remerging) and rate of empty postmergings (non-remerging)	1.20 75.1%	1.13 81.0%	1.19 84.7%
non-empty post-REmerging linear chain and rate of empty postmergings (non-remerging)	1.24 74.2%	1.17 83.4%	1.24 84.6%
Length of first & last branches (non-remerging)	1.18/1.28	1.12/1.23	1.12/1.25
Length of min & max branches (non-remerging)	1.03/1.43	1.02/1.34	1.03/1.36
Length of first & last merging chains (non-remerg)	1.32/1.17	1.29/1.15	1.20/1.13
Length of min & max merging chains (non-remerg)	1.04/1.45	1.03/1.41	1.02/1.31

Notations

LS — linearly executed Linear time Session,
NLELS — Non-Linearly Executed Linear time Session,

NLS — logically Non-Linear time Session

* All lengths are measured in number of query modifications $Q^{(i)} \rightarrow Q^{(i+1)}$.

** % of time sessions containing 3+ distinct queries.

† Rate among non-first-two queries /or (‡) among non-last-two/.

Kernel = <i>Subjects+Actions+Features</i>			
	<i>max</i> ∩	<i>min</i> Δ	<i>Jaccard</i>
Time sessions (%) containing:			
only linearly executed chains	78.55%	69.31%	70.27%
non-linearly executed linear chains	6.32%	5.72%	6.02%
branchings	12.09%	21.47%	21.67%
mergings	6.32%	6.31%	4.76%
branchings & mergings	3.83%	4.13%	3.28%
re-mergings	3.39%	3.37%	2.64%

Queries:			
independent queries (% of non-first queries)	42.26%		
dependent on non-direct predecessor †	14.31%	21.59%	21.47%
roots of branching ‡	7.59%	13.53%	13.64%
direct sons of branching †	15.63%	28.57%	29.00%
direct sons per root of branching	2.06	2.11	2.13
mergings ‡	3.98%	4.06%	2.93%
Length of linear chains in:			
linear chains in LS	1.34	1.25	1.25
linearly executed linear chains in NLELS	1.11	1.14	1.14
non-linearly executed linear chains in NLELS	1.46	1.32	1.34
linearly executed linear chains in NLS	1.21	1.15	1.15
non-linearly executed linear chains in NLS	1.50	1.29	1.22
non-empty prebranching lin.chains in branching and rate of empty pre-branchings	1.27 68.5%	1.25 79.2%	1.21 79.4%
non-empty pre-branch. lin.chains in re-merging and rate of empty pre-branchings in remerging	1.21 69.3%	1.17 78.4%	1.14 80.5%
non-empty postmerging lin.chain (non-remerging) and rate of empty postmergings (non-remerging)	1.28 72.3%	1.10 80.9%	1.20 84.6%
non-empty post-REmerging linear chain and rate of empty postmergings (non-remerging)	1.35 70.3%	1.22 81.5%	1.26 84.3%
Length of first & last branches (non-remerging)	1.17/1.34	1.12/1.26	1.13/1.26
Length of min & max branches (non-remerging)	1.03/1.47	1.02/1.36	1.03/1.37
Length of first & last merging chains (non-remerg)	1.39/1.18	1.36/1.13	1.23/1.12
Length of min & max merging chains (non-remerg)	1.03/1.44	1.03/1.44	1.02/1.32

TABLE VI. SHORTEST PATH IS EXECUTED EARLIER THAN LONGEST PATH IN BRANCHINGS AND MERGINGS:

Kernel = <i>Subjects</i>			
	<i>max</i> ∩	<i>min</i> Δ	<i>Jaccard</i>
non-remerging branchings	62.2%	67.2%	68.6%
non-remerging mergings	33.6%	32.9%	38.2%
Kernel = <i>Subjects+Actions+Features</i>			
	<i>max</i> ∩	<i>min</i> Δ	<i>Jaccard</i>
non-remerging branchings	62.6%	67.6%	68.7%
non-remerging mergings	30.2%	29.9%	34.7%

Precedence of different-length paths in non-linearities.

The results presented in Table VI correspond to expectations about a probable searching behavior. The later branches in non-remerging branching correspond to the “last hope” attempts and have more chances to be longer. On the contrary, the final

of two merging paths have more chances to be shorter since (1) a searcher should not forget the current pre-merging state in the first path and (2) a goal of the final path is to “till” results of the first.

VII. CONCLUSIONS

A model presented in the paper allows for query dependencies on non-direct predecessors. As a result, a query modification process may be branching. Dependency on a pair of queries considered as an entity is allowed too. As a result, a process may be merging and re-merging (as a merging of branched chains). It is shown that non-linear dependencies between user's queries are detected as really frequent whatever the technique used. The results are:

(1) a branching search is a frequent manner of an execution of several-query tasks (about 12% of such time sessions contain branchings);

(2) a number of branches is a little bigger than two (about 2.06);

(3) one of the branches consists of only one query;

(4) the first of [usually two] branches is usually shorter than second;

(5) the first of [perfectly two] merging paths in s usually longer than second.

While the branched search is partly supported by search environments, the merging (in particular, re-merging) search is not supported at all. However,

(6) about 6% of time sessions containing 3+ distinct queries contain mergings, and

(7) the merging search is frequently (3-4% of time sessions containing 3+ distinct queries) executed in the re-merging manner, particularly as combining the initial query (i.e. the root of branching) perfectly presenting the information need (but retrieving unsatisfactory results) and the current query modification.

ACKNOWLEDGEMENTS

The author feels obliged to thank Yuri Zelenkov, Amanda Spink, Elena Sheychenko and Izzat Alsmadi for the helpful remarks.

REFERENCES

- [1] M. Bates, “Information search tactics,” *J. of American Soc. for Inf. Sci.*, 30(4): 1979, pp. 205–214.
- [2] M. Bates, “Idea tactics,” *Journal of American Soc. for Inf. Sci.*, 30(5): 1979, pp. 280–289.
- [3] N. Buzikashvili, “The Yandex study,” *Workshop on Evaluating User Studies in Inf. Access* (Glasgow, UK, June 2005), *Uni. of Strathclyde, British Computer Society*, pp. 48–55.
- [4] N. Buzikashvili, “Structure of query modification process: branchings,” *ADMA'08* (Chengdu, China, Oct. 2008). *LNAI*, vol. 6139, Springer, 2008, pp. 717–724.
- [5] N. Buzikashvili, “Non-Linear Query Reformulation Behavior,” *DIR-2009* (Enschede, Netherlands, 2009), *Uni. Twente*, 2009, pp. 42–50.
- [6] R. Fidel, “Moves in online searching,” *Online Review*, 9(1): 1985, pp. 61–74.
- [7] R., Jones, B. Rey, O. Madani, and W. Greiner, “Generating query substitutions,” *WWW 2006* (Edinburgh, Scotland, May 2006), *ACM Press*, 2006, pp. 387–396.
- [8] F. Radlinski, and T. Joachims, “Query Chains: Learning to rank from implicit feedback,” *ACM KDD'05*, *ACM Press*, NY, 2005.
- [9] S.Yu. Rieh, and H.I. Xie, “Analysis of multiple query reformulations on the web: The interactive information retrieval context,” *Inf. Proc. & Manag.*, 42: 2006, pp. 751–768.
- [10] P. Vakkari, “eCognition and changes of search terms and tactics during task performance,” *RIAO'2000*, 2000, pp. 894–907.
- [11] M. Whittle, B. Eaglestone, N. Ford, V.J. Gillet, and A. Madden, “Data mining of search engine logs,” *J. of American Soc. for Inf. Sci. and Technology*, 58 (14), 2007, pp. 2382–2400.
- [12] B. Wildemuth, E. Jacob, A. Fullington, R. de Blieck, and Ch. Friedman, “A detailed analysis of end-user search behaviors,” *54th ASIS Annual Meeting*, *ACM*, 1991, pp. 302–312.