

POLITECNICO DI MILANO
Facoltà di Ingegneria dell'Informazione



POLO REGIONALE DI COMO

Diversification for Multi-Domain Result-sets

Supervisor: Prof. Piero Fraternali
Assistant Supervisor: Alessandro Bozzon

Master Graduation Thesis by:
Michele Follo,
Student Id. number 739480
Andrea Vaccarella,
Student Id. number 735123

Academic Year 2009/2010

Abstract

Recent years witnessed a slow but steady trend toward a more elaborated usage for Web search engines, characterized by a switch from a document search interaction paradigm to an entity search one.

The objective of the Search Computing project is the definition of methods and tools supporting multi-domain search, an entity search paradigm working on domain-specific search engines, whose results are aggregated to create a unique answer covering multiple domains. Due to the combinatorial nature of multi-domain search, the number of combinations in the result set is normally very high, and strongly relevant objects tend to combine repeatedly with many other concepts, requiring the user to scroll down the list of results deeply to see alternative, maybe only slightly less relevant, objects. Improving the diversity of the result set is the aim of *diversification*, which can be defined in the context of multi-domain search as the selection of k elements out of a universe of N combinations, so to maximize a quality criterion that combines the *relevance* and the *diversity* of the objects of distinct types seen by the user, thus improving his information-seeking experience.

This thesis deals with the problem of diversification for multi-domain search, offering: i) a characterization and a formalization of the problem; ii) a comparative study on major information retrieval diversification approaches to test their applicability in this new context; iii) a quantitative evaluation of the performance of four state-of-the-art diversification algorithms, using adaptations of the evaluation metrics currently adopted in the context of diversification for Web documents; iv) a qualitative evaluation of the perception and utility of diversification in multi-domain search with two user studies. Results provide evidences that the usage of diversification techniques in the multi-domain context is worthwhile and effective, improving both the quality and the perceived utility of multi-domain query results.

Sommario

Nel corso degli ultimi anni si è assistito ad un importante cambiamento nelle modalità con cui gli utenti usufruiscono dei servizi di ricerca Web: da un paradigma in cui l'obiettivo era l'individuazione di singoli documenti (eseguita attraverso motori di ricerca come Google o Bing), si assiste ora ad una prevalenza di attività di esplorazione ed estrazione di dati, molto spesso complessi e aggregati, in cui i servizi di ricerca sono uno strumento al servizio dell'esecuzione di attività più complesse.

In tale contesto si inserisce l'attività di ricerca del progetto Search Computing (SeCo), il cui obiettivo è la definizione di metodologie e strumenti a supporto della ricerca multi-dominio, ovvero un tipo di ricerca Web effettuato su motori di ricerca e sistemi di raccolta dati specializzati, i cui risultati vengono integrati al fine di offrire all'utente una risposta univoca che copra approfonditamente molteplici contesti.

L'aggregazione dei risultati forniti da differenti servizi web, provenienti da domini di ricerca molteplici e spesso non in relazione tra loro, pone i sistemi di ricerca multi-dominio di fronte al problema di possibili uguaglianze nei dati restituiti da fonti diverse. La diversificazione dei dati comporta un miglioramento nella qualità dei risultati, limitando queste ripetizioni e garantendo una maggiore soddisfazione per l'utente finale dovuta ad una più ampia copertura dei possibili contesti a cui egli può essere interessato.

La nostra tesi si pone come obiettivo quello di indagare il problema della diversificazione dei risultati nelle ricerche web, proponendo uno studio comparativo dei maggiori algoritmi noti nel dominio dell'Information Retrieval, adattandoli al contesto della ricerca e manipolazione di dati multi-dominio.

In particolare, il lavoro di tesi ha avuto come scopo lo studio dello stato dell'arte delle principali metriche e tecniche di diversificazione,

l'adattamento al contesto di ricerca multi-dominio su dati strutturati degli algoritmi noti, il design di test quantitativi al fine di validarne l'efficacia teorica, e uno studio qualitativo per verificare l'effettivo incremento della qualità dei risultati nella percezione utente.

Contents

Abstract	II
Sommario	III
1 Introduction	2
1.1 Original Contributions	4
1.2 Dissertation Organization	5
2 Related Work	6
2.1 The SeCo Framework	6
2.1.1 Service Marts	7
2.1.2 Liquid Query	9
2.1.3 Ranked mashups of uncertain information . . .	11
2.2 Query Disambiguation	12
2.2.1 Diversification for keyword over structured data	13
2.2.2 Keyword refinement using diversity	14
2.2.3 Ambiguous query identification	15
2.2.4 Sub-query generation using query logs	16
2.2.5 Query decomposition	16
2.2.6 Query diversification through Wikipedia . . .	17
2.3 Diversification over unstructured data	18
2.3.1 Intent-aware document diversification	18
2.3.2 Maximal marginal relevance	20
2.3.3 Information nugget model	21
2.4 Diversification over structured documents	23
2.4.1 Desiderata, limitations and optimality	23
2.4.2 Trees of diversity	25
2.4.3 User preferences over relational databases . .	28
2.4.4 User preferences, items competition and trade-	
offs	29
2.5 Related Works conclusion	29

3	Framework	31
3.1	Problem formalization	31
3.2	Relevance	32
3.3	Diversity	32
3.3.1	Categorical diversity	33
3.3.2	Quantitative diversity	33
3.4	Algorithms	34
3.4.1	MaxSum	34
3.4.2	MaxMin	36
3.4.3	MMR	37
3.4.4	MaxCov	37
3.5	Datasets	38
3.5.1	Milan set	39
3.5.2	University set	42
4	Quantitative test	46
4.1	Metrics	46
4.1.1	α -DCG	47
4.1.2	MD-Recall	49
4.2	Design	50
4.3	Results	52
4.3.1	Results for α -DCG	52
4.3.2	Results for MD-Recall	57
4.3.3	Comments	61
5	Qualitative Test	63
5.1	Design	64
5.1.1	User Test 1	64
5.1.2	User Test 2	69
5.2	Results	79
5.2.1	Experiment 1 - Direct comparison	80
5.2.2	Experiment 2 - Best three combinations	83
5.2.3	Comments	87
6	Conclusion and Future Work	89
6.1	Future Work	90
A	Quantitative test charts	97

List of Figures

2.1	A representation of different Service Marts	7
3.1	Structure of Milan set combinations with attributes and their type.	40
3.2	Structure of University set combinations with attributes and their type.	43
4.1	α -DCG ($\alpha = 0.5$), categorial distance, giving same weight to relevance and diversity.	53
4.2	α -DCG ($\alpha = 0.5$), quantitative distance, giving same weight to relevance and diversity.	54
4.3	First 50 samples of α -DCG, categorial distance, giving same weight to relevance and diversity.	55
4.4	First 50 samples of α -DCG, quantitative distance, giving same weight to relevance and diversity.	56
4.5	MD-Recall, categorial distance, giving same weight to relevance and diversity.	58
4.6	MD-Recall, quantitative distance, giving same weight to relevance and diversity.	59
4.7	First 50 samples of MD-Recall, categorial distance, giving same weight to relevance and diversity.	60
4.8	First 50 samples of MD-Recall, quantitative distance, giving same weight to relevance and diversity.	61
5.1	Welcome page for the first user test	65
5.2	Introductory page for the first user test	66
5.3	First scenario presentation and tabular view of the first set	66
5.4	Second set in the comparison for the first scenario	67
5.5	Screenshot of the details available for each item in the list	67
5.6	Comparison of the two sets for the Study Abroad scenario	68

5.7	A simple thanking page to greet the attending users .	68
5.8	Users need to identify themselves to proceed	70
5.9	A short introductory page was presented to explain the test and the context	70
5.10	Users were asked to fill a short demographic survey .	72
5.11	Introduction to the Rome tutorial	73
5.12	View of the selection page for the tutorial	73
5.13	Warning message for managing exceptions	74
5.14	Introductory page for the first task (Milan experience)	74
5.15	Detailed view of selected combination	75
5.16	View of the questionnaire after Milan experience . . .	75
5.17	Example of a University experience task web page . .	76
5.18	Detailed view for a combination of the University set	77
5.19	Introductory step for the final questionnaire	77
5.20	Final questionnaire	78
5.21	Thanking page to greet attending users	78
5.22	User votes in the two scenarios	80
5.23	User preferences in the Study Abroad scenario . . .	81
5.24	User preferences in the Night Out scenario	82
5.25	User preferences in the Night Out scenario	84
5.26	Duration of the test for the Study Abroad scenario .	85
5.27	Number of clicks for the Study Abroad scenario . . .	86
5.28	Visited pages comparison between scenarios	86
5.29	Duration and clicks for the user test compared	87
A.1	α -DCG for $\lambda=0$, categorial diversity	98
A.2	α -DCG for $\lambda=0.25$, categorial diversity	99
A.3	α -DCG for $\lambda=0.5$, categorial diversity	100
A.4	α -DCG for $\lambda=0.75$, categorial diversity	101
A.5	α -DCG for $\lambda=1$, categorial diversity	102
A.6	α -DCG for $\lambda=0$, quantitative diversity	103
A.7	α -DCG for $\lambda=0.25$, quantitative diversity	104
A.8	α -DCG for $\lambda=0.5$, quantitative diversity	105
A.9	α -DCG for $\lambda=0.75$, quantitative diversity	106
A.10	α -DCG for $\lambda=0$, quantitative diversity	107
A.11	MD-Recall, categorial diversity	108
A.12	MD-Recall, quantitative diversity	109

Chapter 1

Introduction

Recent years witnessed a slow but steady trend toward a more elaborated usage for Web search engines, characterized by a paradigmatic switch from document search to object (or entity) search [7][10].

On one hand, search services have become more and more specialized, defining specific domains and contexts; on the other hand, users are searching for more complex information, which require multiple sources. These data are scattered through the Web, and the services able to retrieve them are not meant for mutual interactions.

The research activities of the Search Computing (SeCo) project take place in this context. The objective of SeCo is the definition of methods and tools supporting multi-domain search, a Web search paradigm working on domain-specific search engines services, whose results are aggregated and integrated in order to offer to users a unique answer covering multiple domains. For instance, when dealing with complex queries like *"find the best museum to visit, then a restaurant to eat and then a nice hotel to spend the night in"*, even if all the atomic information are available on the Web, the correct answer is a data combination, achievable only querying all services and composing their answers.

Formally, multi-domain queries can be represented as rank-join queries over a set of relations, representing the wrapped data sources [17][23][32]. Each item in the result set is a combination of objects that satisfy the join and selection conditions, and the result set is ranked according to a scoring function, which can be expressed as a combination of local relevance criteria formulated on objects or associations (e.g., price or rating for a hotel, distance between the conference

	<i>Hotel</i>		<i>Restaurant</i>		<i>Museum</i>		$S^{(a)}(\tau, q)$
	HName	HPrice	RName	RPrice	MName	MPrice	Total price
τ_1	Hotel Amadeus	€35	Miyako	€25	Galleria d'Arte Moderna	€0	€60
τ_2	Hotel Amadeus	€35	Miyako	€25	Museo Civico di Milano	€0	€60
τ_3	Hotel Amadeus	€35	Miyako	€25	Museo di Storia Contemporanea	€0	€60
τ_4	Hotel Amadeus	€35	Porca Vacca	€25	Galleria d'Arte Moderna	€0	€60
τ_5	Hotel Amadeus	€35	Porca Vacca	€25	Museo Civico di Milano	€0	€60
τ_6	Hotel Amadeus	€35	Porca Vacca	€25	Orto Botanico di Brera	€0	€60
τ_7	Hotel Amadeus	€35	Spontini 6	€25	Galleria d'Arte Moderna	€0	€60
τ_8	Hotel Amadeus	€35	Spontini 6	€25	Museo Civico di Milano	€0	€60
τ_9	Hotel Amadeus	€35	Spontini 6	€25	Orto Botanico di Brera	€0	€60
τ_{10}	Hotel Amadeus	€35	Spontini 6	€25	Museo di Storia Contemporanea	€0	€60

Table 1.1: Top- k result set based on relevance.

venue, hotel, and restaurant).

Due to the combinatorial nature of multi-domain search, the number of combinations in the result set is normally very high, and strongly relevant objects tend to combine repeatedly with many other concepts, requiring the user to scroll down the list of results deeply to see alternative, maybe only slightly less relevant, objects.

Let assume that the first searching system returns 20 hotels and 20 restaurants, and the second does the same for its objects (20 restaurants and 20 museums). Whenever the same restaurant appears in both sets, and a cartesian product is used to aggregate results, the composition of the terms (museum-restaurant-hotel) will lead to a repetition in the result set. For illustration, Table 1.1 shows an example result set, which contains the top-10 combinations ranked according to total price. We observe that the result is rather poor in terms of diversity, as only 1 hotel, 3 restaurants and 4 museums are represented. Indeed, the number of distinct objects that appear in the top- k results is sensitive to the distribution of attribute values used to compute the score of the combination.

Hence, budget hotels will appear repeatedly in the top- k list, lowering the number of distinct objects seen by the user. The same observation applies when, fixed a hotel, one considers the price range of restaurants compared to the price range of museums.

Improving the diversity of the result set is the aim of *diversification*, which can be defined in the context of multi-domain search as the selection of k elements out of a universe of N combinations, so to

	<i>Hotel</i>		<i>Restaurant</i>		<i>Museum</i>		$S^{(a)}(\tau, q)$
	HName	HPrice	RName	RPrice	MName	MPrice	Total price
τ_1	Hotel Amadeus	€35	Miyako	€25	Galleria d'Arte Moderna	€0	€60
τ_2	Hotel Amadeus	€35	Porca Vacca	€25	Museo di Storia Contemporanea	€0	€60
τ_3	Hotel Amadeus	€35	Miyako	€25	Orto Botanico di Brera	€0	€60
τ_4	Hotel Delle Nazioni	€36	Miyako	€25	Galleria d'Arte Moderna	€0	€61
τ_5	Hotel Delle Nazioni	€36	The Dhaba	€25	Orto Botanico di Brera	€0	€61
τ_6	Hotel Delle Nazioni	€36	Spontini 6	€25	Pad. d'Arte Contemporanea	€2	€63
τ_7	Hotel Zefiro	€39	Matto di Bacco	€25	Galleria d'Arte Moderna	€0	€64
τ_8	Hotel Zefiro	€39	Porca Vacca	€25	Museo Civico di Milano	€0	€64
τ_9	Hotel Delle Nazioni	€36	Porca Vacca	€25	Museo della Permanente	€6	€67
τ_{10}	Hotel Zefiro	€39	Miyako	€25	Museo Civico di Storia Naturale	€3	€67

Table 1.2: Top- k result set based on relevance and diversity.

maximize a quality criterion that combines the *relevance* and the *diversity* of the objects of distinct types seen by the user. In this respect, Table 1.2 shows an example of result set with diversified combinations. We observe that the set does not necessarily contain the top-10 combinations in terms of total price. Nevertheless, the result is much richer: 3 hotels, 5 restaurants and 7 museums are selected.

1.1 Original Contributions

The aim of this thesis is to analyze existing Information Retrieval diversification approaches to test their applicability in the context of multi-domain search. Our original contribution can be summarized as follows:

- We characterize the result set diversification problem in multi-domain query, which implies the need to study and eventually adapt state-of-the-art algorithms for structured and unstructured Web data.
- We formalize the concept of multi-domain result distance and propose two kind of diversity: one based on object equality (*categorical*) and the other based on attribute distances (*quantitative*).
- Since diversification can be shown to be NP-hard also in the multi-domain context, we study the behavior of four known

greedy algorithms experimentally. Specifically, we test the hypothesis that the diversification algorithms improve the quality of the result set with respect to a baseline constituted by the selection of the most relevant k combinations. The performances of the four approaches are then evaluated using information retrieval metrics suitable for the multi-domain context.

- We evaluate the perception and utility of diversification in multi-domain search with two user studies based on selection tasks involving combinations of correlated objects.

1.2 Dissertation Organization

The rest of the thesis is organized as follows:

- **Chapter 2** defines current approaches toward diversification and the main characteristics of the Search-Computing project with respect to our work.
- **Chapter 3** presents the framework, from a threefold perspective: first, the chapter describes four state-of-the-art diversification algorithms; then, it formalizes specific distance metrics for multi-domain result sets and, to conclude, it elaborates on the result sets created for testing purposes.
- **Chapter 4** describes the design of the quantitative tests and their results.
- **Chapter 5** describes the design of the user studies and comments their results.
- **Chapter 6** conclusions are drawn and future work directions are presented.
- In **Appendix A** we present graphs from main results of the quantitative analysis described in Chapter 4.

Chapter 2

Related Work

The purpose of this chapter is to contextualize our work with respect to academic literature and industrial practices for result diversification and data disambiguation. We also introduce *Search-Computing* (SeCo) and its possible relation with diversification.

The chapter is divided as following: section 2.1 contains some of the key aspects of Search-Computing and the possible roles of diversification, section 2.2 discusses the problem from a query perspective, section 2.3 and section 2.4 analyze how diversification deals with unstructured and structured documents, and finally section 2.5 summarizes main approaches from other fields.

2.1 The SeCo Framework

As stated in chapter 1, the goal of the Search-Computing project is to build a system capable of finding answers to complex searches through the interaction of multiple, cooperating, services. One of the crucial aspect in the result composition step is the use of different services, whose answers have to be ranked, joined, and invoked in a proper order to satisfy the user searching needs. The aggregation of different services (and their results) deals with the concept of *Service Marts*, and the diversification on items managed by the SeCo system cannot prescind from its definition. The first section (2.1.1) is in charge of explaining the basic concepts of Service Marts, underlying strengths and limitations. The second section (2.1.2) explains the concept of how the query is formulated by users through the *Liquid Query* paradigm, and how results of this query interacts each another. The last section (2.1.3) is intended to expand

the problem of investigating information under uncertainties over unstructured sources.

2.1.1 Service Marts

Service Marts [5] are interfaces for *Web objects* that hide the underlying data source structures, presenting them in terms of inputs, outputs, and rank attributes. Attributes may have multiple values and be clustered within repeating groups. When objects are accessed through Service Marts, the responses are ranked lists, presented divided in chunks, so as to avoid receiving too many objects at one time. Search Computing is a new paradigm for composing search services [7] and the composition of returned chunks has a direct connection with data diversification.

Service Marts:

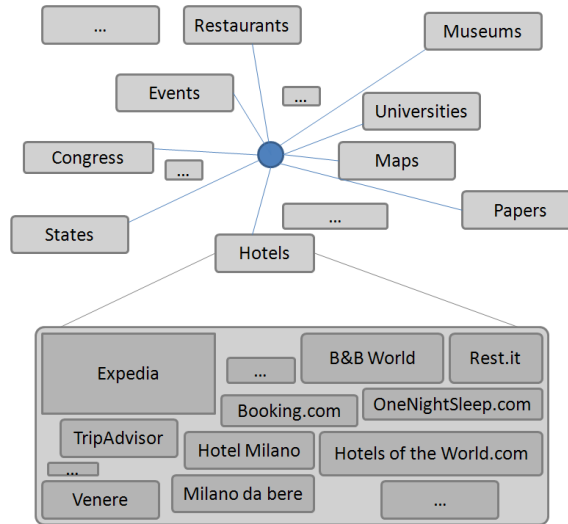


Figure 2.1: A representation of different Service Marts

Modern search systems answer generic or domain-specific queries, Search Computing enables answering queries via a constellation of cooperating search services, correlated by means of join operations. While the SOA (Service Oriented Architecture) principles are becoming widespread, distinct standards, languages, and programming styles are proposing themselves as the best options. The definition of Service Mart as abstractions for data source publication and composition, has the goal of ease the linking of search services [29]. Every Service Mart is mapped to one or more Web object available on Internet, but its related to only one "category" (in the "travel"

domain there will be a service mart responsible for the flights, one for the hotels, one for the restaurants and so on). Service Marts are augmented with connections, so as to support their linking to the system and between returned objects. Service Marts are abstractions, and publishing a Service Mart means bridging an abstract description to several concrete implementations of the service. We can distinguish three levels of description for a service mart: *conceptual*, *logical*, and *physical*.

Conceptual level: a Service Mart is an abstraction describing a class of Web objects.

Every Service Mart definition includes a name and a set of exposed attributes (atomic or repeating groups). Attributes and sub-attributes are typed and semantically tagged when they are defined. In this way it is possible to provide more expressive query interpretations and model 1:M or N:M relationships.

Logical level: each Service Mart is associated with one or more specific Access Patterns, describing how the Service Mart can be addressed. It is a specific signature of the Service Mart with the characterization of each attribute as input (I) or output (O). This allows a connection between different service marts, using the outputs of one service as inputs for another one. This Connection patterns (the conjunction of two predicate expressions) is identified with a conceptual name and a logical specification.

Physical level: Each Service Mart has service interfaces mapped to concrete data sources. These interfaces ensure the possibility of linking different Marts together, reducing the cases where no connections can be created. The lack of an attribute or of an attribute property is often balanced by these interfaces, able to highlight commonalities and gaps.

The use of metadata (*descriptors* in SeCo), is an approach used by Liu [20] to speed up diversification with structured information as we will explain in section 2.4.1. Here is a short list of the main descriptors in the Search-Computing and their use.

- *Ranking descriptors* classify the service interface as a search service (producing ranked lists by themselves) or an exact service
- *Chunk descriptors* deal with output production by a service interface. The service is chunked when it can be repeatedly invoked and at each invocation a new set of objects is returned.

- *Cache descriptors* to speed up the retrieval of results when repeated invocations of a service are involved
- *Cost descriptors* deal with associating each service call with a cost characterization (response time, monetary cost,...)

In conclusion: Service Marts is an interoperability concept for building Search Computing applications, with associated technologies for registering and adapting Web services. The Web world is described as a resource graph with Service Marts linked by connection patterns, and then Service Marts are associated with service interfaces and implementations. When different Service Mart results have to be aggregated, there is an actual risk to have a great number of repetitions. This is avoidable in two ways: through the use of specific calls which minimize repetitive results (query which probabilistically minimize overlapping items) or diversifying the results. Our approach focuses on the second solution since it doesn't involve a prior knowledge of all the possible query calls and the entire result set of each Service Mart, nor probabilistic or statistical information on the distribution functions of the item population. Diversification in fact reduces, according to user preferences, possible overlapping items directly in the result set, dealing with a limited number of Service Marts and objects (only the one requested by the user query).

2.1.2 Liquid Query

SeCo operates under the *Liquid Query* paradigm [4], a user interaction paradigm that helps users to find responses to multi-domain queries through information exploration on Service Marts.

User needs cannot always be satisfied by individual pages on the surface Web or with single calls to a search service. Information seeking is a process-intensive task often related to *exploratory search*, defined as the situation in which the user starts from a not-so-well-defined information need and progressively discovers more. It contains high level of uncertainty. The focus of Liquid Query is on the support of multi-domain queries, using many domain-specific search engines to cover a multitude of different domains. Nowadays web services offer the possibility of building more ambitious systems, based upon search service orchestration. Diversification is highly related to the way results are presented , and therefore is connected to service orchestration.

The Liquid Query interface consists of interaction primitives that let users pose questions and explore results, spanning over multiple sources, incrementally. Those sources can have different nature (structured, unstructured), can belong to different domains (hotels, restaurants, museums...) and respond differently when invoked (single item, chunks, multiple items).

Diversification plays a key role to maximize user satisfaction when different services (or different means for a single service) are implied.

The Liquid Query interface presents to the users composite answers, obtained by aggregating search results from various services. Result composition is achieved through the use of *join*, a classic operation of data management. Exploiting the structural information of search service interfaces it is possible to wrap the single, domain-specific, data sources together, bridging on identical or similar attributes.

Once results are composed and displayed, the user has many possibilities:

- expand the query with an extra search service,
- add or drop attributes of an object
- ask for more results from a specific service
- aggregates results
- reorder results
- adding details (drill-down) or removing them (roll-up)
- choose the best data visualization format

Initial seed query can be expanded by the user in various directions, inspecting new related objects, according to the exploratory search paradigm. Due to the search service federation approach, elements in the result set of Liquid Query are object combinations, and each of them is associated with the search service that has returned it. They can be manipulated individually using features of the specific service, or in groups as a single combination, using composite features (which weren't available as single items). Since the number of aggregated results grow exponentially with the number of service invoked, diversification become essential to avoid repetitions and maximize users satisfaction.

The query life cycle is now presented:

1. Application configuration phase
2. Query submission phase
3. Query execution phase
4. Result browsing phase

Diversification mainly regards the last phase.

The unawareness of the data repository and primitives specifications used by the query doesn't affect data-differentiation. Diversification reduces the repetitiveness of the results, and encourages the service repository enrichment, since it will increase diversity in the results set and improve its overall quality!

2.1.3 Ranked mashups of uncertain information

The merging of different sources (Service Marts outputs) is inevitable to obtain a good multi-domain coverability of topics. *Mashups* [32] are situational applications that join multiple sources to better meet the information needs of Web users. MashRank is a mashup authoring and processing system built on concepts from rank-aware processing, probabilistic databases, and information extraction to enable ranked mashups of unstructured sources. To handle uncertain preference scores, represented as intervals enclosed in possible score values, it uses new semantics, formulations and processing techniques.

MashRank integrates information extraction with query processing by asynchronously pushing extracted data on-the-fly into pipelined rank-aware query plans, and using ranking early-out requirements to limit extraction cost. SeCo adopts similar procedures within its scheduler and query planner components. Mashups at Web scale triggers the need to rank large volumes of data, and traditionally ranking queries compute the top-K results on a given scoring function. Those results are the ones that we already explained need to be diversified and disambiguated. Moreover, the central idea of rank join is to allow for early query termination by making use of sorted inputs and scoring function monotonicity to upper bound the scores of non-materialized join results. In this situation, diversification would also be possible on partial results, inputs of the rank-join instead of diversifying its outputs. Unfortunately, due to the low predictability of partial results and due to the reciprocal unawareness between joined services on one another outputs, diver-

sification cannot be easily achieved during intermediate steps. Final results can still benefit from diversification, and in very few cases it diminishes the effects of join operations.

When multiple domains are involved, a *extract-match-rank* paradigm is used by the users. The MashRank system runs on a mashup editor, a mashup planner, and many content wrappers and grabber. Equivalent components can be found in the SeCo system, called *Query planner*, *Service Mart wrappers* and so on. The integration of data extraction with rank join processing is possible, but diversification will operate at his best only on final results after the last of the join operations has terminated.

In conclusion: MashRank addressed integrating information extraction with joining and ranking under uncertainty in the context of Web mashups. The integration of information extraction, relational optimizations (e.g., rank join), and the concepts of probabilistic databases can lead towards a relational and probabilistic query engine. SeCo is an example of possible evolution of the MashRank system, since diversification deals with ranking and ordering (especially in mashups) and is strictly affected by join operations and rank-joins too. We have addressed the problem of diversification considering all the possible steps where it can be achieved. The optimal solution is to diversify on the final results and not during intermediate steps so our interest has moved from a general perspective to the algorithms and ideas which share this intuition. Nevertheless SeCo is still investigating on the partial approaches for diversification, since similar problems have been already tackled by the "more-one" command under a multiple domain situation. The use of forecasting systems able to predict which of the involved services would probably maximize the user satisfaction among involved ones, can possibly be adjusted to maximize diversity and minimizing repetitions, but it is not the case of our study.

2.2 Query Disambiguation

This section addresses the problem of diversification from a query perspective. It is in fact possible to obtain diverse and novel results using different keywords for the same query.

A first possible approach to diversification is to disambiguate on query interpretations, which has the advantage of being computationally lighter than diversification on real data. This aspect is cru-

cial when considering the huge amount of information most modern search systems deal with. User queries are often ambiguous and may have more than one correct answer. Let us consider for example a single-word query: "Jaguar". There are at least three domains where "Jaguar" can belong to: it can be intended as the animal, the car constructor company and the operating system. There is no way to know in advance which of these interpretations is the one sought by a generic user. A naive solution to this uncertainty is to study the behavior and preferences of each user through a personal logging systems [28] supported by reasoning tools, which is theoretically possible but needs large amount of data and huge computational effort. Another possible solution is to adopt statistical measures to address the majority of users, studying the most frequent means distributions. Some studies in this direction, trying to match the same query to different meanings have been made in [27][30], exploiting search engines logs to address different interpretations using subsequent query refinement. These experiments though revealed drawbacks and limitations due to changes in people's intentions over time and unpredictable behavior, hard to overcome.

A different solution to query disambiguation is to offer the greatest possible variety of interpretations to users, maximizing the diversification and covering the widest spectrum of results; the probability of satisfying the user, without subsequent query refinements, hence is maximized.

Other approaches involves the use of different semantics in the query (mapping keywords to ontologies), semantics in the attributes when the query relies on structured systems, or clustering results based on previously-computed hierarchical structures to achieve the maximum coverability.

Every approach is again showed presenting the papers which we identified as the most representative on the topic.

2.2.1 Diversification for keyword over structured data

A possible approach to query disambiguation is to map query keywords to database attributes to give them a semantic meaning. Demidova et al make a move in this direction using the concept of *query template* and structured pattern frequently used to query the database [9]. For instance, let's consider a movie database, if the inserted query is something like "Cameron Titanic", a possible query

template could be "A director X of a movie Y". This keyword mapping is obviously not an easy task and it is not always possible to link every word to attributes. In this case we talk about *partial matching*, while we have a *complete matching* when all the query keyword can be connected to at least one attribute. Once this step is completed, the output is a set of possible query interpretations, linked to different query templates, and we can choose the best ones introducing the concept of relevance. In order to give a different importance to interpretation, a probabilistic model is used where $P = (Q|K)$ is the probability that a certain interpretation is the right one for a given bag of keywords. Query interpretation are also examined on their similarity using Jaccard distance $Sim(Q1, Q2) = \frac{Q1 \cap Q2}{Q1 \cup Q2}$. Having both relevance and diversity (as diversity = 1-similarity), Demidova et al. suggest the use of *MMR* (an algorithm that will be better examined later) to calculate query interpretation score as the trade-off between the two dimensions.

Beyond the interesting model, [9] suggest also 2 possible evaluation metrics called α -*NDCG* and *S-recall*, derived from standard information retrieval concepts but keeping diversity into consideration.

2.2.2 Keyword refinement using diversity

In the direction of query disambiguation, we can also consider [13]. Griffith and Pfeifer analyze 3 different approaches to attain this goal exploiting *Wikipedia disambiguation pages* and collected query logs.

- **CBC Word Sense Similarity** CBC Word Sense Similarity [25] tries to refine queries using a feature vector of words in a particular syntactic context. As an example [sip + Verb-Object] is a feature for wine. From these vectors, different subquery are obtained and their results are clustered to select representative elements. Drawbacks of this approach are the excessive dimension of obtained clusters, the difficulty of semantic identification and a high degree of overlap.
- **Web Semantic Clustering** To solve CBC Word Sense Similarity problems, Web Semantic Clustering is considered: query refinement are obtained from the top 50 results retrieved from Yahoo™ organized as binary feature vector. Two of these vectors are assigned to each possible refinement term, one based on their URL, one constructed with unique hostname as feature.

The disambiguation phase is then performed through Euclidean distance and clustering. Like the previous one, this approach has also few drawbacks related especially to the difficulty of fitting points in clusters and to the high computational cost.

- *MMR* The last way to clarify query disambiguation proposed by Griffith is, as already seen in [9], to adopt *MMR*, using the refinement term occurrences compared to the original one as relevance, and the trade-off between domains and URL terms as diversity .

MMR proved to perform better than other methods but the results of the user testing were controversial: sometimes refinements were even worse than original query.

2.2.3 Ambiguous query identification

Song [33] takes the issue from another perspective. Their objective is not to propose an efficient way to disambiguate query interpretations but to understand when a query actually needs to be refined in more specific subqueries. The idea is to perform refinement algorithms only when necessary, since they are heavy from a computational perspective. Song et al. identify 3 different kinds of query:

- **Ambiguous query:** having more than a possible meaning
- **Broad query:** covering a variety of subtopics thus needing refinement
- **Clear query:** specific meaning and single topic of interest

From the user testing they show how human being are easily able to identify ambiguous queries while have more trouble distinguishing between clear and broad ones.

From this starting point Super Vector Machine model [37] were suggested.

Documents are represented by a vector of categories and plotted on a graph where each dimension corresponds to the confidence that the document belongs to that specific category. Exploiting the intuition that documents with different interpretation probably belong to many categories, their method analyzes the degree of matching between a document and its field of interest, and is able to distinguish clear query, belonging to a single main category, from ambiguous ones, spreading over multiple dimensions.

2.2.4 Sub-query generation using query logs

All related works examined until now focussed mainly on server-side diversification, attempting either to cover most diverse topics for a certain bag of keywords or to identify the most probable query interpretation. [27] recognizes the heavy computational workload necessary to understand user intent and, as a solution proposes a client-side approach. Query results can be diversified by the end user system without much effort, but it needs enough elements to perform a meaningful diversification. To solve this, the proposed method considers different interpretation based on query logs. Given a query q , a set $R(q)$ of k related query is generated. Data logged for 6 weeks are used seeking for subsequent query in a 30 minutes window and 3 ways are proposed to identify best representative subquery:

- **Most Frequent:** select queries that most often follow the original one in the 30 minutes lapse
- **Maximum result variety:** *MMR* approach, greedily select queries that are both frequent in reformulation and different from those already selected
- **Most satisfied:** select queries that are not often reformulated, yet occur with minimum frequency

Once the different subqueries are retrieved, i.e. $R(q)$ is generated, let n be the number of desired results, $\frac{n}{k+1}$ elements are taken from each related query and the original one. This result selection approach is similar to clustering.

2.2.5 Query decomposition

Another work dealing with query disambiguation is [3], whose objective is to decompose a starting query into a set of sub-query to retrieve a set having a good coverage even in case of ambiguous and broad queries [33]. This direction is a bit different from already seen ones like result clustering and query recommendation . These approaches aim at retrieving a set of queries ordered by relatedness and frequency or clusters of results, while the the target of query decomposition is to:

- Find the query set that covers the whole result set of the original query
- Queries in the set have minimum overlap

- Each query should not cover many documents outside the initial result set

Bonchi et al. [3] identify many different potential applications for their model, like query filtering(remove too close query recommendations), query diversification(good coverage with low overlap creates diversity), query-set model (selecting terms to represent documents) and query result presentation(adapt representation according to different needs). Their query decomposition method can be executed in 2 different ways:

- **Top-down:** solves set covering problem with either greedy or LP(Linear Programming) algorithms. Each query has its own weight given by its coherence with the topic, percentage of documents covered compared to the original query, percentage of insignificant documents covered and its degree of overlap with other queries.
- **Bottom-up:** original query result set is clusterized in a hierarchical tree [18] and queries are matched to tree nodes using a matching-score function that considers all the parameters already seen for top-down approach

This model has not been tested in large-scale experiments but results from sample applications were appreciable, with better results from top-down method

2.2.6 Query diversification through Wikipedia

An interesting user intent analysis attempt can be found in [15] where the suggested strategy is to exploit the great power of Wikipedia. This time, the free encyclopedia is addressed for its articles and not for its disambiguation pages, as seen in 2.2.2.

The paper deals with 3 types of challenge

- **Semantic Representation:** precisely define semantic representation able to understand user intents. This is currently done using query logs with the obvious drawback of the huge amount of data needed for a meaningful analysis
- **Domain coverage:** understand which domains a query falls in, traditionally done with query logs as well
- **Semantic Interpretation:** consider the possibility that same syntactic may correspond to different semantic. For instance,

searching "Steve Jobs" does not have to be associated with jobs as working careers

Each intent domain for a query is mapped as a set of Wikipedia article and category based on close elements. For example, the domain "Travel" has sibling elements as "Hotel" and "Car Rental" and linked elements like "Travel Agencies". From this set, a tree showing its hierarchy is built, giving a score to each concept according to Markov random walk algorithm. Exploiting these scores and a probabilistic model for user intent based on previous query logs, best sub-queries are selected. A problem may arise if the query can't be easily mapped to Wikipedia domains. In this case queries are linked to most related concepts using ESA(Explicit Semantic Analysis) [11] which performs a semantic analysis on keywords. If the query is too ambiguous even for ESA, the proposed algorithm is not applied to avoid meaningless results.

2.3 Diversification over unstructured data

Like we have already seen in section 2.5, diversification on unstructured document, like most web pages, is extremely difficult to develop lacking a framework to map query keywords. Nevertheless we can take advantage of certain feature characterizing these kinds of documents like their category or the information they carry. This knowledge is not always given a-priori and many assumption are often needed to model these algorithms.

This section deals with unstructured document diversification. This topic may seem irrelevant to our work but proposed models, adaptable with few expedients, lay the basis for algorithms used for structured diversification (like we will see for *MMR* in section 3.4.3).

2.3.1 Intent-aware document diversification

In [2], the authors propose an algorithm for diversification on unstructured documents called *IA-select* where the IA stands for "Intent Aware" since it considers users intention in the result set computation. The idea is to take advantage of the knowledge of topics the query or documents refers to, hence documents are mapped to a taxonomy of category.

The approach is probabilistic and assumes the knowledge of the distribution of user intent over categories which is a strong assumption on most systems. To formalize the model, we call $P(c|q)$ the probability that query q refers to category c and we assume to have complete knowledge i.e. $\sum_c P(c|q) = 1$. We also define $V(d|q, c)$ as the value of document d for query q intended for category c , corresponding to the likelihood of a particular document to satisfy the user with a specific intent(category). Like most diversification algorithms, the objective of IA-Select is to maximize the probability that the average user finds at least one useful result in top-k ones and this problem is well known to be NP-hard, hence this method is a greedy approximation exploiting sub-modularity of $P(c|q)$. Sub-modularity is a particular property, often used in economics, that implies a greater benefits adding a document to a small collection rather than adding it to a large one. Considering this property and the assumption that the conditioned probabilities of 2 documents satisfying the user are independent, IA-select define its scoring function as:

$$Score(d) = V(d|q, c) * U(c|q, S) \quad (2.1)$$

where $U(c|q, S)$ is the conditioned probability that query q belongs to category c , given that all the documents in S (set of already selected documents) failed to satisfy the user. For each iteration, the scoring function is computed, the best document is selected and inserted into S and $U(c|q, S)$ is lowered for all category of the chosen document, according its value $V(d|q, c)$.

Being a greedy algorithm, IA-Select does not grant optimality of its output but in the case where all documents belong to a single category. The resulting set is still a sub-optimal approximation computed in polynomial time. Agrawal et al. [2] also adapt standard information retrieval metrics to consider user intent:

- $IA-NDCG(Q,k) = \sum_c NDCG(Q, k|c) * P(c|q)$
- $MRR-IA(Q,k) = \sum_c MRR(Q, k|c) * P(c|q)$
- $MAP-IA(Q,k) = \sum_c MAP(Q, k|c) * P(c|q)$

where Q is a ranked result set of documents. NDCG will be further discussed later in 4.1 since a modified version has been used for the quantitative testing phase while MRR and MAP were not used at all. Just to give a brief explanation *mean reciprocal rank* (MRR) is defined as

$$MRR = \frac{1}{Q} \sum_{i=0}^Q \frac{1}{rank_i} \quad (2.2)$$

where Q is a sample of queries, while *mean average precision* (MAP) can be expressed as

$$MAP = \frac{\sum_{r=1}^N P(r) * rel(r)}{\text{number of relevant documents}} \quad (2.3)$$

where r is the rank, N the number of document retrieved, $rel()$ a binary function of the relevance of a given rank and $P(r)$ precision at a given cut-off rank is equal to:

$$P(r) = \frac{|\text{relevant retrieved documents of rank } r \text{ or less}|}{r} \quad (2.4)$$

In the initial phases of our work, *IA-Select* was considered but due its strong assumptions, it was ignored for any further implementation.

2.3.2 Maximal marginal relevance

A really valuable work for our thesis is [6] since it introduces the concept of *maximal marginal relevance* (*MMR*). A document has high marginal relevance if it's both relevant and contains minimal similarity compared to previously selected documents. From this definition it is clear that *MMR* is composed by two dimensions, relevance and diversity, and they are linked according to the equation:

$$MMR = \arg \max_{D_i \in R \setminus S} (\lambda Sim_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j)) \quad (2.5)$$

where D_i is the document we are computing marginal relevance for, Q the query executed, R the set retrieved for Q , S the set of already selected items, λ the trade-off parameter and Sim_1 and Sim_2 are two similarity function. The idea behind equation 2.5 is to create a trade-off, through λ , between the two function Sim_1 and Sim_2 , respectively the similarity between i -th document and the query and between same document and already selected ones.

Sim_1 and Sim_2 can even be the same function. Given equation 2.5, *MMR* computes incrementally standard ranked list for $\lambda = 1$ and maximal diversity ranking for $\lambda = 0$. For λ values in $[0,1]$, a linear combination of both criteria is optimized. [6] suggests as an effective search strategy to start with a small λ to cover the information space and to increase its value to drill-down on a refined query.

The application scope Carbonell and Goldstein use *MMR* for is document summarization but the algorithm is easily adaptable to multi-domain case as shown in section 3.4.3 where *MMR* is implemented with a slightly different equation.

2.3.3 Information nugget model

Clarke et al. in [8] consider diversification problem in an interesting way. They show how current information retrieval metrics, like MAP and NDCG, fail in modern systems since they consider each document relevance as independent from other documents one. They state the *probability ranking principle* as:

Definition 1. *If an IR system's response to each query is a ranking of documents in order of decreasing probability of relevance, the overall effectiveness of the system to its user will be maximized*

but take it is as a starting point to define a new effectiveness criterion considering diversification as well. Their model consider user information need as a set of *nuggets*, any binary property as a piece of information or topic. Both user need u and document d are composed of nuggets so that $u, d \subseteq \mathcal{N}$ with $\mathcal{N} = n_1, \dots, n_m$ where n_i identify a single nugget. Given this model, a document d can be considered relevant if it contains at least one nugget from query q : $P(n_i \in d)$, the probability that document d includes nugget n_i is equal to

$$P(n_i \in d) = \begin{cases} \alpha & \text{if } J(d, i) = 1 \\ 0 & \text{else} \end{cases}$$

where $J(d,i)=1$ if the assessor has judged that document d contains nugget i . $\alpha \in [0, 1]$ is a constant considering the possibility of human error in the judgment.

To formulate a suitable objective function for this nugget based approach a strong assumption has to be made: $n_i \in u$ and $n_{j \neq i} \in u$ have to be considered as independent, i.e. nuggets forming the user need are not correlated. Under this assumption we have that

$$P(R = 1|u, d) = 1 - \prod_{i=1}^m (1 - P(n_i \in u)\alpha J(d, i)) \quad (2.6)$$

where R is a random variable representing relevance. Without any knowledge of user preferences, it is reasonable to make the assumption that all nuggets are independent and equally likely to be relevant hence we can set $P(n_i \in u) = \gamma$ with γ constant for all i and equation 2.6 becomes:

$$P(R = 1|u, d) = 1 - \prod_{i=1}^m (1 - \gamma\alpha J(d, i)) \quad (2.7)$$

So far we have found the equation(2.7) to find the document to be ranked first. We need now to consider the lower benefits of adding a document with already selected nuggets compared to one completely new. To attain this, we define the probability that the user is interested in n_i considering already selected ones:

$$P(n_i \in u|d_1, \dots, d_{k-1}) = P(n_i \in u) \prod_{j=1}^{k-1} P(n_i \notin d_j) \quad (2.8)$$

and $r_{i,k-1}$ as the number of documents ranked up to position $k-1$ judged to contain n_i :

$$r_{i,k-1} = \sum_{j=1}^{k-1} J(d_j, i) \quad (2.9)$$

Considering $r_{i,0} = 0$, we have that

$$\prod_{j=1}^{k-1} P(n_i \notin d_j) = (1 - \alpha)^{r_{i,k-1}} \quad (2.10)$$

We can finally define equation 2.7 for subsequent documents as

$$P(R_k = 1|u, d_1, \dots, d_k) = 1 - \prod_{i=1}^m (1 - \gamma \alpha J(d_k, i) (1 - \alpha)^{r_{i,k-1}}) \quad (2.11)$$

In addition to this model, [8] also provides its own adaptation of α -NDCG metric where k -th element of gain vector is equal to

$$G[k] = \sum_{i=1}^m J(d_k, i) (1 - \alpha)^{r_{i,k-1}} \quad (2.12)$$

This metrics can be adapted to multi-domain context, as we will see in section 4.1. Overall, Carbonell and Goldstein set an interesting framework for document diversification despite the strong assumption made in the formalization.

2.4 Diversification over structured documents

Recent studies have proven that 50% of keyword searches on the web are for *information exploration* purposes [20], and inherently have multiple relevant results. Such queries are classified as *informational queries*, where a user would like to investigate, evaluate, compare and synthesize multiple relevant results for information discovery and decision making. In contrast are the *navigational queries*, whose intent is to reach a particular website. Without the help of tools that can automatically or semi-automatically analyze multiple results, a user has to manually read, comprehend, and analyze the results in informational queries. Such process can be time consuming, labor-intensive, error prone or even infeasible. Text documents are unstructured, making it extremely difficult if not impossible to develop a tool that automatically compares the semantics of two documents. Structured data can have meta information, presenting a potential way to enable result comparison.

2.4.1 Desiderata, limitations and optimality

Structured search result differentiation has been addressed by Liu et al. [1][20], using an XML relational database for their experiments. The main idea was to use markups in the XML language to identify the features of different items, intended as tuple attributes, to enable

a later comparison between those objects. Let assume we want to compare an "apple" with an "orange", and we are operating within the same table. Those items are distinguishable by attributes such as "size", "color", "isFruit", and so on. In relational databases, though, other attributes may also be present, especially when the compared items belong to different tables. Those "other" attributes may not be significant for this specific comparison, but essential for others. Knowing in advance the needed attributes to enable the comparison between objects or adopting (if exist!) a method for identifying those attributes, will allow the system to easily identify similar and diverse results.

The author of this paper first defined *Differentiation Feature Set* (DFS) as the list of attributes needed for the comparison. They then identify three desiderata of selecting DFS from search results and propose an objective function to quantify the degree of differentiation among a set of DFSs. Identifying valid features that maximize the objective function given a size limit is an NP-hard problem and therefore they suggest two local optimality criteria which judge the quality of an algorithm for selecting DFSs: *single-swap optimality* and *multi-swap optimality*

Each differentiation feature set should have

1. limited size
2. reasonable summary
3. maximal differentiation

Maximal differentiation is the optimization goal in generating DFSs, limited size and reasonable summary are necessary conditions. The limited size helps users to compare search results, and enable them to be quickly differentiate. Summarizing query results means capturing only the main characteristics within the maximum set of attributes in which the DFS can be found. Otherwise, the differences shown in two differentiation feature sets do not reflect the actual differences.

Differentiability of DFSs: two results are comparable by their DFSs if their DFSs have common features types. Two results are differentiable if the DFSs have different characteristics of those shared feature types.

The diversification problem is therefore moved to the DFS construction, which is proven to be NP-hard.

Two properties become important from this point of view:

Single Swap optimality A set of DFSs is single-swap optimal for query results $R_1; R_2 \dots R_n$; if, by changing or adding one feature in a DFS D_i or R_i , $1 \leq i \leq n$, while keeping $valid(D_i, R_i, p)$ and $|D_i| \leq L$, their degree of differentiation, DoD (D_1, D_2, \dots, D_n) , cannot increase.

Multi Swap optimality A set of DFSs is multi-swap optimal for query results $R_1; R_2 \dots R_n$; if, by making any changes to a DFS D_i or R_i , $1 \leq i \leq n$, while keeping $valid(D_i, R_i, p)$ and $|D_i| \leq L$, DoD (D_1, D_2, \dots, D_n) , cannot increase.

Single-swap optimality guarantees that the DoD of a set of DFSs won't increase by changing one feature in a DFS. On the contrary, multi-swap optimality requires that the DoD cannot increase by changing any number of features in a DFS.

In conclusions: the paper presented addresses the novel problem of designing tools that automatically differentiate structured search results. Thus relieve users from labor intensive procedures of manually checking and comparing potentially large results. The notions of Differentiation Feature Set (DFS) for each result and the Degree of Differentiation (DoD) became useful to identify three desiderata for good DFSs: differentiability, validity and small size. The problem of constructing DFSs that are valid and can maximally differentiate a set of results within a size bound is an NP-hard. To provide practical solutions, two local optimality criteria, single-swap optimality and multi-swap optimality were proposed. In our work we use similar optimality criteria, adjusting their concepts to adapt in a multi-domain environment and maintaining the properties of differentiability and validity (DFS desiderata).

2.4.2 Trees of diversity

As seen in the previous section, a possible approach is to make the user specify queries that can be factorized into subqueries covering different domains [38]. If we ask users to specify queries through a form interface that allows a mix of structured and content-based selection conditions, it would be possible to divide the results into different domains instead of subqueries. Intuitively, the goal of diverse query answering is to return a representative set of top-k answers [14][16] from all the tuples that satisfy the user selection condition. For example, if a user is searching for cars and we can only display

five results, we wish to return cars from five different models, as opposed to returning cars from only one or two models. Obviously if the user is more interested into different sellers for the same car model, the diversification we are about to suggest is to consider as counterproductive, but it will be always possible to specify this kind of user intent with a proper query specification.

Now consider items stored in relational databases as tuples. A query Q on a relation R is a conjunction or disjunction of two kinds of predicates: scalar predicates (in the form of "attribute" = "value") and keyword predicates (in the form "keyword" is_contained_into "attribute"). Vee et al. suggest to adopt an ordering on attributes to evaluate the diversity of a set.

A *diversity ordering* of a relation R with attributes $A_1, A_2 \dots A_n$, is a total ordering of all the attributes within A . In the car example, we can define a diversity ordering on the attributes "Color", "Model", "Maker", "Year" as

$$\text{Make} \prec \text{Model} \prec \text{Color} \prec \text{Year}.$$

A score can be associated to each tuple, evaluating that tuple with respect to both the query Q and the diversity order just specified. This score can be the result of different similarity functions. The easiest to use is

$$f(x) = \begin{cases} 1 & \text{if the tuples have the same value for a certain attribute} \\ 0 & \text{otherwise} \end{cases}$$

Since there is an order for the attributes, it is possible to choose different similarity functions, with tighter or more loose scopes, as well as a more specific definition for the attributes (*leveling*), depending of the system needs.

In the given example, since *model* precedes *color*, it wouldn't be possible to receive a set of different color for the same model if different models were still available for the selection. Unfortunately, it is not always possible to define such ordering in our situation (Section 5.1)

Nevertheless, this criterion allows to easily measure diversity between any two sets. Those can be defined as *diverse* if all the attributes until a certain depth (level), take different value one from the other.

Vee et al. propose an algorithm based on *Dewey tree*. Each item is assigned a unique id, using the Dewey encoding [41]. Every leaf value is obtained by traversing the tree top down and assigning a different integer to siblings. They use a WAND-like algorithms [39] to get the first item satisfying the query returned.

There are four version of the algorithm: **one-pass unscored**, **one-pass scored**, **unscored probing** and **scored probing**.

one-pass unscored left to right traversal, without going back, using only "next" calls, query Q, requesting K items, the tentative result set starts as the first of the k item is matching the query Q. Then it can skip to the next dewey id that could possibly improve the diversity in their tentative result set. Repeat until no more items match the query. Key steps of this approach is to decide how far to skip in the dewey tree, and how to decide which element to remove from the tentative tree to make room for the latest item found (like a "branch and bound knapsack" problem).

one-pass scored the skipping phase is ruled by this: skip only to the smallest of the original Dewey id and the next item whose score is greater than or equal to the minimum score among items in the current result set.

unscored probing probes the dewey tree searching the branch that will be most beneficial in generating a diverse result set("next" and "prev" here implies left-to-right navigation followed by right-to-left on "children").

In conclusions: Vee et al. have formalized diversity in structured search and proposed inverted list algorithms (proven to be scalable and efficient). Diversity is achieved through little overhead and can be implemented on relational databases as well as other structured documents. The natural extension of their approach is to produce weighted results by assigning weights to the different attributes or to different attribute values. Our work goes toward this direction, using as main distance between tuple single or multiple attributes (see diversity measures in section 4.1) and setting the basis for a future refinement where the weight of the attributes can be decided by the user itself.

2.4.3 User preferences over relational databases

Keyword-based searches may return an overwhelming number of results, often loosely related to the user intent. In [34], the authors propose a way to personalize keyword database searches by utilizing user preferences. Query results are ranked based on both their relevance to the query and their preference degree for the user. To further increase the quality of results, they consider two new metrics that evaluate the goodness of the result as a set, namely *coverage* of many user interests and content diversity. An algorithm for processing preference queries that uses the preferential order between keywords to direct the joining of relevant tuples from multiple relations.

In relational databases, existing keyword search approaches exploit the database schema or the given database instance to retrieve tuples relevant to the keywords of the query. Keyword search is intrinsically ambiguous. Personalizing keyword search means giving to different users different results, based on their personal interests. Preferences can be considered as choices in different context. Let consider for example the movie domain, and a database with "Director", "Actors" and "Genres" as attributes.

A user can specify two, contrasting, preferences, valid under precise conditions:

$$\begin{aligned} &\text{"Oldman"} \prec \text{"Allen"} \text{ when genre} = \text{"Thriller"} \\ &\text{"Allen"} \prec \text{"Oldman"} \text{ when genre} = \text{"Comedy"} \end{aligned}$$

Preferences may be specified ad-hoc during the query formulation (forms) but can also be created automatically, based on explicit (questionnaires) or implicit user feedback (logs). Since keyword search is often best-effort, given a constraint k on the number of results, the idea is to combine the order of results as indicated by the user preferences with their relevance to the query. The algorithm studied are based on the winnow operator (\prec), applied on various levels to retrieve the most preferable choices at each level.

In conclusions: by extending query-relevance ranking with preferential ranking, users are expected to receive results that are more interesting to them. Keeping the simplicity of the keyword search paradigm and selecting k representative results that cover many user interests and exhibit small overlap is the right choice for maximising the user satisfaction.

2.4.4 User preferences, items competition and trade-offs

Another way to treat diversity is to consider all attributes as equally weighted. Feng et al [26] propose a framework which operates under this hypothesis. They propose **MAPS** - a personalized Multi-Attribute Probabilistic Selection framework which integrates trade-offs and preferences. The MAPS framework consists of four key components: the inter-attribute tradeoff, the inter-item competition, the personalized user preferences, or the probability-based ranking scores.

1. First, given a set of items to be ranked, they use a visual angle model to map multi-attribute items into points (*stars*) in the corresponding multidimensional space (*sky*).
2. Second, the *dominating area* of an item in the preference space is defined to model the inter-item competition. The concept of dominating area is based on two observations: items which have similar attribute values are "neighbors" in the preference space, compete with each other to be a user's best choice. Plus, as the items become farther apart from one another, the competition between them reduces.
3. Third, using a density function over the whole preference space, they infer where the best choice will more likely reside given a user's personal preferences .
4. Finally, the probability of each item being a user's best choice can be calculated as the integral of the user's density function over each item's dominating area.

In conclusion: the MAPS framework suggests the use of users preferences combined with the inter-item competition and probability-based ranking altogether to estimate the best and most diverse user choices. As side effect of their work, they prove that the set of attributes that can really affect users' decision is typically much smaller than the one provided by most existing systems, and our work have taken this result in consideration.

2.5 Related Works conclusion

Diversification is a problem studied in several fields: economics theories, recommendation systems [44] and multimedia search engines

use diversification concepts to improve the performances of their tasks.

The portfolio diversification theory aims at reducing the risks and maximizing revenues spreading the investments over different stock options. Recommendation systems use diversification to provide suggestions to users [42], computing similarity measures on items and images. For shopping recommendation systems, items similar to those bought or viewed by the user are suggested. Image diversification is divided in two paradigms: suggest resembling the contents of viewed images [35], or uses community based tags to return look-alike and similar pictures [36].

Interesting researches on mixing diversification with user preferences have been made [22]. Since the data structure can be seen as skylines of attributes and values, diversity and similarity assume different meanings and interpretations [21].

As seen in this chapter, diversification possibilities and uses are almost boundless, therefore addressed from different perspectives. Our work intends to discover its utilization within the SeCo project scope.

Chapter 3

Framework

To understand topics discussed in this work, it is useful to consider its framework. This chapter is divided in four sections. The first section introduces a formalization of the diversification problem, the second focuses on the notions of *relevance* and *diversity*, the third provides insights on the tested algorithms and the last one is about dataset structure used for quantitative and qualitative testing phases.

3.1 Problem formalization

A proper problem formalization is needed in order to explain algorithms in a clear way. The first basic concept to define is *multi-domain query*. Let R_1, R_2, \dots, R_n be a set of relations where R_i denotes the result set obtained as output querying a web search service σ_i . Each tuple $t_i \in R_i = \langle a_i^1, a_i^2, \dots, a_i^{m_i} \rangle$ has the schema $R_i(A_i^1 : D_i^1, \dots, A_i^{m_i} : D_i^{m_i})$, where $A_i^{m_i}$ is an attribute of relation R_i and $D_i^{m_i}$ is the domain associated to the result set. This work considers two kind of domains: *categorical* and *quantitative* ones. The former implies only binary values, comparable with equality tests, while for the latter, values are set up in vector within a given metric space. This division will be further discussed in section 3.3. Given these premises, we can define a multi-domain query over search services $\sigma_1 \dots \sigma_n$ as the join query $q = R_1 \bowtie \dots \bowtie R_n$ over relations R_1, \dots, R_n with an arbitrary join predicate.

Another important notion to qualify is the *combination* one: we call combination an element of the join $\tau = t_1 \bowtie \dots \bowtie t_n = \langle$

$a_1^1, a_1^2, \dots, a_1^{m_1}, \dots, a_n^1, a_n^2, \dots, a_n^{m_n} >$, and *result set* \mathcal{R} the set of combinations satisfying query q .

3.2 Relevance

Most well known search approaches, like Google one, are based on *relevance* notion: the objective is to return to users a list of results ranked according to their importance. The relevance function can be formalized as $S(\tau, q)$ normalized in $[0,1]$ where τ is the combination under exam and q is the query performed to get the result set. This sorting can be achieved as a simple SQL ordering on one or multiple attributes or based upon user preferences. In both cases the resulting set \mathcal{R}' has a monotonically decreasing relevance scoring i.e. $S(\tau_k, q) > S(\tau_{k+1}, q)$ where τ_k is the k -th combination in \mathcal{R}' . In case of multi-domain queries this function is not immediate, since it's based on attributes belonging to different sets.

An example can help: suppose we have a multi-domain query q selecting combinations of museums, restaurants and hotels in Milan and we order it by relevance. In this situation we probably want to use different aggregation functions for different attributes: for the "total price" $sum()$ is an obvious choice, but for the "global rating", $avg()$ would make more sense; if we have to consider "geographical distance", a complex function is need to be used instead. Each attribute can require a different aggregation function, as explained in section 3.5.

3.3 Diversity

The new paradigm introduced by multi-domain query has added a further dimension to relevance in order to satisfy user needs. Using as example the query seen in 3.2 and sorting by "price", we may see an extremely homogeneous set since a really cheap hotel or restaurant would *dominate* the others and as consequence would be repeated many times in the top- k elements. Such result set is low valuable and repetitive, since a better coverage of museums/restaurants/hotels would be more interesting for the user. This is the typical situation where diversification plays a key role, attempting to reorder the set, to increase its variety. Two types of diversity can be identified: *categorical diversity*, where the distance between combinations is computed as binary, i.e. equality is compared, and

quantitative diversity where the diversity can be obtained using vectors in a metric space. To understand these concepts, consider two combinations τ_u and τ_v and define $\delta : \mathcal{R}X\mathcal{R}$ as the diversity function, normalized as usual in $[0, 1]$ to be compared with relevance one.

3.3.1 Categorical diversity

Given the background just introduced, we can define categorical diversity as:

$$\delta(\tau_u, \tau_v) = 1 - \frac{1}{n} \sum_{i=0}^n \mathbb{1}_{v_i(\tau_u)=v_i(\tau_v)} \quad (3.1)$$

where n is the number of relations R_i , $\mathbb{1}$ is the indicator function that returns 1 if the predicate is satisfied, 0 otherwise, and $v_i(\tau) = [a_i^{j_1,1}, \dots, a_i^{j_i,d_i}]^T$ is the projection of the combination τ on the subset $A_i^{j_1,1}, \dots, A_i^{j_i,d_i}$ of d_i attributes of the relation R_i

Despite the apparent complexity of the equation, the principle is really simple: categorical distance between two combinations is given by the normalized comparison between examined attributes. For instance, if the combinations are composed by the relations hotel/restaurant/museum and we want to compute the distance between $\tau_u = \{H1, R1, M1\}$ and $\tau_v = \{H1, R2, M2\}$ where H_i, R_i, M_i are primary keys of each relation, we have $\delta(\tau_u, \tau_v) = 1 - \frac{1}{3}(1+0+0) = \frac{2}{3}$. We used the primary keys as diversification attributes as they are suitable for categorical diversification, especially since our aim is to return most differentiated instances for top-k results. Other suitable attributes for this kind of diversification are categorical ones, like restaurant or museum types, or the services offered by hotels.

3.3.2 Quantitative diversity

Quantitative diversity can be defined as:

$$\delta(\tau_u, \tau_v) = \frac{1}{\delta_{max}} \sqrt[p]{\sum_{l=1}^d w_l |v_l(\tau_u) - v_l(\tau_v)|^p} \quad (3.2)$$

where δ_{max} is a normalization constant, $v_i(\tau)$ is the same as in the categorial diversity case, d is the length of $v(\tau) = [v_1(\tau), \dots, v_n(\tau)]^T$ and w_1, \dots, w_d are user-defined weights. This kind of diversity has the form of weighed l_p - *norm* of the difference between $v(\tau_u)$ and $v(\tau_v)$ and the normalization constant we used for testings is the maximum distance between couple of combinations in the set.

An example is useful to understand equation 3.2: let $p=1$ (Manhattan norm), $w_l=1$ and $v(\tau) = [HRating(\tau), RRating(\tau), MRating(\tau)]$, distance between tuple τ_u and τ_v is computed using $v(\tau_u) = [35, 25, 0]$ and $v(\tau_v) = [36, 25, 2]$ and it's equal to $\delta(\tau_u, \tau_v) = \frac{1+0+2}{\delta_{max}}$. This distance metric is more dependent on the attribute value distribution than the previous one. Data structure, in fact, can greatly influence the effects of algorithms.

3.4 Algorithms

Let's now examine the tested algorithms. Since diversification problem is NP-Hard [24], these approaches are all greedy approximations whose quality needs to be verified, as done in the testing phase. Greedy algorithms are usually iterative and do not grant optimality of the solution. Although they adopt different scoring functions to determine the best combination, they are all based onto the same principle: create a trade-off between relevance and diversity using a tunable parameter[19]. Modifying this parameter ensures the possibility to easily adapt and apply them to different situations, covering a great variety of circumstances.

3.4.1 MaxSum

The first two examined algorithms, *MaxSum* and *MaxMin* are presented in [12]. The aim of selecting top-k most *relevant* and *diverse* combination is formalized as the following optimization problem:

$$\mathcal{R}_k^* = \arg \max_{\mathcal{R}_k \subseteq \mathcal{R}, |\mathcal{R}_k|=K} F(\mathcal{R}_k, S(., q), \delta(., .)) \quad (3.3)$$

where $F(.)$ is an objective function suitable for our purpose since takes into account both the properties we are interested into. [12] proposes the following function for *MaxSum*:

$$F(\mathcal{R}_k) = (K - 1) \sum_{\tau \in \mathcal{R}_k} S(\tau, q) + 2\lambda \sum_{\tau_u, \tau_v \in \mathcal{R}_k} \delta(\tau_u, \tau_v) \quad (3.4)$$

where $S(\tau, q)$ and $\delta(\tau_u, \tau_v)$ are respectively the *relevance* and the *diversity* component, and $\lambda \in [0, 1]$ is the trade-off parameter in-between them.

As mentioned, this problem is NP-Hard by its definition, hence 3.4 applied to 3.3 won't return an optimal solution in polynomial time. *MaxSum* greedy approximations can give a sub-optimal solution. The idea is to iterate over all combinations, seeking for the one which maximises the scoring function and, as a consequence, is both the most relevant and diverse among selected items.

In *MaxSum* (Algorithm 1), the scoring function δ' depends on the sum of the examined couple τ_u, τ_v relevance plus the distance between its elements as in following:

$$\delta'(\tau_u, \tau_v) = S(\tau_u, q) + S(\tau_v, q) + 2\lambda\delta(\tau_u, \tau_v) \quad (3.5)$$

For each iteration, two combinations are selected with a distance varying with λ . Ideally when $\lambda = 1$ selected couples should be composed by the ones with a sufficiently high relevance which are also the most diverse elements among all.

Algorithm 1: Greedy algorithm for *MaxSum*.

Input : Set of combinations \mathcal{R} , K

Output: Selected combinations \mathcal{R}_K

begin

 Define $\delta'(\tau_u, \tau_v) = S(\tau_u, q) + S(\tau_v, q) + 2\lambda\delta(\tau_u, \tau_v)$

 Initialize the set $\mathcal{R}_K = \emptyset$, $U = \mathcal{R}$

for $c = 1 : \lfloor K/2 \rfloor$ **do**

 Find $(\tau_u, \tau_v) = \arg \max_{x, y \in U} \delta'(x, y)$

 Set $\mathcal{R}_K = \mathcal{R}_K \cup \{\tau_u, \tau_v\}$

 Set $U = U \setminus \{\tau_u, \tau_v\}$

end

 If K is odd, add an arbitrary combination to \mathcal{R}_K

return \mathcal{R}_K

end

This selection method has a huge drawback: the tuples selection at each iteration is independent from any previous steps. The selected tuples are not influenced by already selected ones, i.e. at a certain iteration it may happen to select two tuples very different one another, but almost identical to some couple chosen in previous

iterations (Algorithm 1). This approach has the risk of obtaining a set even worse than the starting one.

This case is relevant especially when considering binary distance, as we will see in Chapter 5.

3.4.2 MaxMin

MaxMin (Algorithm 2) and *MaxSum* try to solve the same optimization problem (3.3). In this case the function proposed by [12] is

$$F(\mathcal{R}_k) = \min_{\tau \in \mathcal{R}_k} S(\tau, q) + \lambda \min_{\tau_u, \tau_v \in \mathcal{R}_k} \delta(\tau_u, \tau_v) \quad (3.6)$$

while the scoring function used for its greedy approximation is

$$\delta'(\tau_u, \tau_v) = \frac{1}{2} (S(\tau_u, q) + S(\tau_v, q)) + \lambda \delta(\tau_u, \tau_v) \quad (3.7)$$

balancing once again the trade-off between the two dimensions.

On *MaxMin* first iteration, two tuples are chosen with a *MaxSum*-like method: the most diverse among all are selected and placed in the result set. Then in each round is extracted the item which maximizes δ' , i.e. the item with minimum distance from already selected objects and maximal relevances. Both *MaxSum* and *MaxMin* behave in the same way at λ variation: the closer λ is to 1, the more importance is given to diversity.

Algorithm 2: Greedy algorithm for *MaxMin*.

Input : Set of combinations \mathcal{R}, K

Output: Selected combinations \mathcal{R}_K

begin

Define $\delta'(\tau_u, \tau_v) = \frac{1}{2} (S(\tau_u, q) + S(\tau_v, q)) + \lambda \delta(\tau_u, \tau_v)$
Initialize the set $\mathcal{R}_K = \emptyset$
Find $(\tau_u, \tau_v) = \arg \max_{x, y \in \mathcal{R}} \delta'(x, y)$ and set $\mathcal{R}_K = \{\tau_u, \tau_v\}$
while $|\mathcal{R}_K| < K$ **do**
 $\tau^* = \arg \max_{\tau \in \mathcal{R} \setminus \mathcal{R}_K} \min_{x \in \mathcal{R}_K} \delta'(\tau, x)$
 Set $\mathcal{R}_K = \mathcal{R}_K \cup \{\tau^*\}$
end
return \mathcal{R}_K

end

3.4.3 MMR

The third algorithm uses *Maximal Marginal Relevance* (**MMR**, Algorithm 3) as objective function [6], a method based following the scoring criterion:

$$MMR = \arg \max_{D_i \in R \setminus S} \lambda Sim_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \quad (3.8)$$

We have modified this function, using $S(\tau, q)$ as relevance component and changing the second part to use diversity instead of similarity (this has the effect of turn the minus into a plus since higher value of δ increase the tuple score). After this adaptation, the scoring function is in the form:

$$\tau^* = \arg \max_{\tau \in \mathcal{R}} \lambda S(\tau, q) + (1 - \lambda) \min_{x \in \mathcal{R}_k} \delta(\tau, x) \quad (3.9)$$

As from equation 3.7, *MMR* is very similar to *MaxMin*, with the main difference in the scoring function. Compared with previous algorithm *MMR* can also give higher importance to diversity than to relevance since λ is in this approach a trade-off parameter effecting both properties, while in *MaxSum* and *MaxMin* it only affects the diversity component.

Algorithm 3: Greedy algorithm for *MMR*.

Input : Set of combinations \mathcal{R}, K

Output: Selected combinations \mathcal{R}_K

begin

 Initialize the set $\mathcal{R}_K = \emptyset$

 Find $\tau = \arg \max_{x \in \mathcal{R}} S(\tau, q)$

 Set $\mathcal{R}_K = \{\tau\}$

while $|\mathcal{R}_K| < K$ **do**

$\tau^* = \arg \max_{\tau \in \mathcal{R} \setminus \mathcal{R}_K} \lambda S(\tau, q) + (1 - \lambda) \min_{x \in \mathcal{R}_K} \delta(\tau, x)$

 Set $\mathcal{R}_K = \mathcal{R}_K \cup \{\tau^*\}$

end

return \mathcal{R}_K

end

3.4.4 MaxCov

The last algorithm is proposed in [31] and is called *MaxCov* (**Algorithm 4**) since it attempts to attain the maximum coverage over result set

choices both in binary(categorical) and attribute-based (quantitative) case. Its steps are identical to *MMR* except for the objective function having λ as exponential for the relevance part:

$$\tau^* = \arg \max_{\tau \in \mathcal{R} \setminus \mathcal{R}_K} S(\tau, q)^\lambda * \min_{x \in \mathcal{R}_K} \delta(\tau, x) \quad (3.10)$$

In [31] the relevance function is called DOM and represents the *degree of matching* between the query q and service R , which is easy to relate to " $S(\tau, q)$ " we introduced before. The concept of diversity in their work is defined as *coverage error* between a service S and a sub-set \mathcal{S}_k of query results, computed as:

$$cerr(S, \mathcal{S}_k) = \min_{S' \in \mathcal{S}_k} \delta(S, S') \quad (3.11)$$

which is once again similar to the concept of diversity we use, in the definition of δ , computed as in 3.3.

Algorithm 4: Greedy algorithm for *MaxCov*.

Input : Set of combinations \mathcal{R}, K

Output: Selected combinations \mathcal{R}_K

begin

 Initialize the set $\mathcal{R}_K = \emptyset$

 Find $\tau = \arg \max_{\tau \in \mathcal{R}} S(\tau, q)$

 Set $\mathcal{R}_K = \{\tau\}$

while $|\mathcal{R}_K| < K$ **do**

$\tau^* = \arg \max_{\tau \in \mathcal{R} \setminus \mathcal{R}_K} S(\tau, q)^\lambda * \min_{x \in \mathcal{R}_K} \delta(\tau, x)$

 Set $\mathcal{R}_K = \mathcal{R}_K \cup \{\tau^*\}$

end

return \mathcal{R}_k

end

3.5 Datasets

In order to test algorithms described in 3.4, we designed and created 3 datasets: **Rome**, **Milan** and **University**. The first two are composed by hotels, restaurant and museums from the homonym cities in Italy, the last one contains information on universities, flats and demographical information from the U.S.. Rome set has been used for training purposes only, while extensive analysis have been performed over Milan and University set. Since Rome and Milan

sets have identical structure, we will describe only one, implying all considerations to be valid for the other as well.

Set attributes were chosen in order to create realistic situations for the qualitative evaluation (Section 5). The first scenario, called *Night Out* (N.O), is based upon Milan dataset. We imagined a possible visitor to the city of Milan. This hypothetical user is seeking for a classical touristic experience. He is interested in visiting a museum, having dinner in a restaurant and spend the night in a close hotel. Information such as the total price of the evening, the distance to cover between sites, the quality of the hotel and opening hours, can be useful for the user choice. We tried to imagine all his interests and needs, and modeled the datasets (and test) consequently. To answer the requirements, information range from single prices (i.e. "museum cost") to global distance ratio (i.e. distance with the train station), from the hotel offered services to the restaurant category.

The second scenario, called *Study Abroad* (S.A), is based on a classical study-abroad experience, where the user has to decide where to study and where to live for the next years. In this situation the hypothetical student might be interested in the quality of the university and its field of studies, the campus life, the cost of the flat and many attributes we included in set 3.5.

To be as realistic as possible, we built these sets extracting information from existing, reliable websites and data.

3.5.1 Milan set

The **Milan** dataset has been composed picking 50 tuples for each service (hotels, museum and restaurants) and performing a full-join to obtain 125,000 total combinations. This amount of data gave us a good variety of choice, but also lead to few computational problems (explained in Chapter 4) that forced us to use smaller subsets during the qualitative analysis.

For this set, web sources were:

- **Milano da bere**[www.milanodabere.it] for restaurants
- **TripAdvisor**[www.tripadvisor.it] for hotels and restaurants
- **ZeroDelta**[www.zerodelta.net/milano-musei.php] for museums
- **Milan municipality**[www.comune.milano.it] for museums

- Venere[www.venere.com] for hotels
- Hotel Milano[www.hotelmilano.com] for hotels
- Booking[www.booking.com] for hotels

From an attribute prospective, all datasets were built to ensure a good range of possibilities both for categorial and quantitative distance(section 3.3).

Hotel		Museum		Restaurant	
id	int	id	int	id	int
name	string	name	string	name	string
address	string	address	string	address	string
latitude	float	latitude	float	latitude	float
longitude	float	longitude	float	longitude	float
stars	int	category	string	category	string
category	string	morning	bool	places	int
avgRating	float	afternoon	bool	price	int
reviewers	int	evening	bool	avgRating	float
price	int	weekHours	string	idealFor	multiple bool
services	multiple bool	weekendHours	string	lunch	bool
		fullFee	float	dinner	bool
		reducedFee	float	lunchHours	string
		services	multiple bool	dinnerHours	string
				services	multiple bool

Figure 3.1: Structure of Milan set combinations with attributes and their type.

Milan set attributes are proposed in figure 3.1, and here described:

Id	A unique number used to distinguish entities
Name	Entity name, used to identify the service item as well
Address	Entity address used to retrieve geographic coordinates
Latitude, Longitude	Geographic coordinates to compute distances
Category	Hotel: <i>B&B, Residence ,Hotel , Apartment</i> Museum: <i>Artistic, Historical, Archaeologic</i> , etc. Restaurant: <i>Italian, Chinese, American</i> , etc.

AvgRating	Average rating by users. Hotel has also the attribute reviewers , indicating the number of people who expressed their opinion
Price, FullFee, ReducedFee	Hotel: lowest price Museum: full\reduced entry fee Restaurant: average cost for a meal
Stars	Number of hotel stars, usable as a quality indicator
Places	Number of seats available in the restaurant
Morning, Afternoon, Evening, Lunch, Dinner	Boolean values indicating museum (Morning, Afternoon, Evening) and restaurant (Lunch, Dinner) opening periods
WeekHours, WeekendHours, LunchHours, DinnerHours	Opening and meal times for museum and restaurant respectively
IdealFor	A set of boolean attributes pointing out who the suggested public for the restaurant is: <i>couples, parties, business</i> , etc.
Services	A set of boolean attributes indicating services offered Hotel: <i>Air Conditioning, Safe, Wi-fi</i> , etc. Museum: <i>Laboratory, Parking Area, Shop</i> , etc. Restaurant: <i>Credit card, Summer opening, Disabled friendly</i> , etc.

The above attributes were all referred to single services (for instance *stars* is related only to the hotel service, *places* only to restaurant one). After the join operation, 125'000 combinations were obtained, but we wanted to have attributes able to characterize the whole combinations rather than single services within. To attain this goal we added the following attributes to the final set:

Global Id	The 3 Ids from each table are joint in a single one with the purpose of identify a specific combination and each of the specific component at a glance.
------------------	---

Distance\Time by car	Distance and time needed to reach the museum, the restaurant and the hotel, in this straight order by car, starting from Milan Centrale train station, according to Google maps services. This ordering is reasonable when related to the task "Plan your weekend in Milan", where a tourist would like to spend a day visiting a museum, then having a dinner in a not-so-far restaurant and finally get some rest in a Hotel.
Distance\Time on foot	Same as the previous one, but the same route is now performed on foot, which is more plausible for a foreign tourist.
Total price	Sum of hotel, museum and restaurant costs, useful to obtain the cheapest combination through a simple ordering.
Average rating	An average mean of the hotel and restaurant ratings.

Considering this list of attributes, it is easy to recognize those usable for a categorial diversification, like "global" ids and single service categories, and those appropriate for a quantitative distance. The latter class groups attributes that can be linked to a high number of attributes, since the only requirement is the ability to be set up in a metric space. Distances, prices, ratings and even opening hours fit to this condition (at most with few adaptations).

3.5.2 University set

The last dataset is composed by U.S. universities and close flats within the same state. This set was created from 58 U.S. institutes joined with 1,100 close-by flats, which gave us a final set composed of 5,100 combinations. Looking at figure 3.2 it is notable that *state* is an entity itself but we use it only to add information to the combinations rather than to increase variety, hence we won't consider categorial diversity on it.

All the data for universities and flats came from the following sources:

- University of California - Machine learning database[archive.ics.uci.edu/] for universities
- Trulia Real Estate Search[www.trulia.com/] for flats

This set attributes are:

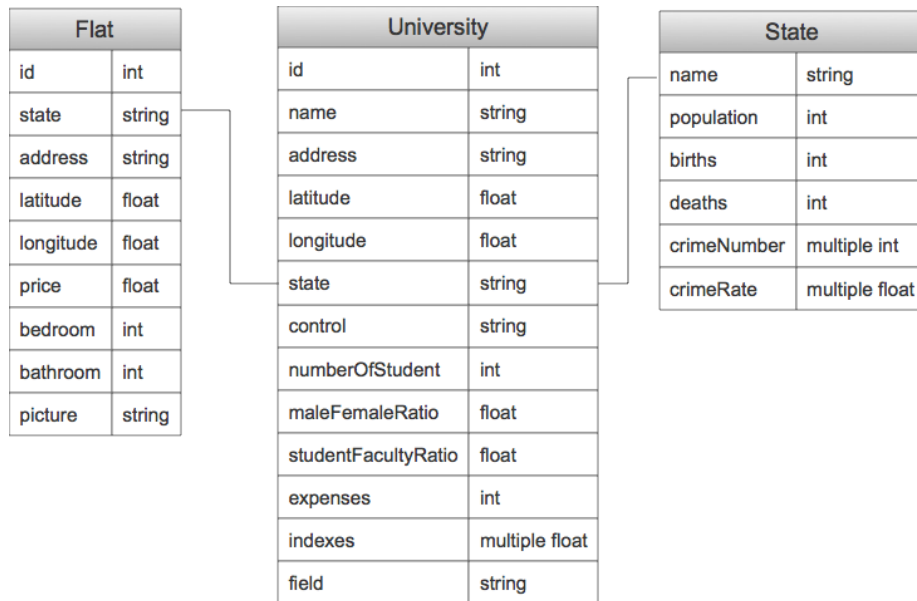


Figure 3.2: Structure of University set combinations with attributes and their type.

Id	A unique number, used to distinguish entities
Name	Name for the state or the university
Address	Entity address used to retrieve geographic coordinates and as primary key for flats
Latitude, Longitude	Geographic coordinates to compute distances
Price, expenses	Flat: monthly rent University: yearly tuition fee
State	Flat and university state, used as join attribute with state name
Bedroom, bathroom	Number bedrooms and bathrooms in the flat
Picture	An URL linking to the flat picture
Control	Management of the university: public of private
NumberOfStudents, maleFemaleRatio, studentFacultyRatio	Statistical data about the total number of students, male \female ratio and students per faculty ratio

Indexes	A series of indexes for the university <i>Percentage Aid</i> : <i>Admittance</i> : admittance rate <i>Enrolled</i> : enrollment rate <i>Academic scale</i> : institute rating <i>Social scale</i> : social rating <i>Walkability</i> : livability of the campus without any transport
Field	University fields: <i>engineering, biology, business</i> etc.
Field	University fields: <i>engineering, biology, business</i> etc.
Population, Births, Deaths	Demographical data for the state
CrimeNumber, CrimeRate	Total amount and rate per 100,000 inhabitants for different crime type: <i>Violent crime, Murder, Rape, Robbery, Assault, Property Crime, Burglary, Larceny, Vehicle Theft</i>

As seen in section 3.5.1 for Milan set, additional attributes for facilitating the join between entities has been created. For University dataset we have:

Global Id	The 2 ids are joint in a single one with the purpose of identify a specific combination.
Distance\Time by car	Distance and time by car computed by Google services
Distance\Time on foot	Same as previous attributes but the same route is now performed on foot.
Yearly total price	Total price to spend for a one year period considering both the flat rent for 12 months and the university yearly tuition fee.
Quality Index	A complex quality indicator calculated as a function of the academic quality score, the walkability index of a university and the crime rate in a state.

Attributes suitable for categorial distance are basically the same already specified for Milan set, i.e. the (universities) "ids" or the academic field. Indexes and prices are examples of appropriate attributes for the quantitative diversity for this dataset.

To summarize, the most important attributes for the two datasets, used as sorting and diversifying attributes are:

1. Milan dataset

Total price

Total distance

Average rating

2. University dataset

Quality index

Total distance

Yearly total price

Chapter 4

Quantitative test

In order to test the performance of the algorithms introduced in section 3.4 on a multi-domain query scenario, we designed a set of quantitative evaluations. The purpose of these evaluations is to assess if - and how - the adoption of diversification techniques can improve the quality of a multi-domain result set, using appropriate evaluation metrics. This chapter is organized as follows: Section 4.1 describes the adopted evaluation metrics; Section 4.2 illustrates experiment design and specifications; to conclude, Section 4.3 presents the results of the evaluation.

4.1 Metrics

Several works addressed diversification as an information retrieval technique (see Chapter 2). Given that our work lays in a well-explored research field, we decided to start from well-known evaluation metrics in literature, and adapt them to fit with the requirements of our scenario.

Some of these measures have already been discussed in section 2, within related works, and in some case adaptations have already been considered. To deal with the new quality paradigm given by diversity, quality metrics needed to express the idea of novelty, diversity, together with relevance and similarity.

Our quantitative tests adopted two evaluation metrics, derived from those found in [8] and [9]: α -DCG and MD-Recall.

4.1.1 α -DCG

The α -DCG metric, introduced in section 2.3.3, is an evolution of the Discounted Cumulative Gain (DCG) measure that also takes into account information novelty. Standard DCG is calculated as follows:

- *Cumulative gain* $CG[k]$ is computed as the sum of graded relevance values of all the results in a search-result list till position k .

$$CG[k] = \sum_{i=1}^k rel_i \quad (4.1)$$

where rel_i is the graded relevance of the result at position i

- $CG[k]$ is then discounted to penalize documents appearing lower in the ordered result set since they require more effort to be reached. This measure is called *discounted cumulative gain*:

$$DCG_k = \sum_{i=1}^k \frac{rel_i}{\log_2 i + 1} \quad (4.2)$$

- DCG is often normalized to allow comparison between computations with different queries and *normalized discounted cumulative gain* is obtained

$$nDCG[k] = \frac{DCG[k]}{IDCG[k]} \quad (4.3)$$

where $IDCG[k]$ is the ideal DCG at position k . $nDCG[k]$ ranges into $[0,1]$, being equal to 1 when the result set ordering is the ideal one.

α -DCG is computed in the same way, changing the gain function to add novelty as an additional feature [8]. α -DCG defines gain as:

$$G[k] = \sum_{i=1}^m J(d_k, i)(1 - \alpha)^{r_{i,k-1}} \quad (4.4)$$

Section 2.3.3 provides further details about Equation 4.4.

Table 4.1: Example of the result set model

Hotel	Museum	Restaurant
H1	M1	R1
H1	M1	R2
H2	M2	R3

We adapted α -DCG to the multi-domain case as follows: an information nugget is now related with relation identifiers, therefore m in Equation 4.4 is equal to the number of services invoked by the query; for instance, in the **Milan** set described section 3.5.1, $m=3$ since there are 3 entities, each one with an identification field. $J(d_k, i)$ is used in [8] to consider the assessor judgment about the presence of an information nugget. In our setting there is no need for an external judgment since the identity between service ids is evaluated and can't produce errors (ids can be either identical or different when compared one another), thus a safe assumption is to set $J(d_k, i) = 1$ for all services m .

$r_{i,k-1}$, that in the original work [8], it now corresponds to the number of document containing information nugget i till position $k - 1$, is interpreted in the multi-domain context as the number of times a tuple having a given identifier appeared in the result set till position $k - 1$.

According to these assumptions and adaptations, α -DCG at position k can be written as

$$\alpha DCG[k] = \sum_{k=1}^K \frac{\sum_{i=1}^n J(\tau_k, t_i)(1 - \alpha)^{r_{t_i, k-1}}}{\log_2(1 + j)} \quad (4.5)$$

The following example better details the application of α -DCG in our context. Consider table 4.1:

Computing DCG with $\alpha = 0.5$

$$\alpha DCG[1] = \frac{(1 - 0.5)^0 + (1 - 0.5)^0 + (1 - 0.5)^0}{\log_2 2} = 3 \quad (4.6)$$

$$\alpha DCG[2] = \alpha DCG[1] + \frac{(1 - 0.5)^1 + (1 - 0.5)^1 + (1 - 0.5)^0}{\log_2 3} = 4.26 \quad (4.7)$$

$$\alpha DCG[3] = \alpha DCG[2] + \frac{(1 - 0.5)^0 + (1 - 0.5)^0 + (1 - 0.5)^0}{\log_2 4} = 8.01 \quad (4.8)$$

From 4.6, 4.7 and 4.8 some considerations can be made:

- α -DCG[1] is always equal to m , the number of services, since there can't be any repetition on the first combination.
- The maximum increment at step k is $\frac{m}{\log_2(k+1)}$, when all the tuples composing the new combination are all diverse from the already-selected ones.

4.1.2 MD-Recall

The second metric used for the quantitative tests is suggested in [9], and derives from *recall*, one of the most important metric in information retrieval. Recall is defined as the fraction of documents relevant to the query that have been successfully retrieved:

$$Recall = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} \quad (4.9)$$

For instance $recall = 1$ if all the relevant documents have been retrieved. [43] adapts this metric to documents, considering for the first time the concept of subtopics.

It is not always obvious how to rank properly a set of documents, in order to grant a good recall measure based on the their content. Intuitively, documents containing different subtopics should be ranked higher and those covering already included subtopics should be penalized. In multi-domain search we can assimilate tuple combinations with subtopics, thus S-recall could be a suitable metrics. From [43]:

$$S - Recall[K] = \frac{|\cup_{i=1}^K subtopics(d_i)|}{n_a} \quad (4.10)$$

where $subtopics(d_i)$ is the set of subtopics to which document d_i is relevant and n_a is the number of subtopics for a certain topic T . Adapting 4.10 to multi-domain field we define *multi-domain recall* as:

$$MD - Recall_K = \prod_{i=1}^n \frac{|\cup_{k=1}^K R_i^k|}{|R_i|} \quad (4.11)$$

where R_i with $i = 1, \dots, n$ is a relation involved in a multi-domain query. MR-Recall measures for all the relations R_i the set of distinct tuples retrieved at position k in the result set compared to the entire population of relation $(|R_i|)$.

Considering once again table 4.1 as example and supposing all relations to have cardinality $|R_i| = 10$:

$$MD - Recall[1] = \frac{1}{10} * \frac{1}{10} * \frac{1}{10} = 0.001 \quad (4.12)$$

$$MD - Recall[2] = \frac{1}{10} * \frac{1}{10} * \frac{2}{10} = 0.002 \quad (4.13)$$

$$MD - Recall[3] = \frac{2}{10} * \frac{2}{10} * \frac{3}{10} = 0.012 \quad (4.14)$$

From 4.12, 4.13 and 4.14 some considerations can be made:

- $MD - Recall \rightarrow 1$ if the result set is complete, i.e. it contains $|R_i|$ different tuples for relation R_i
- Similar to α -DCG, the maximum increment is attained when a combination composed just by new tuples is chosen(consider equation 4.14 an example)

4.2 Design

Each algorithms described in section 3.4 was studied varying the λ parameter, so to analyze how λ effects their performance according to the α -DCG and MD-Recall measures. Our goal was to understand which performs better and to find a suitable parameter value that granted increased result set quality using diversity, while keeping relevance as high as possible. Value assigned to λ ranged between 0 and 1, with a 0.25 step, thus 5 configurations were analyzed for each approach.

$$\lambda = [0, 0.25, 0.5, 0.75, 1] \quad (4.15)$$

For each λ value both kinds of diversity, categorical and quantitative, were considered.

Categorical diversity was evaluated by ordering the datasets according to 3 combination attributes (distance, cost and rating for **Milan** dataset and price, quality index and distance for **University** one), and, then diversify them with the chosen algorithm. Results for the same algorithm and λ were then averaged to avoid query-dependent bias. A similar approach was followed for the quantitative diversity testing, where 6 experiments were executed for each λ and each algorithm: sorting and diversification attributes were taken from all possible permutations of the 3 ordering attributes used for categorical diversity. For the **Milan** dataset, "distance by car", "total cost" and "average rating" were chosen; in the **University** set we adopted as attributes "yearly price", "quality index" and "distance by car".

This preference of combinational attributes over single relation ones is due to the inherent diversity they provide to the result set: for instance, a starting set sorted by "total price" has a greater initial variety compared to one ordered by "hotel price", since in the latter case a cheap hotel could appear many times before a more expensive one is found.

While $MD - Recall$ is identical for some λ values and some initial ordering (it depends on the sorting attribute and eventually the one used for diversifying), α -DCG, depends also on the value of α . During α -DCG computation this relation was considered, therefore measurements were always performed for α values ranging as λ ones.

$$\alpha = [0, 0.25, 0.5, 0.75, 1] \quad (4.16)$$

As stated in section 4.1.1 DCG is often normalized for cross-query comparison. During this testing phase, the ideal gain was computed for each possible ordering.

Rather than showing the normalized version of α -DCG by itself, we will present it (in the next section) plotted along with the other algorithms, in order to have a better comparison between the maximum hypothetical value and the actual measure of diversified solutions. For the same reason, the un-diversified curve is plotted as well in relation with the diversified cases.

To retrieve the ordering with the maximum α -DCG value, i.e. the

ideal one, an iterative greedy algorithm was used as follows: for each iteration all remaining combination are analyzed and the one leading to the highest increment in α -DCG value is chosen, breaking ties according to the initial order. Within this approach, it is clear how the ideal sorting is not unique: any combination composed just by new tuples would imply the highest α -DCG increment (see section 4.1.1).

Given λ and α variation range, many experiments were possible. In the next section results from most relevant ones will be described. The charts of all performed tests can be found in appendix A.

4.3 Results

The evaluation compared algorithms performance according to different λ values. It is interesting to notice that the formulae for the analyzed algorithms (see section 3.4 for algorithms scoring functions) manage λ in different ways: to balance diversity and relevance in *MMR*, for instance, λ needs to be equal to 0.5; the other 3 algorithms, instead, require $\lambda = 1$ to obtain the same balanced effect. Both metrics were studied with this parameter selection to grant a proper comparison of their performances.

4.3.1 Results for α -DCG

Figures 4.1 and 4.2 show the performance on the whole set, according to the α -DCG metric, of the analyzed algorithms with $\lambda=0.5$ for *MMR*, $\lambda=1$ for *MaxMin*, *MaxSum* and *MaxCov*, and with $\alpha = 0.5$. This α value has been chosen from the literature (see Wang [40]), where 0.5 is suggested as the best value to grant maximal balance between relevance and diversity. This is intuitive from α -DCG formula (Equation 4.5) where large values of α give more importance to redundant combinations while $\alpha = 0$ (the minimum) totally ignores the set diversity.

Several comments can be made about charts in figures 4.1 and 4.2. As predictable, no algorithm was able to match the performance of the ideal α -DCG curve; moreover, the non-diversified set always provide worst performances, except in figure 4.1(a), where *MaxSum* appears to be worse than the initial set.

Before further discussion about algorithms comparison, a noticeable

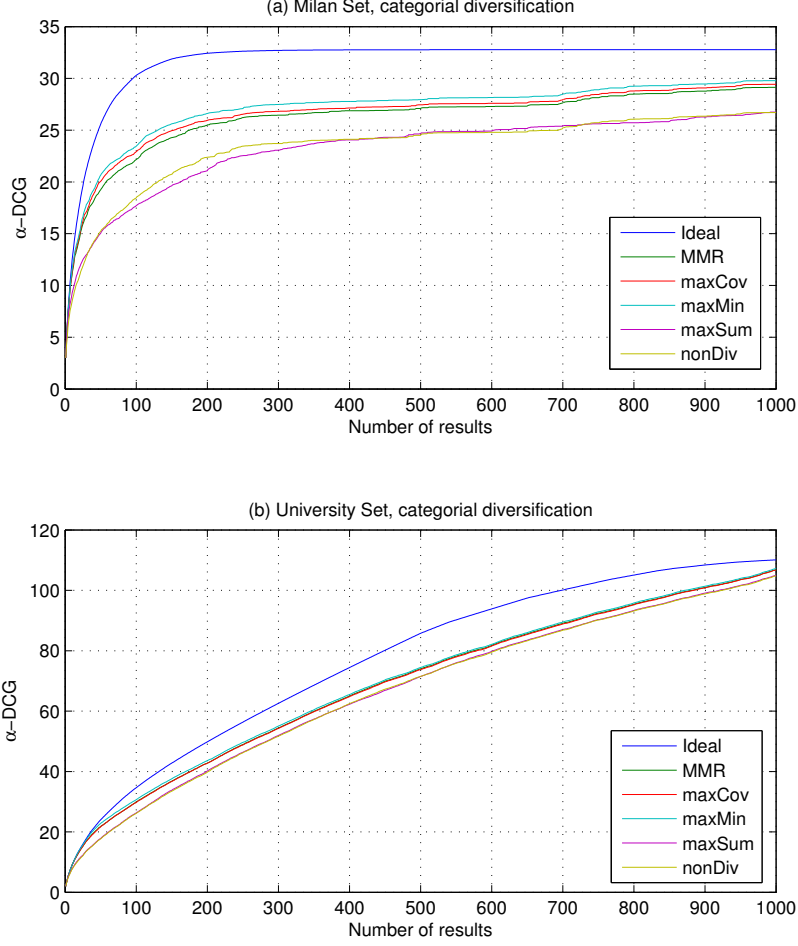


Figure 4.1: α -DCG ($\alpha = 0.5$), categorical distance, giving same weight to relevance and diversity.

difference between **Milan** and **University** sets can be highlighted: the former α -DCG increases till a certain value and then stabilizes, while the latter keep increasing till the last result. This is even more evident comparing ideal curves, where **Milan** one turns into an horizontal line after 200 result, while **University** one keeps a nearly constant slope. This can be explained by the dataset structure(section 3.5): the cardinality of **Milan** set relations is much lower than those in **University**(50 hotel/restaurant/museum with respect to 1100 flats and 58 universities). Therefore all the distinct values of the first set will be retrieved faster than the second ones,

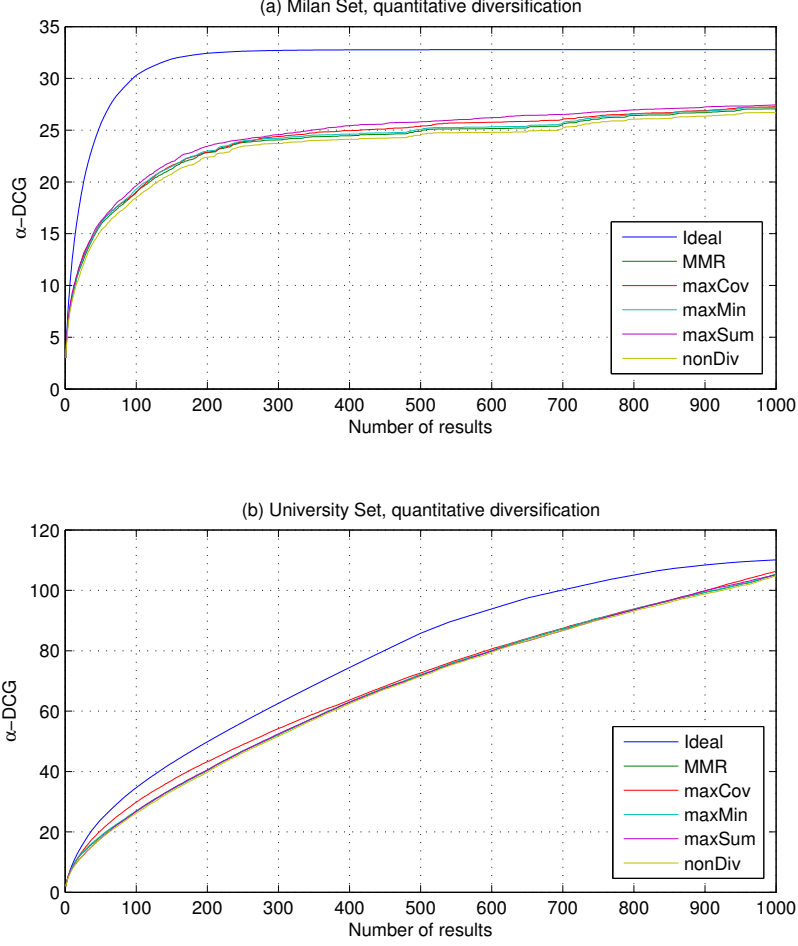


Figure 4.2: α -DCG ($\alpha = 0.5$), quantitative distance, giving same weight to relevance and diversity.

implying a strong reduction in α -DCG increment for **Milan** set after a certain amount of combinations(200 in this case).

Considering performances, with categorial diversification on **Milan** set (Figure 4.1(a)), the analyzed algorithms behave in most different ways. *MaxSum* gives poor results, comparable with the non-diversified curve, a problem already anticipated in section 3.4 due to its execution pattern. *MaxSum* selects for each iteration the couple of most distant and relevant results and each iteration ignores previous selections, thus the obtained set can be very similar to the initial one. In the user studies (Chapter 5), we will notice that this

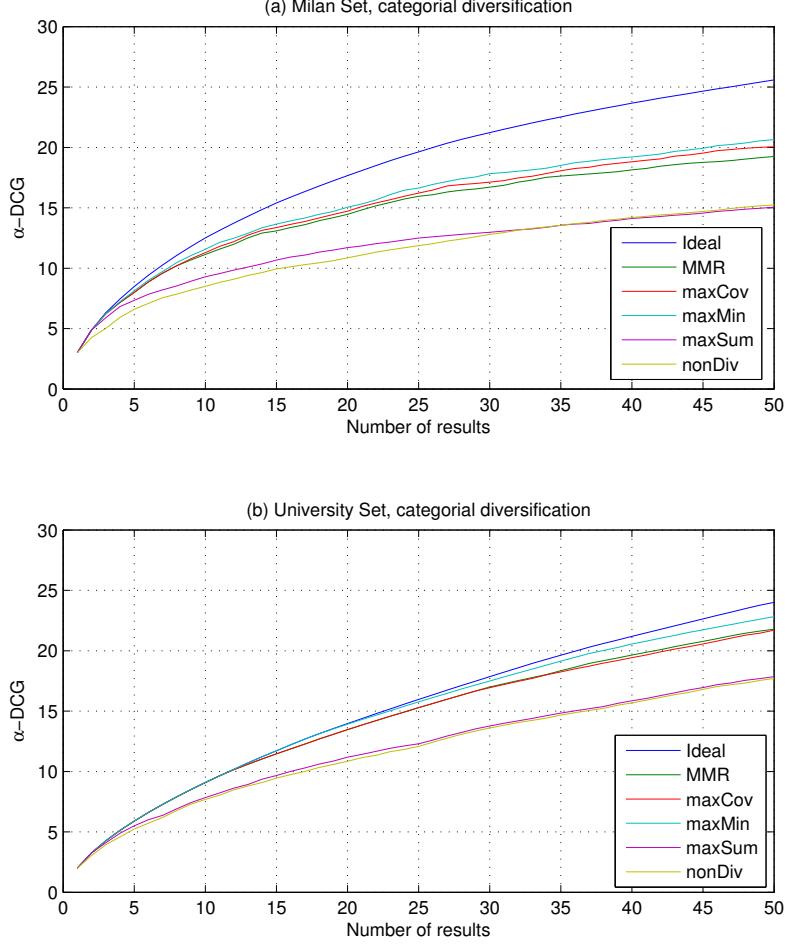


Figure 4.3: First 50 samples of α -DCG, categorical distance, giving same weight to relevance and diversity.

limitation is an highly perceived drawback of the algorithm. We remand to section 5.2.1 for further explanation.

According to figure 4.1(a), *MaxMin*, *MaxCov* and *MMR* behave similarly, improving in an evident way the initial set quality, with *MaxMin* providing scored better performance than the others.

Test on quantitative distance on the **Milan** set (Figure 4.2(a)) shown a noticeable degrade in performances; nonetheless, *MMR*, *MaxMin* and *MaxCov* curves, are still better than the baseline. The decrease in performance can be justified by the chosen metric, as α -

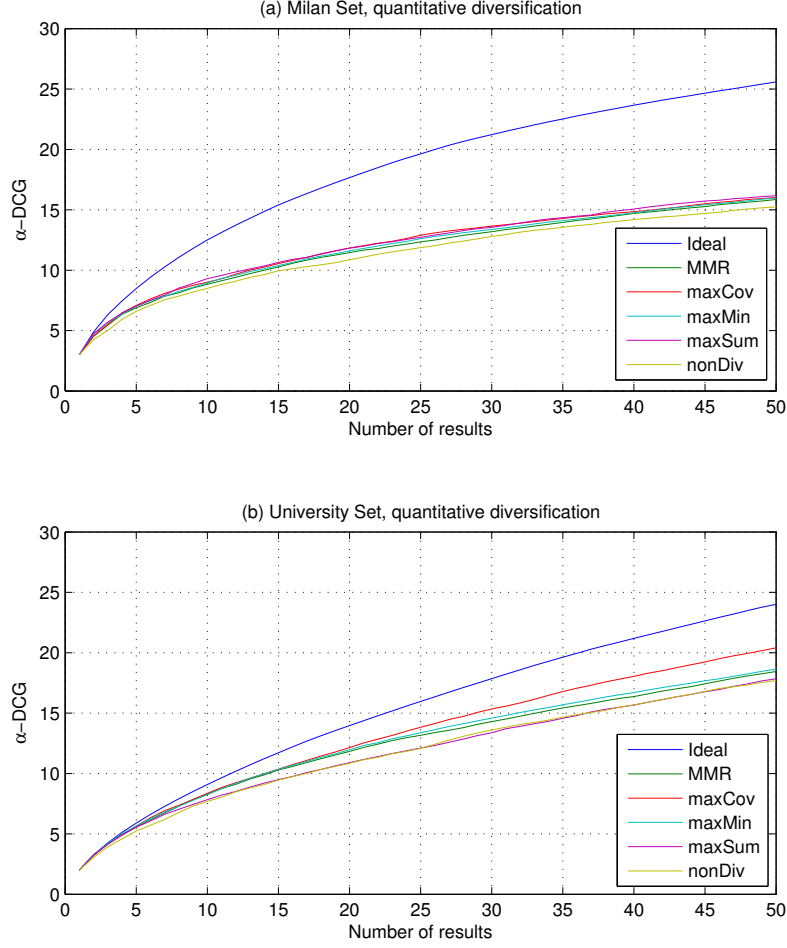


Figure 4.4: First 50 samples of α -DCG, quantitative distance, giving same weight to relevance and diversity.

DCG, which is based on the diversity of extracted objects, is perhaps more suitable to evaluate categorial diversification than quantitative one.

The α -DCG curves for **University** dataset (Figure 4.1(b) and 4.2(b)) are quite unexpected. It appears than no algorithm can improve significantly the initial set for both diversity metrics and types. In this case, diversification seems useless, if not harmful. This is probably due to the fact that these sets were already quite different before diversification took place.

In fact, it happened that diversification produced sets that could be considered worse than original ones. Most of the times this happened when the initial set were very fragmented and widespread.

An initial analysis on the dataset structure has been useful to determine whether diversification can be more specifically effective: tested algorithms perform better when the original set contains repetitive items with low variance of attributes values. Further studies in this direction (described in Section 6)) may help finding a threshold for the variance that softs the usefulness of diversification of a set. The logical conclusion is that in some cases, diversification algorithms don't improve the quality of the set.

Results described so far are related to figures 4.1 and 4.2, which show different algorithms comparison over the whole result set. From a user prospective it is highly unlikely to browse such an amount of data (1000 items - 20 pages): most users are used to Google paradigm and if the desired item is not found in the first few pages, the query is refined or a different search engine is used. Under this assumption, figures 4.3 and 4.4 and shows the same results considering only the first 50 combinations. In this case α -DCG values in **Milan** set (Figures 4.3(a) and 4.4(a)) do not show anything different from previous results (those considering 1000 combinations) while **University** ones (Figures 4.3(b) and 4.4(b)) give an improved overview: in figure 4.3(b), *MMR*, *MaxCov* and *MaxMin* show an ideal-like behavior from for the first 30 combinations. This positive trend is lost with quantitative diversity (Figure 4.4(b)), where all algorithms curves tend to the non-diversified case with a consequent loss of quality.

4.3.2 Results for MD-Recall

The results of the MD-Recall measure for our tests are shown in figures 4.5 and 4.6, where we can draw considerations similar to the ones expressed for α -DCG. In particular, the shapes of MD-Recall curves reflect α -DCG ones. **Milan** set has a step-like trend which means that in certain ranges the selected combination are composed just by already selected tuples therefore MD-Recall remains constant. **University** set, as stated in the previous section, is inherently more spread than **Milan** one; therefore MD-Recall keeps increasing with a slope similar to α -DCG.

Like for previous metric, **Milan** dataset gives the best results, even though it is difficult to identify a winner algorithm. Considering for

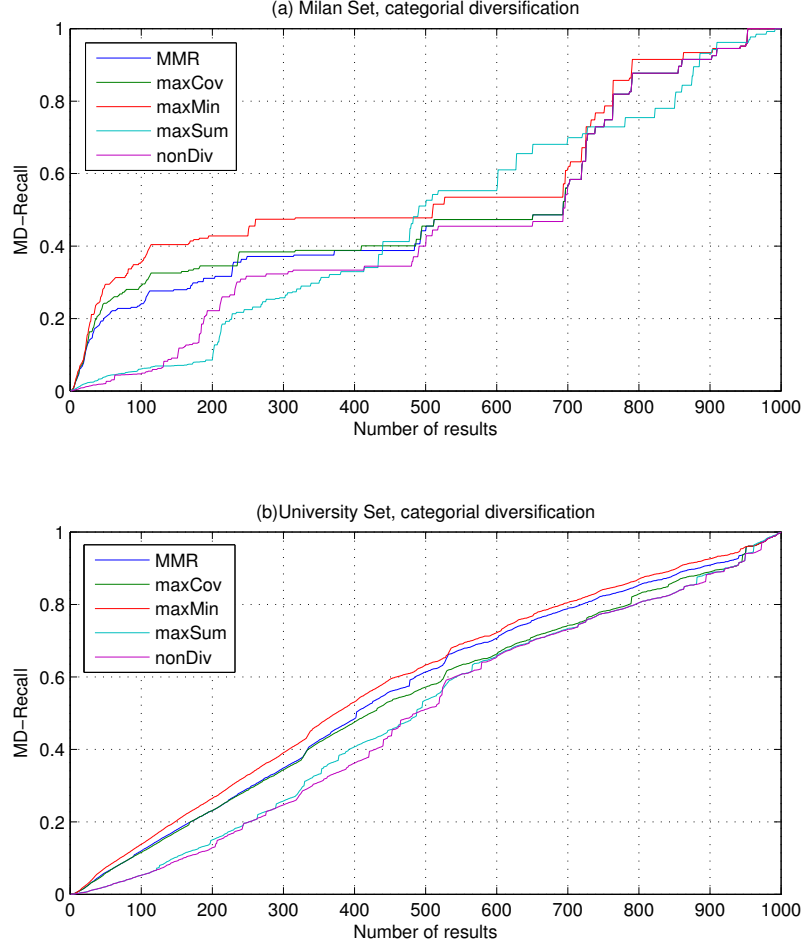


Figure 4.5: MD-Recall, categorical distance, giving same weight to relevance and diversity.

instance figure 4.5(a), all of them are better than others only when studied in certain ranges with *MMR* being the best one for the first 400 items and *MMR*, *MaxCov* and *MaxMin* giving the same identical positive results in the first combinations extracted. Looking figure 4.6(a), *MaxSum* unexpectedly outperformed all other algorithms (all similar to each other) but *MaxCov*. Like for α -DCG, we can assume that this behavior is somehow enhanced by the adopted metric, since MD-Recall is meant to compute objects recall rather than single relation attributes. MD-Recalls computed on **University** set (Figures 4.5(b) and 4.6(b)) are, for reasons similar to the

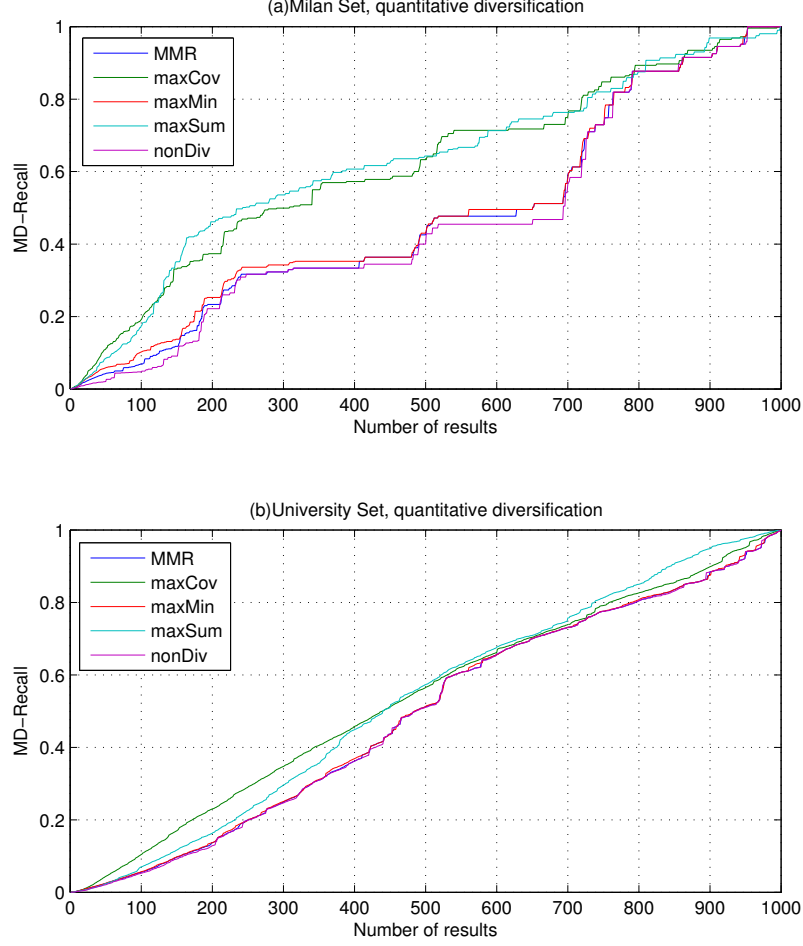


Figure 4.6: MD-Recall, quantitative distance, giving same weight to relevance and diversity.

ones explained for α -DCG, difficult to analyze but in the categorical case we can notice *MMR* and *MaxMin* as best approaches.

Considering the user prospective, figures 4.7 and 4.8 show MD-Recall values for the first 50 combination on both dataset. These results are more significant than for the previous metric: is it now possible to identify the best performing algorithm on these reduces sets. When applying categorical distance to **Milan** set (Figure 4.7(a)), *MMR*, *MaxCov* and *MaxMin* all attain a good MD-Recall in comparison with *MaxSum* and the baseline. *MaxMin* is the best one, while *MaxCov* gives the best result for quantitative distance (Fig-

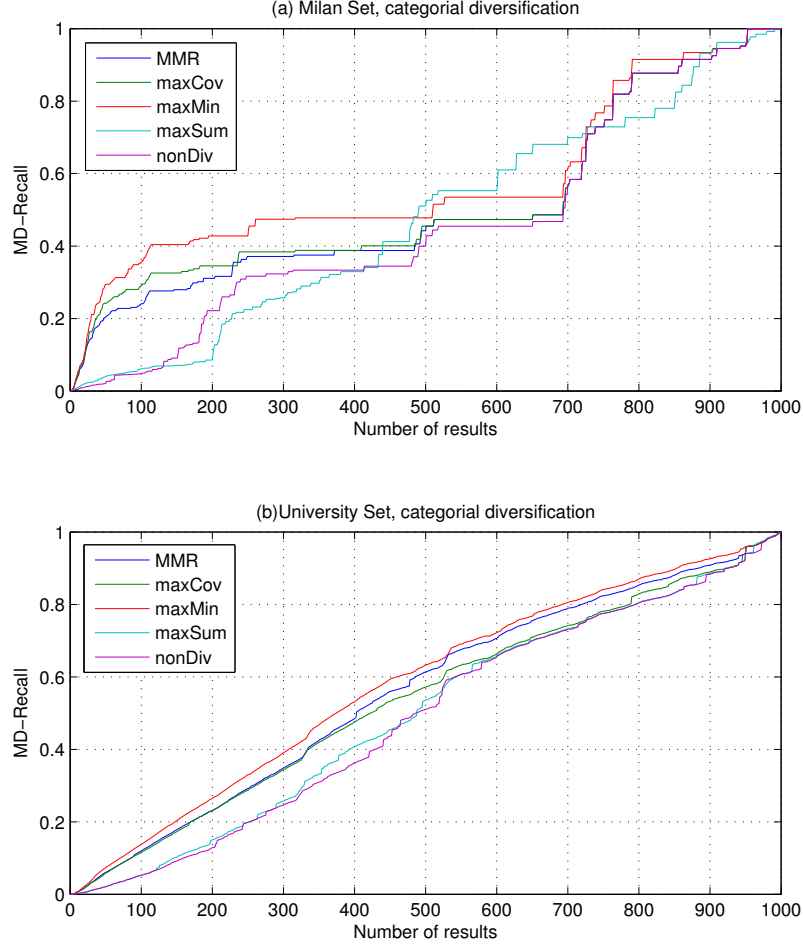


Figure 4.7: First 50 samples of MD-Recall, categorical distance, giving same weight to relevance and diversity.

ure 4.7(a)). It is also interesting to notice how the high level performances *MaxSum* had for the whole **Milan** set in quantitative case (Figure 4.6(a)) are not matched when considering just the first results.

Analyzing now the MD-Recall of different approaches on the first 50 elements of **University** set (Figures 4.7(b) and 4.8(b)), *MaxMin* and *MaxCov* seem the most suitable ones for categorical and quantitative distance respectively. This behavior is the same seen for **Milan** set (Figures 4.7(a) and 4.8(a)) which lead to think that these algorithms should actually improve result set variety from a user

prospective.

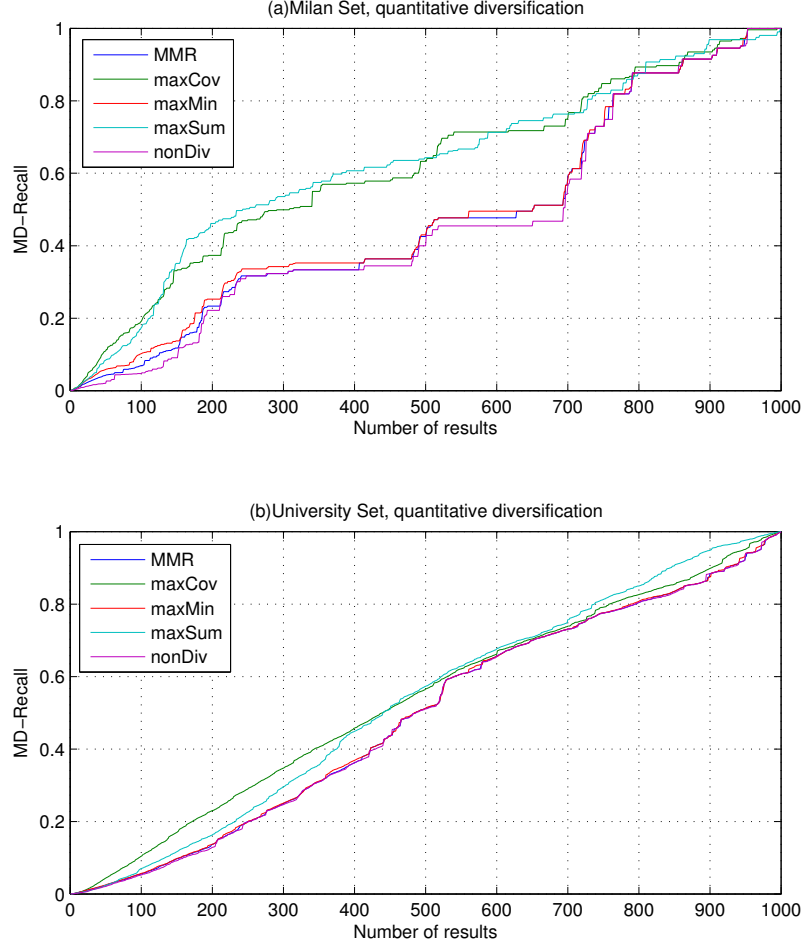


Figure 4.8: First 50 samples of MD-Recall, quantitative distance, giving same weight to relevance and diversity.

4.3.3 Comments

Analyzing the results, we can state that it is not possible to identify an overall best algorithm among the analyzed ones. Given the results of quantitative analysis and the additional charts in the appendix A, *MMR*, *MaxMin* and *MaxCov* on average seem to increase the result set quality more than *MaxSum*. The baseline is most of the time improved by diversification algorithms.

Categorical diversity seems to give better results than quantitative one, where results are often controversial. An a-priori knowledge of the data structure (especially relation cardinality) would help to identify situations where diversification is worthy, and which is most suitable approach the initial set can benefit from.

Chapter 5

Qualitative Test

In order to validate the quantitative results, we conducted two user study experiments.

We designed and implemented two running Web-based prototypes that aimed at testing, in a real-world scenario, 1) if diversification was perceived as an added value by users, and 2) if quantitative results are supported by users perception.

The first experiment (*User Test 1*) was conducted on more than 80 users in a population of "experts" in the search engine domain. The second experiment (*User Test 2*), instead, included 70 undergraduate students from Politecnico di Milano. In both experiments, users interacted with both datasets shown in section 3.5.

Each experiment was composed of the two interaction scenarios explained in 3.5, respectively called *Night Out* (N.O.) and *Study Abroad* (S.A.).

Given that quantitative analysis provided evidence that *MMR* and *MaxMin* algorithms clearly outperformed the baseline when adopting categorial distance, we decided to consider only categorial distances for the qualitative tests. Therefore, from now on the term "diversification" refers to binary diversification, meaning the categorial one over identification attributes. Moreover, since *MaxMin* performances were comparable to *MMR*, experiments conducted evaluated only *MMR* and *MaxSum* with respect to the un-diversified baseline.

The rest of the chapter is organized in two sections: section 5.1 provides details about the experiments design, while section 5.2 details and comments the results obtained from the experiments.

5.1 Design

5.1.1 User Test 1

The aim of the first experiment was to sort out, in a direct comparison between diversified and non-diversified sets, which one would have been perceived as “better” by users.

Data Set Preparation

Since the original datasets had different cardinality and contained spurious items, we decided to prune them into smaller sets. We therefore removed tuples which could have perceived as less relevant from a domain-specific point of view; for instance, we removed from the S.A. datasets tuples referring to less known universities, and trimmed from the N.O. dataset the less-known museums. This operation allowed us to minimize a judgment bias based on the poor performances of the underlying search system.

After sorting the **Milan** dataset by “total price” and the **University** dataset by “total distance”, we cut down the retrieved combinations. This allowed us to deal with two sets containing exactly 5’000 items each. To further avoid biases due to search engine performance, we randomly composed 10 sets of 1’000 combinations each. These sets were then subject to a second sorting using the same ranking function obtaining the final 10 sets for the un-diversified results. We then created 20 diversified sets for each scenario, applying *MMR* or *MaxSum* algorithms. After diversification, we kept only the first 10 combinations and sorted them using “total price” or “total distance” to have comparable sets.

Strategies to avoid evaluation ordering bias

To avoid ordering bias, we adopted a round robin technique to decide which of the 10 result sets had to appear in each user experiment, according to the adopted algorithm (un-diversified, *MMR*, *MaxSum*). Moreover, we randomly chose the order in which datasets were presented according to the evaluated scenario (N.O. or S.A.).

Additionally, we designed the prototype to randomly chose which of the two scenarios should have been presented first for each attending user. Each step is now described in detail and sided by the screenshot of the actual page.

The prototype interface was kept simple and effective, providing only the information needed by users to make their choice. We adopted a presentation strategy based on a tabular view. We decided to show the compared sets in different tabs, to underline the differences between them.

User test interaction design

The user interface is composed of five screens.

Step 1) Welcome Page. Short page to welcome users to the qualitative test. It was mainly used for logging purposes. A click on the link at the center of the page (Figure 5.1) opened a full-screen pop-up with no navigation control, thus preventing users to undo/redo their actions.



Start the test

Figure 5.1: Welcome page for the first user test

Step 2) Introduction. An introduction to the experiments (Figure 5.2), explaining from a general point of view the SeCo context, the scenarios and what is required to the users. There was also an identification box to retrieve information needed in the post-processing phase, to associate logged data to each user.

Step 3) First scenario. Under the hypothesis of a random choice of the order in which the scenario ("Night Out" or "Study Abroad") would have been presented, the round robin selection of the sets and the random order in the tabs, this and the next step were the most important ones. The user was introduced to the real test (Figure 5.3) by a simple description of the scenario, where it was underlined that the choice had to represent "The best set in terms of *quality* and *variety*." Assuming that the user received first two Milan sets, the tabular view allowed him

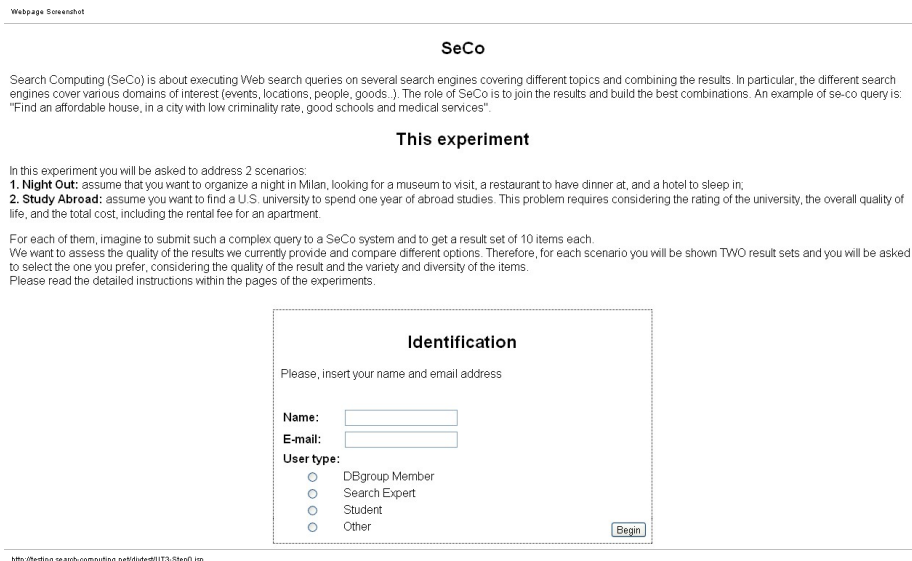


Figure 5.2: Introductory page for the first user test

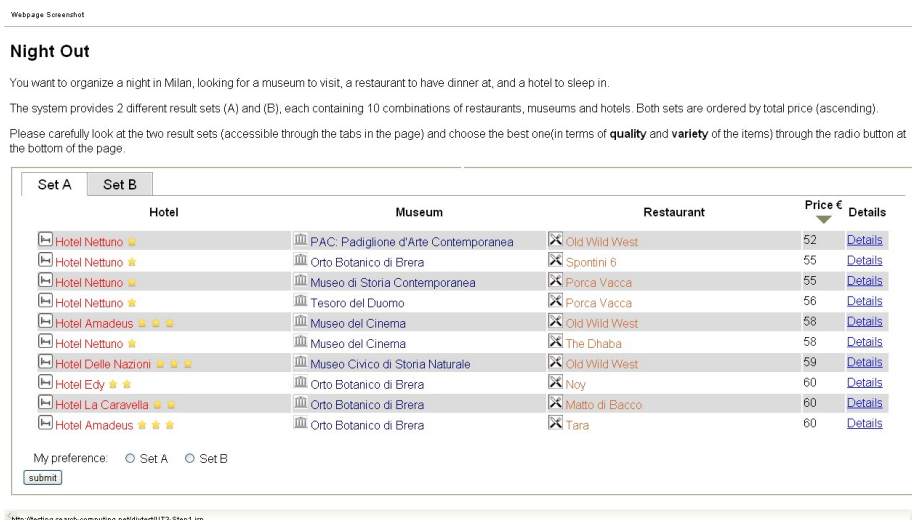


Figure 5.3: First scenario presentation and tabular view of the first set

to switch from one to the other easily. We kept the layout of the page as simple as possible, keeping its comprehension immediate.

To express their preferences, a two radio button form had been placed just below the data tables. As picture 5.4 shows, all the information needed to make the comparison were directly placed under the users view, divided by field of interest (Hotel, Museum or Restaurant domain). If a user wanted more

information, the "Details" link on the right side of each term would open a pop-up with all the information available on the selected item. The details information are shown in Picture 5.5

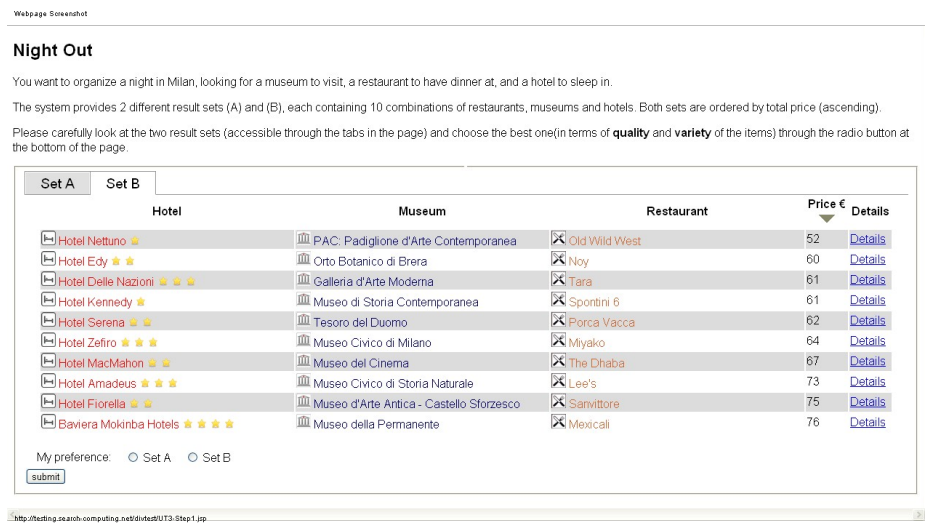


Figure 5.4: Second set in the comparison for the first scenario

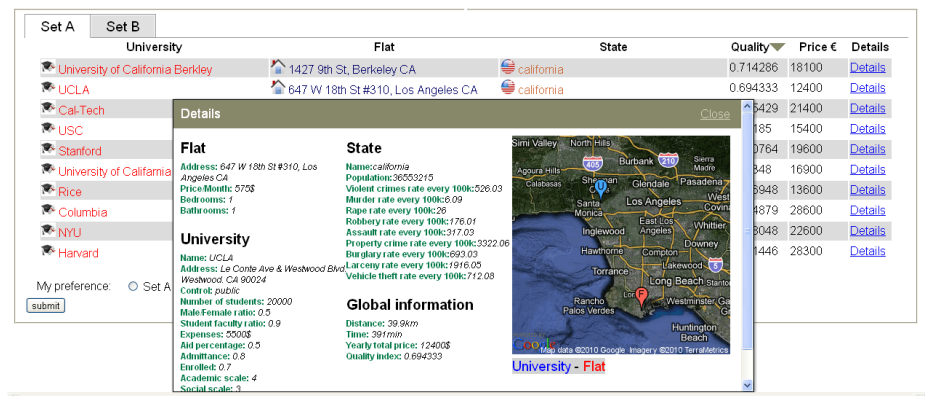


Figure 5.5: Screenshot of the details available for each item in the list

The detail box enforced the separation of domain-specific information, highlighting the field information belong to (Figure 5.5 "Flat" - "University" - "State"), plus the "global" information characterizing the overall term (i.e. "Total cost", "total distance", "quality index").

Step 4) Second scenario. As in the previous step, the user was introduced to his task (Figure 5.6) with a short description of the scenario (in this case "Study Abroad" referred to University sets).

Study Abroad

You want to find a U.S. university to spend one year of abroad studies. This problem requires considering the rating of the university, the overall quality of life, and the total cost, including the rental fee for an apartment.

The system provides 2 different result sets (A) and (B), each containing 10 combinations of university, apartment and US state details. Both sets are ordered by an aggregate quality index that considers the university rating, the "walkability" of the area, the distances between the objects and the crime rate of the state.

Please carefully look at the two result sets (accessible through the tabs in the page) and choose the best one(in terms of **quality** and **variety** of the items) through the radio button at the bottom of the page.

Set A

Set B

University	Flat	State	Quality▼	Price €	Details
University of California Berkley	1427 9th St, Berkeley CA	california	0.714286	18100	Details
University of California Berkley	2425 California St #5, Berkeley CA	california	0.714024	18700	Details
University of California Berkley	1206 Parker St, Berkeley CA	california	0.713808	17500	Details
University of California Berkley	2070 University Ave, Berkeley CA	california	0.713043	17440	Details
University of California Berkley	2070 University Ave, Berkeley CA	california	0.713043	17200	Details
University of California Berkley	2727 Haste St, Berkeley CA	california	0.712827	12400	Details
UCLA	647 W 18th St #310, Los Angeles CA	california	0.694333	12400	Details
UCLA	410 S Sierra Madre Blvd #9, Pasadena CA	california	0.693373	17440	Details
UCLA	49 S Grand Oaks Ave #3, Pasadena CA	california	0.693069	14260	Details
UCLA	1185 E Orange Grove Blvd, Pasadena CA	california	0.692847	15100	Details

Set A

Set B

University	Flat	State	Quality▼	Price €	Details
University of California Berkley	1427 9th St, Berkeley CA	california	0.714286	18100	Details
UCLA	647 W 18th St #310, Los Angeles CA	california	0.694333	12400	Details
Cal-Tech	11611 Darlington Avenue	california	0.685429	21400	Details
USC	647 W 18th St #302, Los Angeles CA	california	0.68185	15400	Details
Stanford	19140 Stevens Creek Boulevard, Cupe	california	0.680764	19600	Details
University of California San Diego	2050 Pacific Beach Dr, San Diego CA	california	0.63348	16900	Details
Rice	9701 Stella Link Rd #176, Houston TX	texas	0.616948	13600	Details
Columbia	407 Jerusalem Ave, Hempstead NY	newyork	0.584879	28600	Details
NYU	28 Elizabeth St, Rye NY	newyork	0.538048	22600	Details
Harvard	268 Grove St #5, Auburndale MA	massachusetts	0.531446	28300	Details

My preference: ☐ Set A ☒ Set B

submit

Figure 5.6: Comparison of the two sets for the Study Abroad scenario

Step 5) Thanks. A page (Figure 5.7) greeting the user for attending the test and thanking all the participants.

Thanks for attending our test

Figure 5.7: A simple thanking page to greet the attending users

5.1.2 User Test 2

The second user test has been designed to study if (and how) diversification can be useful to users in a search scenario. Users were asked to inspect the results sets of both scenario and select the best three results between the available ones. For the N.O. scenario, we asked users to select the best three combinations of Hotel-Restaurant-Museum they could find; in the S.A. scenario, instead, users were asked to identify the best three combinations of University-Flat-State among proposed ones.

The idea was to conduce an implicit test, where the measured variables were the number of page-result navigations, and the duration of the task. The intuition was that, for a diversified result set, the execution of the task would have been easier and quicker.

Data Set Preparation

We selected from the initial database only tuples containing major universities or well-known museums (as in the first user test). We then sorted the combinations of the N.O. data-set by total distance, and the S.A. dataset by total price. We then filtered combinations according to the total distance attributes (around 5 kilometers for the first set and 20 miles for the second set), so to retrieve only the best 1'000 combinations. These first 200 items in the resulting combinations composed the un-diversified experiment dataset. To compose the diversified sets, we applied *MMR* and *MaxSum* binary algorithms to the 1000 combinations and kept only the first 200 terns. During the experiments we presented these 200 objects divided in 20 pages of 10 items each.

With the 6 constructed sets (3 for the Milan experience and 3 for University case) we adopted the same round robin technique for algorithms distribution explained in the previous section to avoid any learning or ordering biases.

User test interaction design

The test was composed of 8 steps, including questionnaires, user task pages and explanation pages. A logging system has been put in place to record task durations, users' clicks, and questionnaires answers.

We now present a description of each step, sided by a screen-shot of the related web page.

Step 1) Identification. Since the test had been proposed to students from the same campus, in the first step users had to specify their personal identification number (Figure 5.8). As in the previous test, the actual task is performed in a pop-up window with no backward navigation.

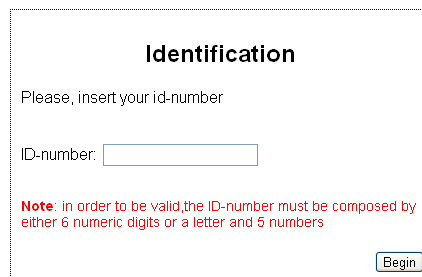


Figure 5.8: Users need to identify themselves to proceed

Step 2) Introduction. A brief explanation (Figure 5.9) of the SeCo project and the context where it operates is followed by a short description on the test and what the user will be asked during this experience.

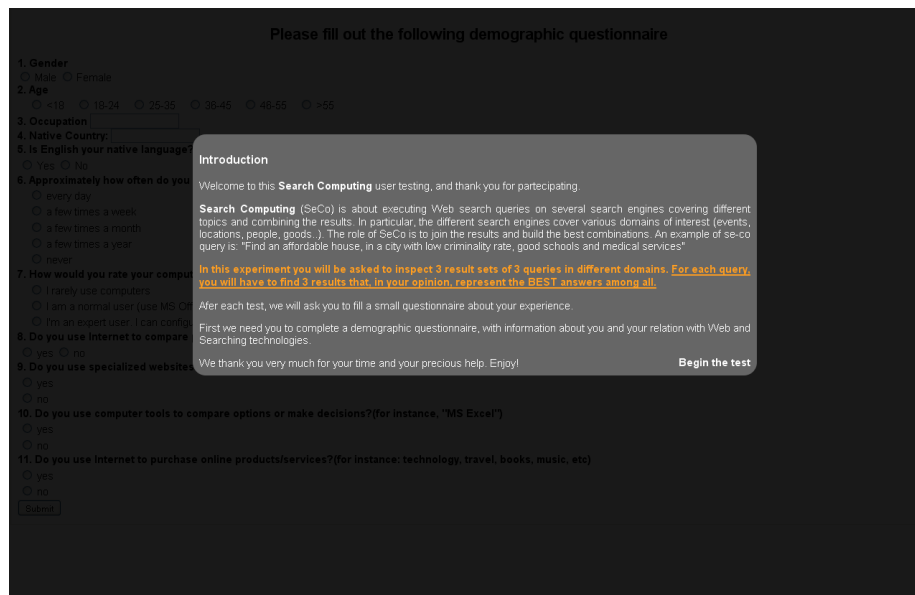


Figure 5.9: A short introductory page was presented to explain the test and the context

Users needed to fill a short demographic questionnaire (Figure 5.10); this step was crucial to identify the sample population and eventually apply post-process filtering.

Step 3) Rome Tutorial. To help users to familiarize users with the system, we included a tutorial step to allow users to practice with the features and the visual appearance of the prototype. The system included a "shopping-cart" widget, where users select items according to their preferences and proceed to the "check-out" only when ready. Similarly, we asked them to chose three combinations and proceed to the next step only when satisfied by the items placed in the "cart" (Figure 5.12 top right of the page). The tutorial was based on Rome dataset (3.5) and consisted in navigating through the 20 pages of combinations, selecting the best three combinations, while getting confident with the system. An introduction page (Figure 5.11) was provided to help users during this initial phase.

Please fill out the following demographic questionnaire

1. Gender
☐ Male ☐ Female
2. Age
☐ <18 ☐ 18-24 ☐ 25-35 ☐ 36-45 ☐ 46-55 ☐ >55
3. Occupation
4. Native Country:
5. Is English your native language?
☐ Yes ☐ No
6. Approximately how often do you use a computer?
☐ every day
☐ a few times a week
☐ a few times a month
☐ a few times a year
☐ never
7. How would you rate your computer abilities?
☐ I rarely use computers.
☐ I am a normal user (use MS Office, surf the web).
☐ I'm an expert user. I can configure a system and/or write pieces of software
8. Do you use Internet to compare products/services/prices?
☐ yes ☐ no
9. Do you use specialized websites to compare products?(for instance: kelkoo.com, pricegrabber.com, eprice.it, tripadvisor.com, etc)
☐ yes
☐ no
10. Do you use computer tools to compare options or make decisions?(for instance, "MS Excel")
☐ yes
☐ no
11. Do you use Internet to purchase online products/services?(for instance: technology, travel, books, music, etc)
☐ yes
☐ no

Figure 5.10: Users were asked to fill a short demographic survey

As figure 5.12 shows, the page structure is extremely simple: a header containing overall information, a sidebar with selected tuples plus the legenda, and a real content frame set in the middle of the page.

The upper part presents useful orientation information (always available during the test): the personal id, the current step number and a hint on the task to complete. The sidebar is split between the above-mentioned "cart", where users' choices will be placed once selected, and the legenda, useful to understand the icons and label used for the items descriptions.

A tabular view has been chosen to present the actual content

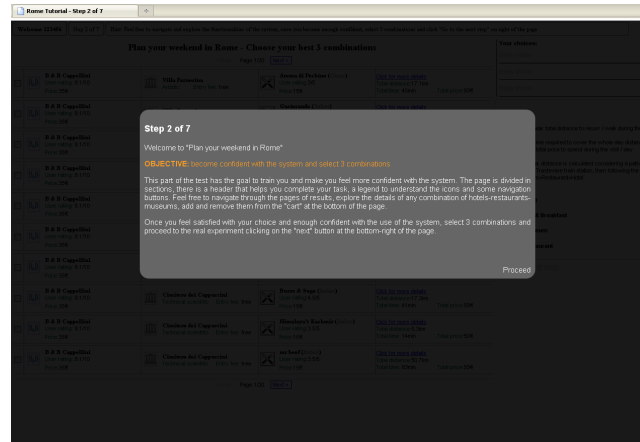


Figure 5.11: Introduction to the Rome tutorial

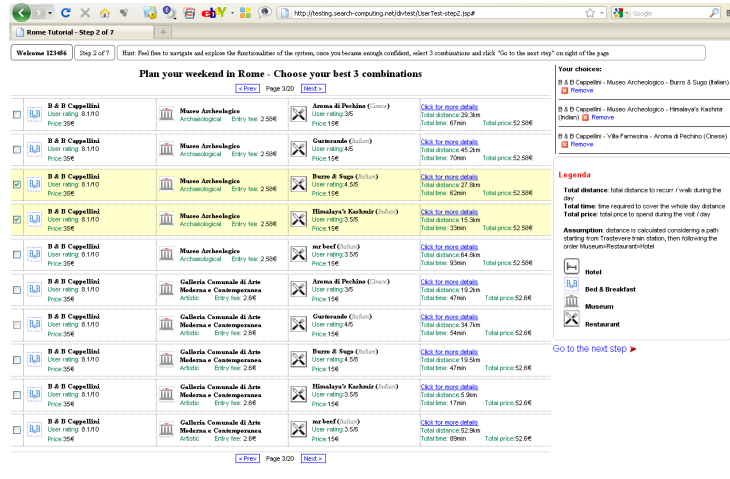


Figure 5.12: View of the selection page for the tutorial

in a clear manner. The 200 items of each scenario were proposed to users in 20 pages, 10 items each. Each combination was presented as in the previous test, showing the most important information first: "user rating", "categories", "price" of the single entity and the global one, the "total distance" and "time". More detailed information were accessible through the use of the "Click for more details" link, as shown in figure 5.15. Whenever a user tried to select more than 3 combinations, a warning message was displayed (Figure 5.13).

Step 4) Milan experience Once the user had become familiar with the system and felt enough comfortable to proceed, the first task (over Milan dataset) began. As usual, an introduc-

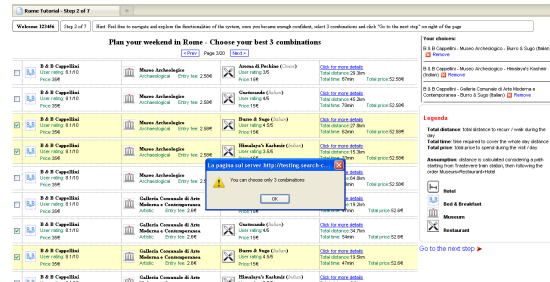


Figure 5.13: Warning message for managing exceptions

tory page have been set up to explain the task and its objective (Figure 5.14).

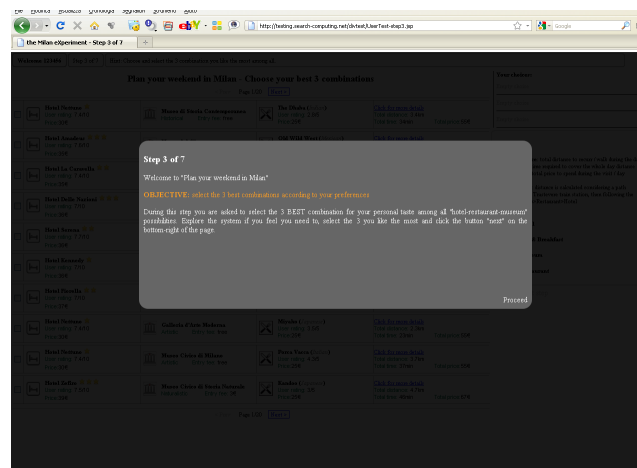


Figure 5.14: Introductory page for the first task (Milan experience)

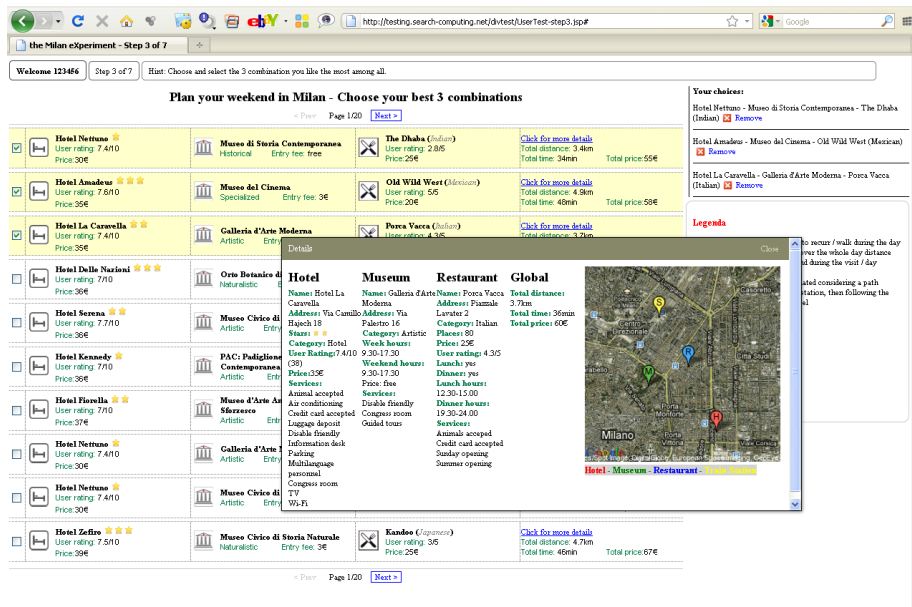


Figure 5.15: Detailed view of selected combination

Step 5) Milan questionnaire After performing the tutorial and the first task, users were asked to fulfill a short questionnaire to evaluate their degree of satisfaction and the correctness of the system. A 7-steps Linkert scale has been adopted for this evaluation, ranging from "Strongly Agree" to "Strongly Disagree" judgments (Figure 5.16). The questions investigated these aspects: the user understanding of the task he was asked to do, previous knowledge on the domain, the degree of confidence for the three choice made for the task and the quality of the system in terms of variety of displayed objects

The screenshot shows a questionnaire titled "Please fill out the following questionnaire about the completed task". It contains four questions with radio button options for "Strongly Agree", "Agree", "Agree Somewhat", "Undecided", "Disagree Somewhat", "Disagree", and "Strongly Disagree".

- I understood the task I was asked to do
- I knew enough about hotels, restaurants, and museums to accomplish this task
- I am confident that I have found the best possible choices from the result list
- The system provided a good variety of information about the proposed objects

Submit

Figure 5.16: View of the questionnaire after Milan experience

Step 6) University experience The second task had the same

specification as previous one (select the best 3 combinations among the proposed ones), but different settings. As explained in the introduction of this chapter, for the second user experience of this test we proposed the S.A. scenario. Similarly to the previous task, 10 combinations were showed per page, and presented in a tabular view (Figure 5.17).

the University experiment - Step 5 of 7

Welcome 123456 Step 5 of 7 Next: Choose and select the 3 combination you like the most among all.

Plan a one year study period in the U.S. - choose your best 3 combinations

Page 1/20 Next >

<input type="checkbox"/>	Arizona State Tuition fee: \$5008 Academic scale: ★★☆☆	1349 E Thomas Rd, Phoenix AZ Monthly rent: 1729 Bedroom: 1	arizona Population: 6338755 Crime rate/100k: 513.02	Click for more details Quality index: 0.40471 Distance: 3.7km Yearly price: 76496
<input type="checkbox"/>	northwestern Tuition fee: \$5009 Academic scale: ★★☆☆	Northampton St, Boston MA Monthly rent: 2008 Bedroom: 1	massachusetts Population: 6448755 Crime rate/100k: 458.09	Click for more details Quality index: 0.414181 Distance: 6.8km Yearly price: 79006
<input type="checkbox"/>	UCLA Tuition fee: \$5008 Academic scale: ★★☆☆	12584 Venice Blvd, Los Angeles CA Monthly rent: 2009 Bedroom: 1	california Population: 3853215 Crime rate/100k: 526.03	Click for more details Quality index: 0.483045 Distance: 6.8km Yearly price: 79006
<input type="checkbox"/>	Florida State Tuition fee: \$5008 Academic scale: ★★☆☆	681 Chapel Dr, Tallahassee FL Monthly rent: 3008 Bedroom: 2	florida Population: 18251243 Crime rate/100k: 708	Click for more details Quality index: 0.496757 Distance: 1.8km Yearly price: 91006
<input type="checkbox"/>	Arizona State Tuition fee: \$5009 Academic scale: ★★☆☆	1081 N 9th St, Phoenix AZ Monthly rent: 3259 Bedroom: 1	arizona Population: 6338755 Crime rate/100k: 513.02	Click for more details Quality index: 0.423458 Distance: 1.3km Yearly price: 94006
<input type="checkbox"/>	Florida State Tuition fee: \$5008 Academic scale: ★★☆☆	1656 McArthur Ave, Tallahassee FL Monthly rent: 3259 Bedroom: 1	florida Population: 18251243 Crime rate/100k: 708	Click for more details Quality index: 0.496842 Distance: 2.2km Yearly price: 94006
<input type="checkbox"/>	Arizona State Tuition fee: \$5008 Academic scale: ★★☆☆	1421 E Roosevelt St #2, Phoenix AZ Monthly rent: 3259 Bedroom: 1	arizona Population: 6338755 Crime rate/100k: 513.02	Click for more details Quality index: 0.42394 Distance: 2.2km Yearly price: 94006
<input type="checkbox"/>	Florida State Tuition fee: \$5008 Academic scale: ★★☆☆	2218 Magnolia Cir, Tallahassee FL Monthly rent: 3929 Bedroom: 4	florida Population: 18251243 Crime rate/100k: 708	Click for more details Quality index: 0.499943 Distance: 2.8km Yearly price: 97006
<input type="checkbox"/>	Arizona State Tuition fee: \$5008 Academic scale: ★★☆☆	1317 W Monroe St #C, Phoenix AZ Monthly rent: 3929 Bedroom: 1	arizona Population: 6338755 Crime rate/100k: 513.02	Click for more details Quality index: 0.423367 Distance: 1.5km Yearly price: 102406
<input type="checkbox"/>	Florida State Tuition fee: \$5008 Academic scale: ★★☆☆	1285 S Meridian Dr, Tallahassee FL Monthly rent: 4029 Bedroom: 1	florida Population: 18251243 Crime rate/100k: 708	Click for more details Quality index: 0.49994 Distance: 2.4km Yearly price: 103006

Page 1/20 Next >

Your choices:
Empty choice
Empty choice
Empty choice

Legend
Quality index: an index of the combination quality considering university academic rate, walkability, crime rate and distance between university and flat
Distance: distance by foot between flat and university
Yearly price: total price for one year, considering both the university fee and the flat rent

University
 Flat
 U.S. state

Go to the next step

Figure 5.17: Example of a University experience task web page

Once again, a detailed view of each item was available (Figure 5.18), showing most of the attributes explained in section 3.5 for this dataset.

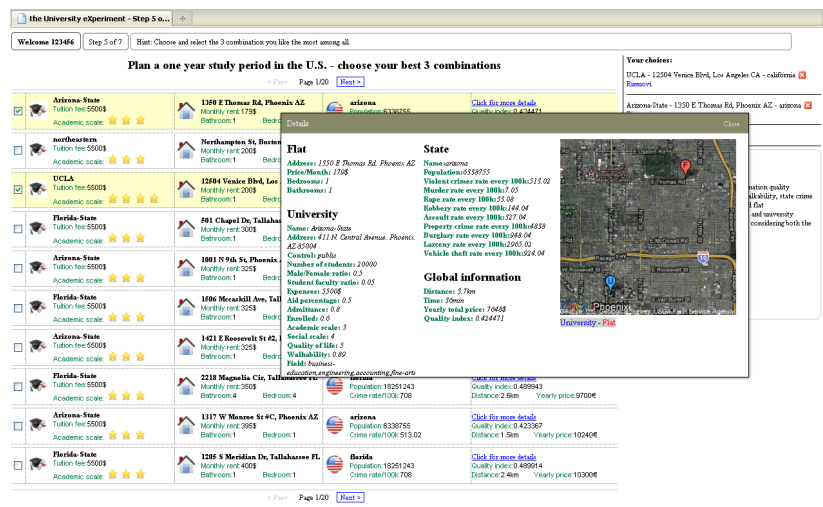


Figure 5.18: Detailed view for a combination of the University set

Step 7) University questionnaire We asked the users to fulfill the same questionnaire as in the Milan experience (Step 5) to understand if the change in of domain field had influenced the system applicability, users behavior or the importance of diversification.

Step 8) Final questionnaire A final questionnaire (Figure 5.20) was presented to the user, introduced by a short explanation (Figure 5.19). Its aim was to understand the overall sense of satisfaction and the goodness of the experience and the system to improve it if results were not positive.

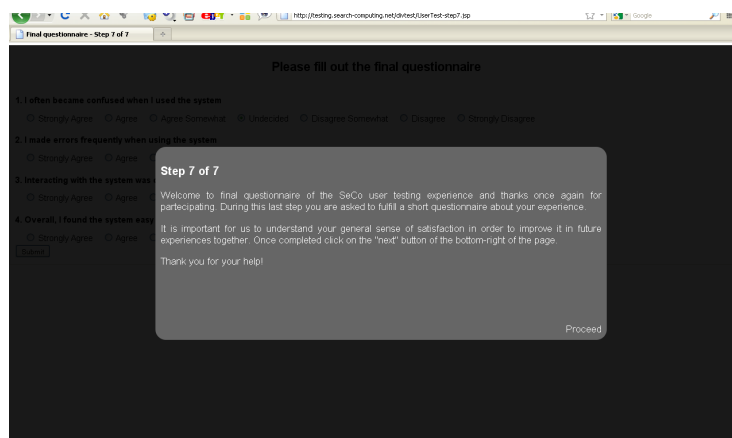


Figure 5.19: Introductory step for the final questionnaire

Final questionnaire - Step 7 of 7

Please fill out the final questionnaire

1. I often became confused when I used the system

☐ Strongly Agree ☐ Agree ☐ Agree Somewhat ☐ Undecided ☒ Disagree Somewhat ☐ Disagree ☐ Strongly Disagree

2. I made errors frequently when using the system

☐ Strongly Agree ☐ Agree ☐ Agree Somewhat ☐ Undecided ☒ Disagree Somewhat ☐ Disagree ☐ Strongly Disagree

3. Interacting with the system was often frustrating

☐ Strongly Agree ☐ Agree ☐ Agree Somewhat ☐ Undecided ☒ Disagree Somewhat ☐ Disagree ☐ Strongly Disagree

4. Overall, I found the system easy to use

☐ Strongly Agree ☐ Agree ☐ Agree Somewhat ☐ Undecided ☒ Disagree Somewhat ☐ Disagree ☐ Strongly Disagree

Submit

Figure 5.20: Final questionnaire

A simple thanking page (Figure 5.21) informed users that the experience was concluded and they had completed the test successfully.

Thanks for attending our test

Figure 5.21: Thanking page to greet attending users

Both tests were optimized to be run using Mozilla Firefox, but have been tested also on Google Chrome, Microsoft Internet Explorer and Safari. Even though were designed to be Web based, we are also aware of tests performed on mobile platforms, such as Apple iPhone and Windows Phone 7.

5.2 Results

In this section, main results from the first and second user tests are presented. For the first experiment, stored information considered only the user choices between the two compared sets. The main goal was to prove that users would have benefit from diversification, and that combinations variety would have been perceived as an important value in the decision making process. The latter experiment, whose aim was to prove that, even without direct comparison, user would have saved time and clicks when operating over diversified set, registered on logging files all user clicks. We remand to section 5.2.2 for further considerations. Before proceeding to the analysis of each experiment, a few points must be remarked. First of all, the round robin technique avoided situation where users completed the exactly same test contemporaneously. Students that attended the second experiment were unaware of the fact that their clicks were logged and registered nor that presented sets were different. For both tests the task was to "choose the best" (set or combinations), according to personal preferences, without specifying any other objective or attribute to evaluate; no external or internal influences had affected users choices. One criticism that has been moved by some of the second experiment attendants was the lack of a sorting mechanism, as explained in section 5.2.2. Since the aim of the test was directly connected to the perceived quality of the sets, and the sets differ one another mainly on their sorting, we could not provide the users with an ordering mechanism, as it would have invalidated the premises of our work. Another interesting comment was made by users: the majority of them had felt the need to explore the details we provided through the related toolbox (Figure 5.15). This underlines an important aspect: information provided at-a-glance were not sufficient to make a satisfiable decision, even though would have been latter considered as "essential" by the same users.

5.2.1 Experiment 1 - Direct comparison

The goal of the first experiment was to establish if a diversified set would have been recognized and preferred by users when facing a direct comparison. Users were asked to evaluate the quality and variety of the items, selecting the set which was considered more satisfactory between the proposed two. The three possible cases users had faced were: *MMR*, *MaxSum* and un-diversified set. The test was performed by 70 users, among which 25% were students and 75% were either search experts from industry or academia. The following diagram (Figure 5.22) represents the total number of votes received by each of the three situations, divided by scenario.

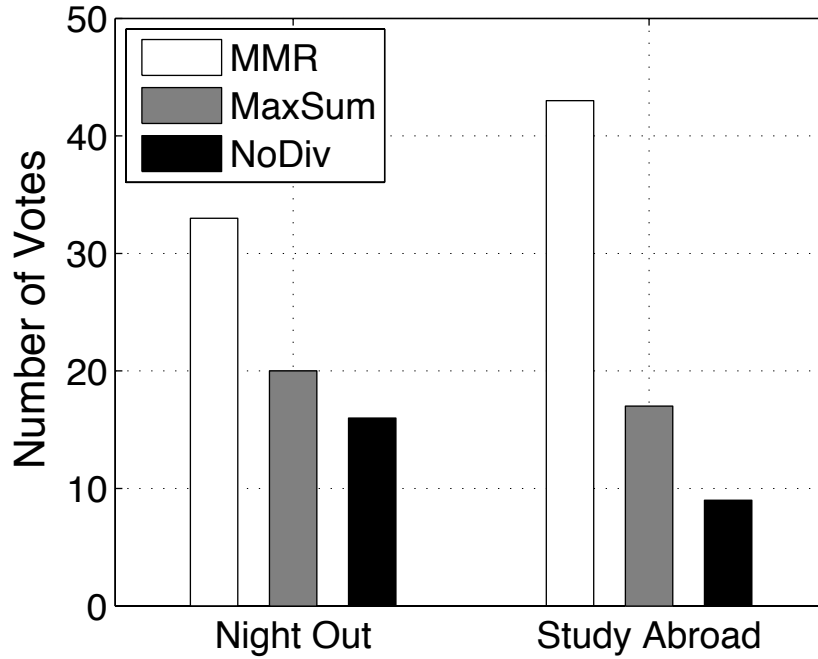


Figure 5.22: User votes in the two scenarios

To provide statistical relevance to these data, we applied a Kruskal-Wallis one-way, non parametric ANOVA test, which confirmed this relevance, obtaining average values differences at significance level $\alpha = 0.01$ for each measure. An immediate, noticeable, result is that *MMR* sets have been much more preferred than the others. More importantly, for both scenarios, *MMR* and *MaxSum* sets have been considered better than the un-diversified one. It is also interesting to point out that *MaxSum* algorithm has led to sets that were

considered just as little better than the initial sets. We adduce as explanation the fact that *MaxSum* works considering only couples of items and tries to maximize distance between them (see chapter 3.4.1), instead of the total "diversity" of the whole. If we consider each of the possible comparisons, as in picture 5.23 and 5.24 we can notice a few differences between the two scenarios.

Considering first the setting based on University set (Study Abroad scenario), we can see (figure 5.23 below) that the collections diversified using *MMR* algorithm have been preferred to both the initial situation and the ones diversified with the *MaxSum* algorithm. A key result to this test is that diversified sets have been preferred to non-diversified ones. The total number of votes received by *MMR* and *MaxSum* sets is at least triple than the original one. When the users have faced the comparison with the un-diversified sets, the votes of *MMR*-based sets almost quintupled the alternative. If we consider how a comparison could have looked like (see figure 5.6), it is obvious that the repetitiveness of the items in the un-diversified set have led users to consider *MMR* groups as better, and consequently vote for them.

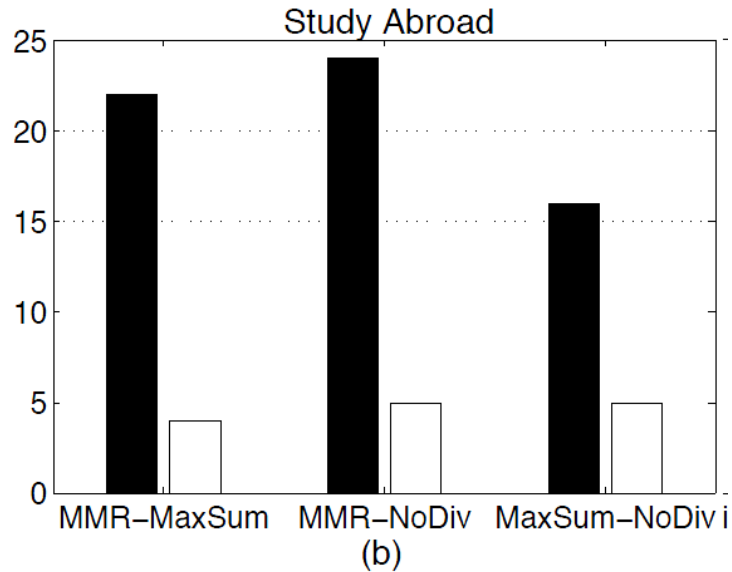


Figure 5.23: User preferences in the Study Abroad scenario

The same ANOVA test was performed, confirming data statistical validity. The net preference of *MMR*-diversified sets over *MaxSum* ones can be explained by two factors. First, *MaxSum* operates at his best when dealing with fragmented sources [12], and since the initial

1000 items were computed randomly, we weren't able to guarantee this property. Second, this algorithm maximizes distance between couple of objects, and can lead to diversified sets that are "similar" in pairs. Nevertheless, *MaxSum* has been voted more than the non-diversified sets, so an improvement in the sets quality after diversification have been perceived by users.

The second scenario has shown some interesting results too. As for the Study Abroad scenario, *MMR*-based sets have proven to be preferred to both the un-diversified and the *MaxSum* based sets. Again, the number of votes received by sets diversified with *MMR* algorithm is at least three time bigger than the alternatives. However, in this scenario the difference of votes between *MaxSum*-based collections and the original ones is really small. Users have not perceived a concrete increments in the quality of the sets when *MaxSum* was applied.

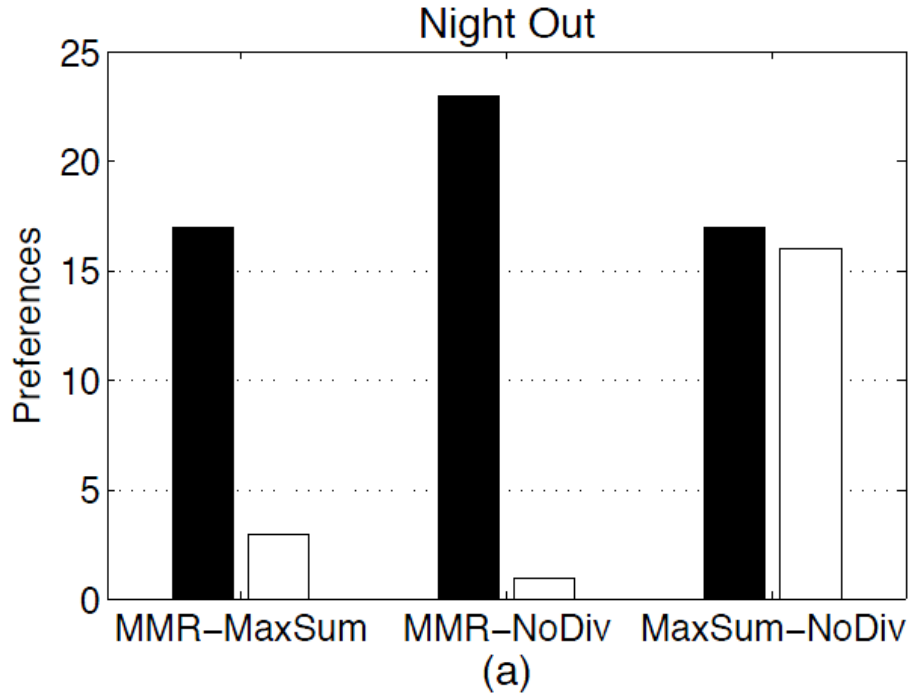


Figure 5.24: User preferences in the Night Out scenario

The discrepancy with the previous scenario (5.23), shown in the third column of picture 5.24, can be attributed to the diversity in the original datasets. Since *MaxSum* algorithm operates calculating pair distances, whenever two identical items (same hotels or restaurants or museums) are spaced by an odd number of other items in

the original set, they have a higher probability to compare in the result set than in any other diversification algorithms. The borderline situation happens when the initial set contains element that this algorithm considers already to be at their maximum distance. The next example shows this limitation. Let assume a dataset composed by 8 items, over three distinct attributes (similar to the "ids" in the two scenarios). The 8 combinations are initially sorted as in figure 5.25. Attributes B and C have distinct values for each terms, while attribute A ranges from 1 to 6. If we assume as distance function between two combinations a binary formulation of the classical identity function, $score = 1$ if both items have the same value for a certain attribute, 0 otherwise. In the proposed dataset the score is based only on the first column, since all terms have different value for the other two attributes (B and C). *MaxSum* will evaluate the first couple of items, considering them as the most possible diverse objects of the whole set (since diverse for all attributes and ranked highest). Therefore the first two items will be placed in the "diversified set". During the second iteration, terms 3 and 4 will be taken into consideration, and since their distance is maximal (all values are different one another), and rank highest, *MaxSum* will set them in the diversified resultset. Similar conclusion will be in the next two steps, where items 5 and 6 will be evaluated and selected as the best possible combinations before the last two objects will be chosen. This example is a classical borderline situation where the initial dataset coincide with the diversified once achieved by *MaxSum* algorithm.

For the same reason, since the initial datasets used for the first test were randomly chosen and could have this kind of discontinuity, *MaxSum* could have misled users to think that no diversification had took place at all, and the quality of the sets they were comparing was actually the same. Nevertheless, we received only 3 feedbacks from users that had to chose between identical sets, over more than 70. This means that users estimated as similar, sets that have just one or two combination replaced from the initial status, validating the hypothesis that to achieve an effective diversification, a significant number of combinations had to be evaluated.

5.2.2 Experiment 2 - Best three combinations

Goal of the second experiment was to prove that diversification is useful during searching tasks over multi-domain data. To assess the perceived usefulness of diversification, we observed the user behavior

	Attr.1		Attr.2		Attr.
1)	A1	-	B1	-	C1
2)	A2	-	B2	-	C2
3)	A2	-	B3	-	C3
4)	A3	-	B4	-	C4
5)	A4	-	B5	-	C5
6)	A5	-	B6	-	C6
7)	A4	-	B7	-	C7
8)	A6	-	B8	-	C8

Figure 5.25: User preferences in the Night Out scenario

in a realistic search activity, and registered it using a logging system. The logged operations were the following: details, next and previous page navigation (within same tasks), selection and de-selection of the combinations and the answers to the various questionnaires. The idea was to prove that the number of clicks and the time spent by users that were exploring diversified dataset were much lower than the users that had to deal with non-diversified sets. The experiment was designed as a controlled between-subject study, where users were required to perform the tasks explained in the design section (5.1.2, under both scenarios. Participants were recruited from the Bachelor and Master student population of the University. During a 1 week period, 66 users from 9 different countries accessed the on-line evaluation tool. From the logs content, we established that participants ranged in age from 20 to 28 years old, and 95% of them were male. All of them were assiduous users of Web search engines. For each experiment we collected the total task completion time and the number of visited result pages. Each participant took an average of 8 minutes to complete the task (including the time spent reading the task explanation and filling the questionnaire).

If the result sets of the three experiments are equally good (taking into account relevance and variety), we would expect users to take a similar amount of time to complete each task, and to visit a comparable number of pages to find the "best three" combinations.

Figure 5.26 shows the average and standard deviation of the comple-

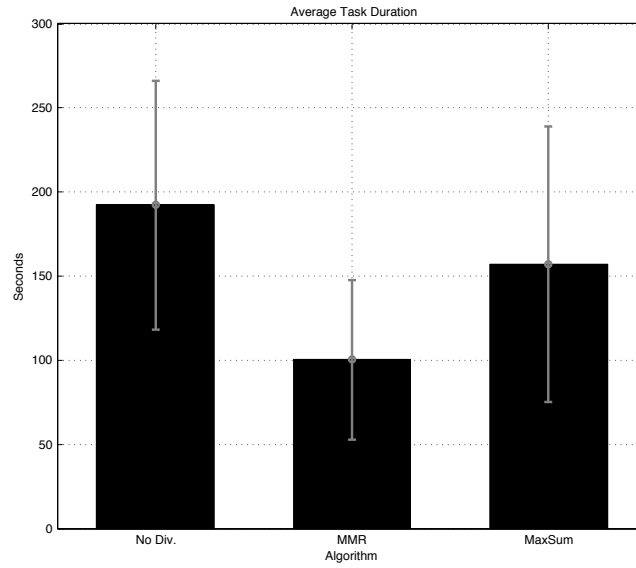


Figure 5.26: Duration of the test for the Study Abroad scenario

tion time for a task of the Study Abroad scenario. It is reasonable to assume that the time difference was due to the impossibility of finding the satisfactory combinations when dealing with un-diversified sets in few pages. Users were in this case forced to explore deeper to get the three best items, taking more time and wasting clicks in the searching process. On the contrary, users facings *MMR* diversified sets reached a satisfying level quicker, since the variety of their sets were perceived as higher. If we consider the number of clicks, instead of the duration, a similar result is achieved (shown in fig. 5.26)

There is a significant reduction in terms of both clicks and duration for users whose set was not the original (un-diversified) one. Once again we attribute this difference to the commitment of users to investigate until they found the best three combinations. For those who were performing this task on the initial dataset, it required both more details exploration, and also to go back and forth many times throughout the whole set, increasing the click counters. *MMR* has been perceived as more effective by users. The clicks number was less than the half if compared to *MaxSum*, and almost a quarter if compared to the non-diversified set.

Similar results have been obtained for the "Night Out" scenario.

A few inconsistencies have been discover from the log analysis. The number of pages visited by users, which we had considered as di-

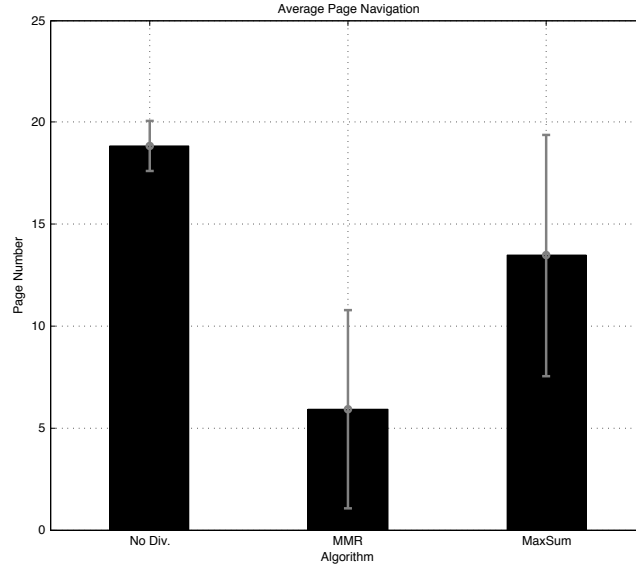


Figure 5.27: Number of clicks for the Study Abroad scenario

rectly connected to the algorithm in a similar way as the total number of clicks, proved to be more complex then expected. The visited pages measures behaved differently in the two scenarios, denying our hypothesis. We were in fact not able to demonstrate, by the test results, that providing a diversified set would have meant for the user a minor number of pages to visit. The total number of clicks and the total number of visited pages, even though are for sure correlated, acts differently with respect to diversification.

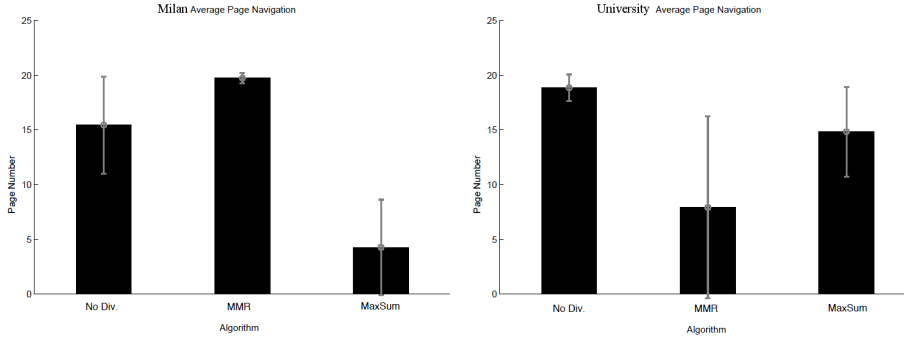


Figure 5.28: Visited pages comparison between scenarios

Kruskal-Wallis ANOVA test provided once again statistical support to our considerations. We justify the situation shown in picture 5.28 with the differences in the sets and the users aptitudes. It is in fact reasonable to think that all users, when facing a repetitive set

(composed by almost identical items) have decided to explore further. Some of them, misunderstanding the hypothesis of not-ordered set, have considered the results pages as in a traditional IR system, where the quality of the results decreases with the ranking. This theory justifies the low counts for users with un-diversified sets and *MaxSum* sets, providing a reasonable explanation on the behaviour showed in the above figure.

However, the global click counts and the global duration of the tasks proved our main hypothesis, validating the results of the quantitative test (Chapter 4).

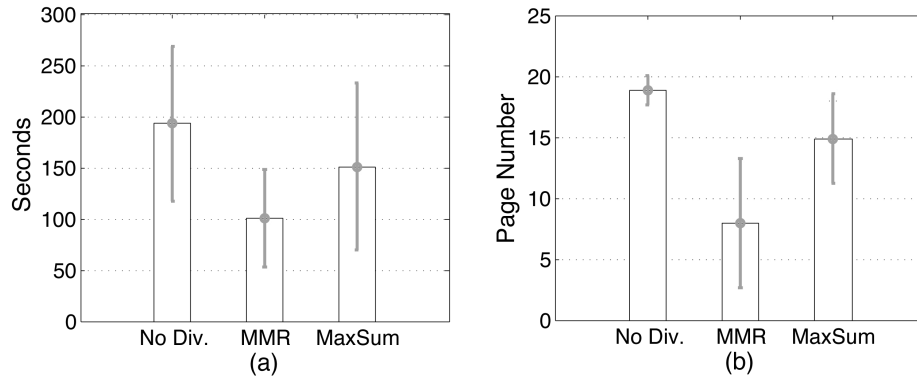


Figure 5.29: Duration and clicks for the user test compared

We observe that *MMR* and *MaxSum* lead, on average, to 45% and 13% reduction in terms of task completion time, and 60% and 16% reduction in terms of number of explored pages, with respect to the baseline.

5.2.3 Comments

Analyzing the results, we can state that users perceived a substantial increment in the overall quality of the sets when diversification took place. From the comments received, it is clear that a sorting mechanism is appreciated, if not required, by users, willing to reorder combinations in the decision making process. The user interface plays an essential role in diversification since it is the one directly responsible for the understanding of the users. A simple layout is effective when the need is to convey basic diversification effects, while a more complex and well-studied one is required to transmit thinnesses and quibblings. Despite that, most of the time

MMR and *MaxSum* have been recognized as better from users since they have effectively reduced time and clicks needed to complete the tasks. Such results are consistent with the ones obtained in the quantitative evaluation (Chapter 4).

Chapter 6

Conclusion and Future Work

Multi-domain search paradigm is quickly becoming more and more important. Modern systems try to create mash-ups of multiple Web data sources to answer complex search purposes. Despite this trend, current search engines are not able to support users with complex information seeking tasks.

The multi-domain query approach proposed in the Search Computing project provides methods and techniques to tackle the problem, but, so far, it has as a major drawback the production of result sets that contain several result repetitions, which may lead to unsatisfactory user experience.

Current research is extensively addressing the problem of increasing data variety, in order to improve the quality of returned query result sets. This thesis investigates in this direction, taking under exam the problem of retrieving relevant and, at the same time, diverse combinations, in the multi-domain context. We analyzed the characteristics and performances of four greedy diversification algorithms (*MMR*, *MaxMin*, *MaxSum*, *MaxCov*), in order to test their suitability in the SeCo context. In addition, two kinds of distance have been implemented and tested: categorial distance, checking object equality, and quantitative distance, comparing objects feature within metric spaces.

We executed a set of quantitative tests, using adaptations of the evaluation metrics currently adopted in the context of diversification for Web documents. Results showed that considered algorithms can actually improve the result set quality. *MMR*, *MaxMin* and *MaxCov*

on average have proved to increase the result set variety, while *Max-Sum* gave controversial results, sometime worse than the baseline. Categorical distance seemed to produce better results than quantitative one, but we could not identify an algorithm able to outperform the others in each test and each data structure. We also identified a clear bias on algorithm performance due to result set composition, a topic that will be further investigated in future works.

To test how users perceive the adoption of diversification algorithms in their information seeking experience, we designed and conducted two user studies, which involved about 150 real-world search engine users. In the first one, expert users were asked to chose between two sets, diversified according to different algorithms and compared to un-diversified sets. The majority of the users recognized as "better" sets that were modified by diversification algorithms.

In a direct comparison between the initial set and a diversified one, the latter received at least three times the votes of the former. In some experiments we found that *MMR* diversified sets obtained up to six times the user preferences. These results prove not only that diversification increases the quality of the set, but also that it is perceived as an added value by users.

In the second test a realistic search experience was simulated, asking more than 70 non-expert users to select the best three combinations over two hundreds, within two scenarios and more than thirty different sets. Qualitative test results supported the quantitative evaluation hypothesis. Even though results were overall positive, attendants of the experiments weren't always able to perceive a gain in the quality of result sets after diversification. We blame the graphic interface for that, as it probably hindered the ability of users to find visual feedbacks on how to evaluate the improvement on the set quality.

6.1 Future Work

Overall, these results provide evidences that the usage of diversification techniques in the multi-domain context is worthwhile and effective, and they suggest several directions for future works.

In the thesis we recognized as an important characteristic the relations cardinality and the variance of numeric attributes when dealing with quantitative diversity. Based on such considerations, it would be helpful to study a method that can show a-priori when

diversification can provide significant improvements in the result set quality, thus saving time and computational resources. We also plan to perform additional experiments to study and classify the effect of different ranking and diversification attributes to the inherent diversity of the un-diversified dataset.

Finally, a possible direction for future work would be the integration of diversification with user preference systems, in order to be able to retrieve combinations that are not only relevant and diverse, but also tailored to specific users or user groups.

Bibliography

- [1] B. L. 0002 and H. V. Jagadish. Using trees to depict a forest. *PVLDB*, 2(1):133–144, 2009.
- [2] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.
- [3] F. Bonchi, C. Castillo, D. Donato, and A. Gionis. Topical query decomposition. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 52–60. ACM, 2008.
- [4] A. Bozzon, M. Brambilla, S. Ceri, and P. Fraternali. Liquid query: multi-domain exploratory search on the web. In *WWW*, pages 161–170, 2010.
- [5] A. Campi, S. Ceri, A. Maesani, and S. Ronchi. Designing service marts for engineering search computing applications. In *ICWE*, pages 50–65, 2010.
- [6] J. G. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.
- [7] S. Ceri, D. Braga, F. Corcoglioniti, M. Grossniklaus, and S. Vadacca. Search Computing Challenges and Directions. *Objects and Databases*, pages 1–5, 2010.
- [8] C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666, 2008.
- [9] E. Demidova, P. Fankhauser, X. Zhou, and W. Nejdl. *DivQ*: diversification for keyword search over structured databases. In *SIGIR*, pages 331–338, 2010.

- [10] M. Drosou and E. Pitoura. Search result diversification, 2010.
- [11] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 6–12, 2007.
- [12] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.
- [13] T. Griffith and J. Pfeifer. Applying Diversity Metrics to Improve the Selection of Web Search Term Refinements.
- [14] J. Haritsa. The KNDN problem: A quest for unity in diversity. *IEEE Data Eng. Bull*, 32(4):15–22, 2009.
- [15] J. Hu, G. Wang, F. Lochovsky, J. Sun, and Z. Chen. Understanding user’s query intent with wikipedia. In *Proceedings of the 18th international conference on World wide web*, pages 471–480. ACM, 2009.
- [16] I. Ilyas, G. Beskales, and M. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):1–58, 2008.
- [17] I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. In *VLDB*, pages 754–765, 2003.
- [18] P. Lakkaraju, S. Gauch, and M. Speretta. Document similarity based on concept tree distance. In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 127–132. ACM, 2008.
- [19] J. Lin, N. Madnani, and B. Dorr. Putting the user in the loop: interactive Maximal Marginal Relevance for query-focused summarization. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 305–308. Association for Computational Linguistics, 2010.
- [20] Z. Liu, P. Sun, and Y. Chen. Structured search result differentiation. *PVLDB*, 2(1):313–324, 2009.
- [21] C. Lofi, U. Guntzer, and W. Balke. Efficient computation of trade-off skylines. In *Proceedings of the 13th international conference on extending database technology*, pages 597–608. ACM, 2010.

- [22] Y. Luo. *Multi-Dimensional Skyline Query Processing*. PhD thesis, Citeseer, 2004.
- [23] D. Martinenghi and M. Tagliasacchi. Proximity rank join.
- [24] E. Minack, G. Demartini, and W. Nejdl. Current Approaches to Search Result Diversification. In *Proc. of 1st Intl. Workshop on Living Web*, 2009.
- [25] P. Pantel and D. Lin. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM, 2002.
- [26] Y. S. T. Y. Q. Feng, L. Liu and Y. Dai. Find your favorite star with maps: a personalized multi-attribute ranking algorithm. In *Technical Report*. TR, 2010.
- [27] F. Radlinski and S. T. Dumais. Improving personalized web search using result diversification. In *SIGIR*, pages 691–692, 2006.
- [28] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In *Proceedings of the 19th international conference on World wide web*, pages 781–790. ACM, 2010.
- [29] A. Rajaraman. Kosmix: high-performance topic exploration using the deep web. *Proceedings of the VLDB Endowment*, 2(2):1524–1529, 2009.
- [30] R. Santos, C. Macdonald, and I. Ounis. Exploiting query reformulations for Web search result diversification. In *Proceedings of the 19th international conference on World wide web*, pages 881–890. ACM, 2010.
- [31] D. Skoutas, M. Alrifai, and W. Nejdl. Re-ranking Web Service Search Results Under Diverse User Preferences. 2010.
- [32] M. Soliman, I. Ilyas, and M. Saleeb. Building Ranked Mashups of Unstructured Sources with Uncertain Information. *Proceedings of the VLDB Endowment*, 3(1), 2010.
- [33] R. Song, Z. Luo, J. Wen, Y. Yu, and H. Hon. Identifying ambiguous queries in web search. In *Proceedings of the 16th international conference on World Wide Web*, pages 1169–1170. ACM, 2007.
- [34] K. Stefanidis, M. Drosou, and E. Pitoura. PerK: personalized keyword search in relational databases through preferences. In

- Proceedings of the 13th International Conference on Extending Database Technology*, pages 585–596. ACM, 2010.
- [35] R. van Leuken, L. Garcia, X. Olivares, and R. van Zwol. Visual diversification of image search results. In *Proceedings of the 18th international conference on World wide web*, pages 341–350. ACM, 2009.
 - [36] R. Van Zwol, V. Murdock, L. Garcia Pueyo, and G. Ramirez. Diversifying image search with user generated content. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 67–74. ACM, 2008.
 - [37] V. Vapnik. Principles of risk minimization for learning theory. *NIPS1991*, pages 831–838, 1991.
 - [38] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Yahia. Efficient computation of diverse query results. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 228–236. IEEE, 2008.
 - [39] M. Wand. A simple algorithm and proof for type inference. *Fundamenta Informaticae*, 10:115–122, 1987.
 - [40] J. Wang and J. Zhu. Portfolio theory of information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 115–122. ACM, 2009.
 - [41] L. Ying, M. Jun, and S. Yuyin. Applying Dewey Encoding to Construct XML Index for Path and Keyword Query. In *Database Technology and Applications, 2009 First International Workshop on*, pages 553–556. IEEE, 2009.
 - [42] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 368–378. ACM, 2009.
 - [43] C. Zhai, W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10–17. ACM, 2003.
 - [44] C. Ziegler, S. McNee, J. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceed-*

ings of the 14th international conference on World Wide Web,
pages 22–32. ACM, 2005.

Appendix A

Quantitative test charts

This appendix contains all charts made for quantitative evaluation that couldn't find space in chapter 4. For each *lambda* ranging from lambda 0 to 1 with a 0.25 steps, α -DCG is analyzed with α values in the same range as λ . Since the metric returns identical results for all approaches when $\alpha = 0$, this case was ignored. MD-Recall for all different λ values is plotted as well.

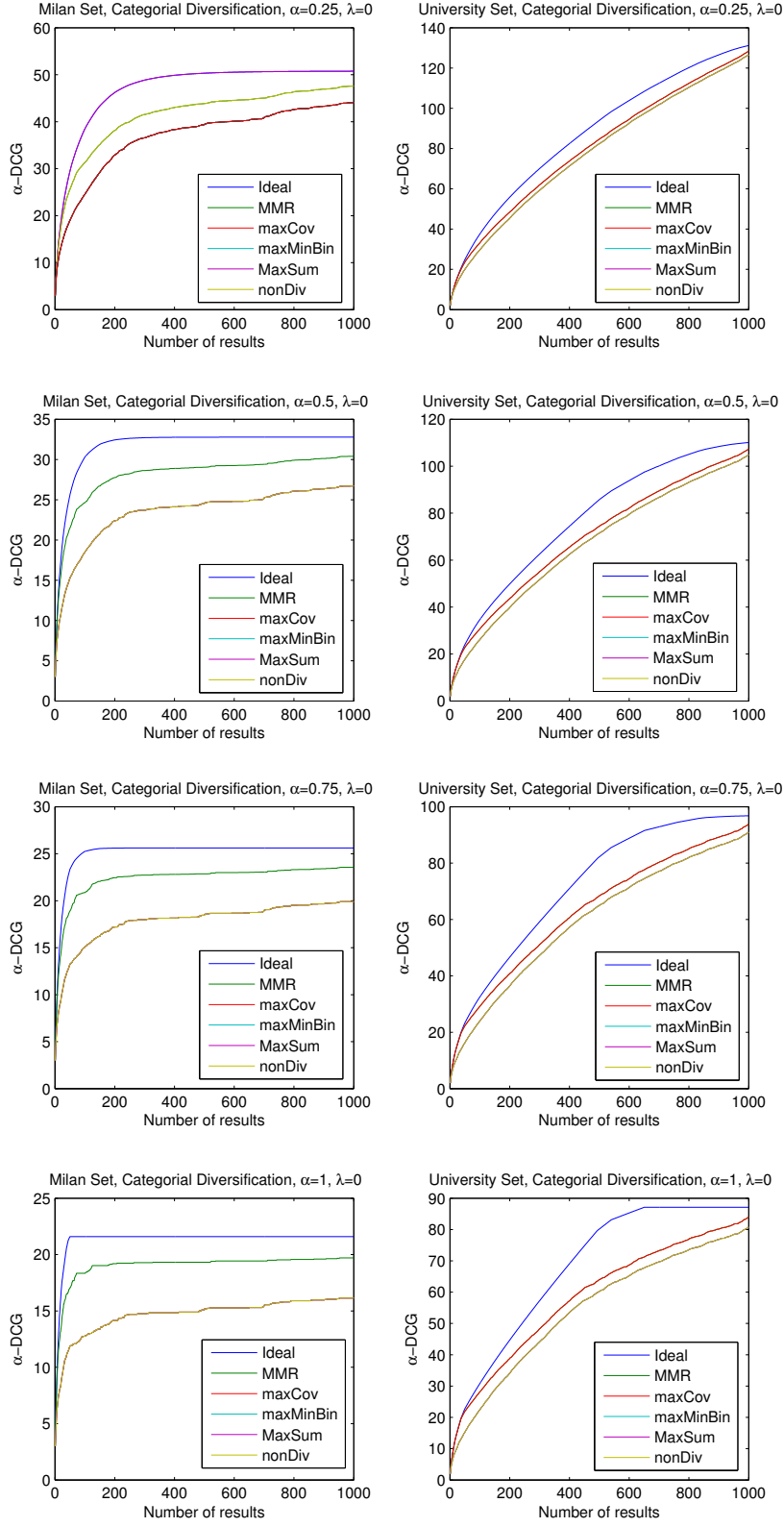


Figure A.1: α -DCG for $\lambda=0$, categorical diversity

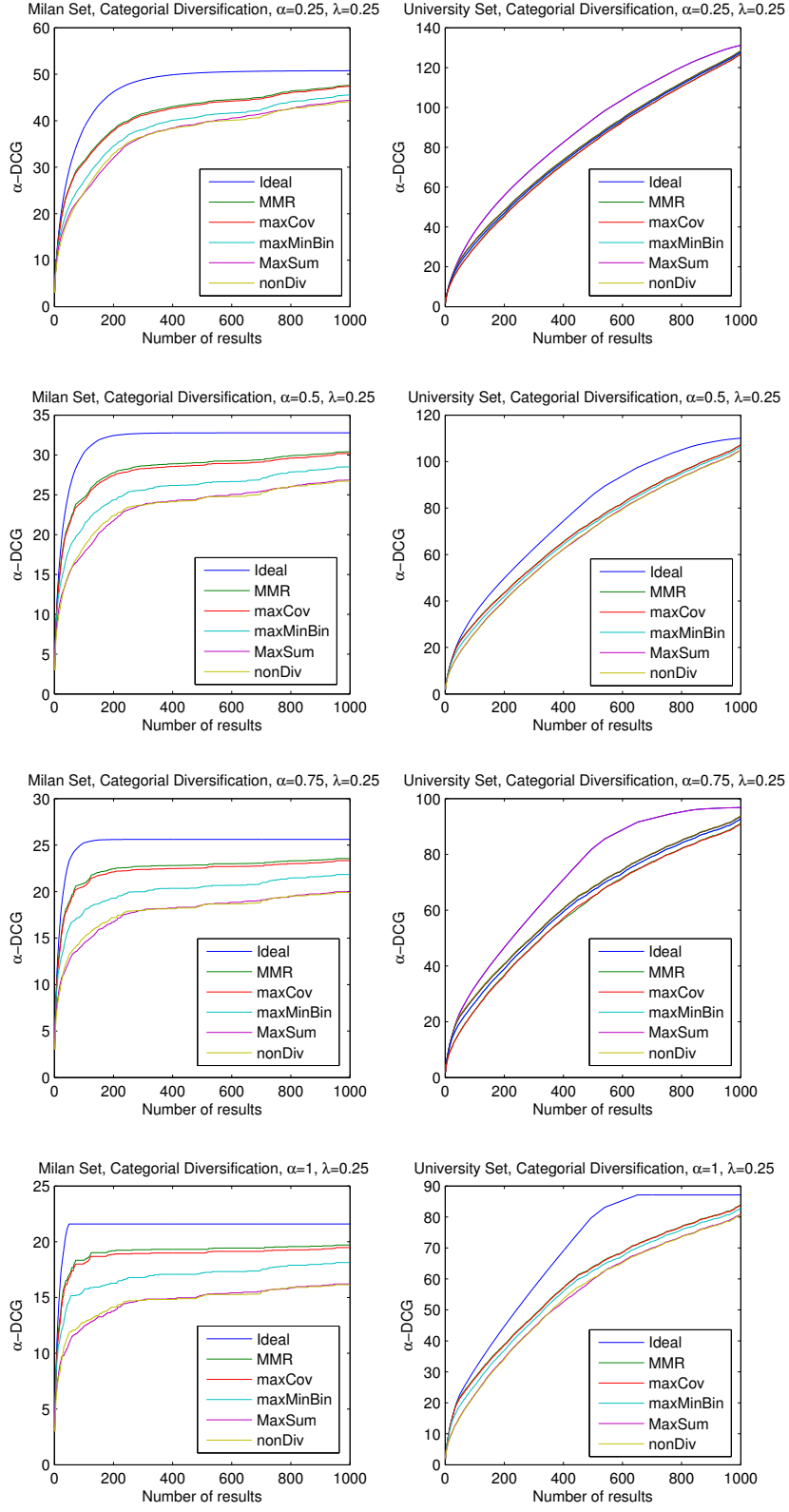


Figure A.2: α -DCG for $\lambda=0.25$, categorical diversity

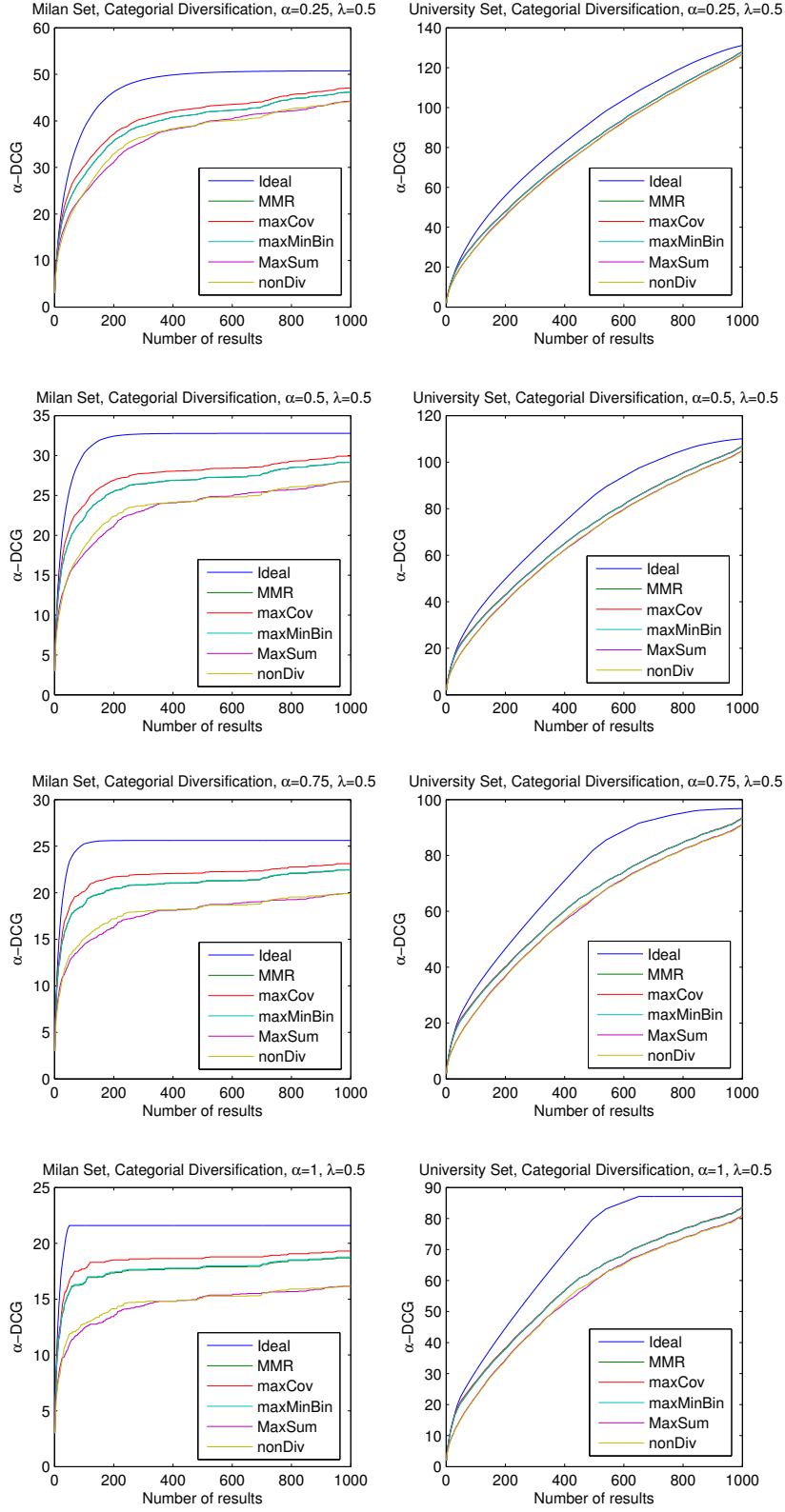


Figure A.3: α -DCG for $\lambda=0.5$, categorical diversity

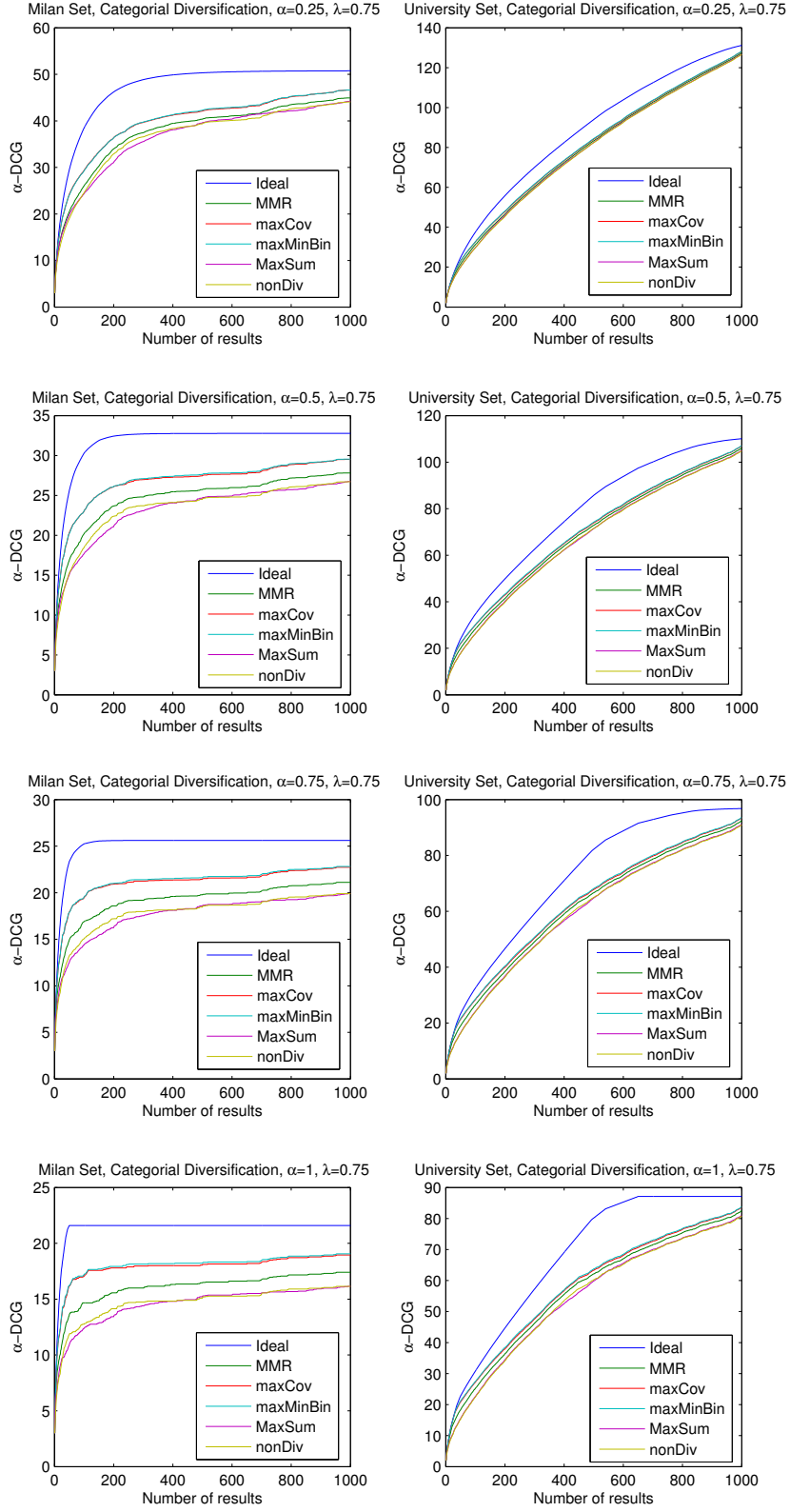


Figure A.4: α -DCG for $\lambda=0.75$, categorical diversity

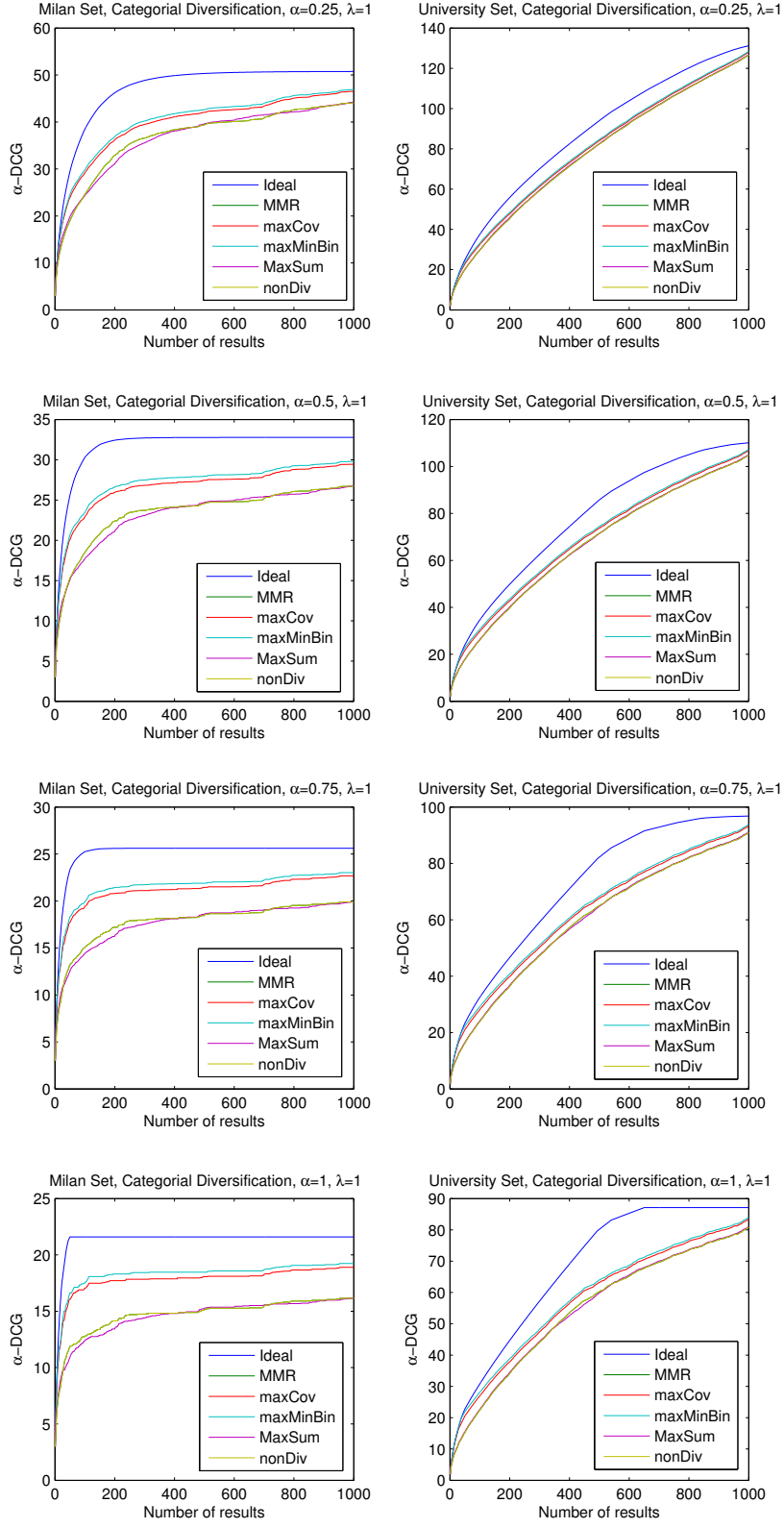


Figure A.5: α -DCG for $\lambda=1$, categorical diversity

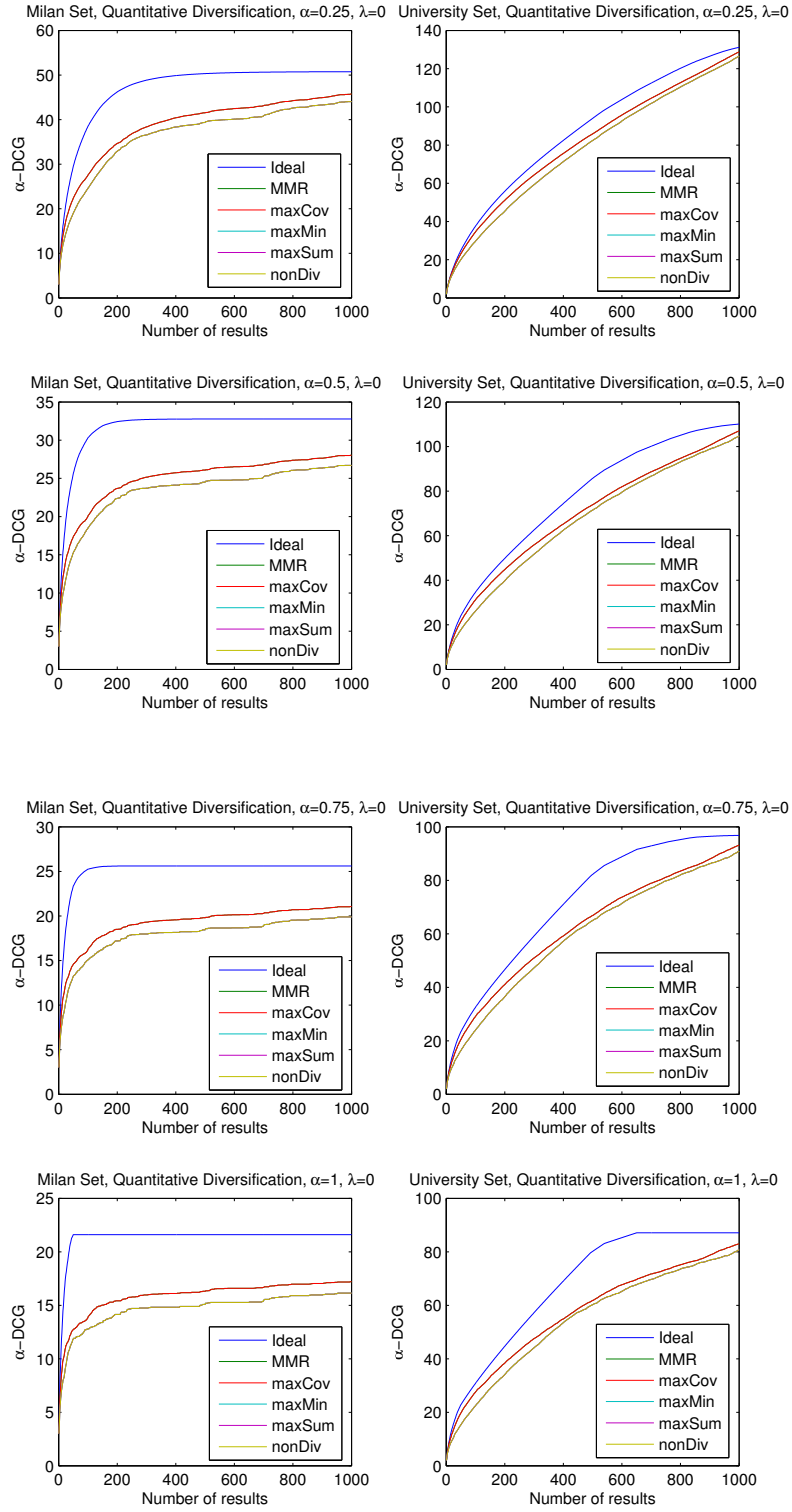


Figure A.6: α -DCG for $\lambda=0$, quantitative diversity

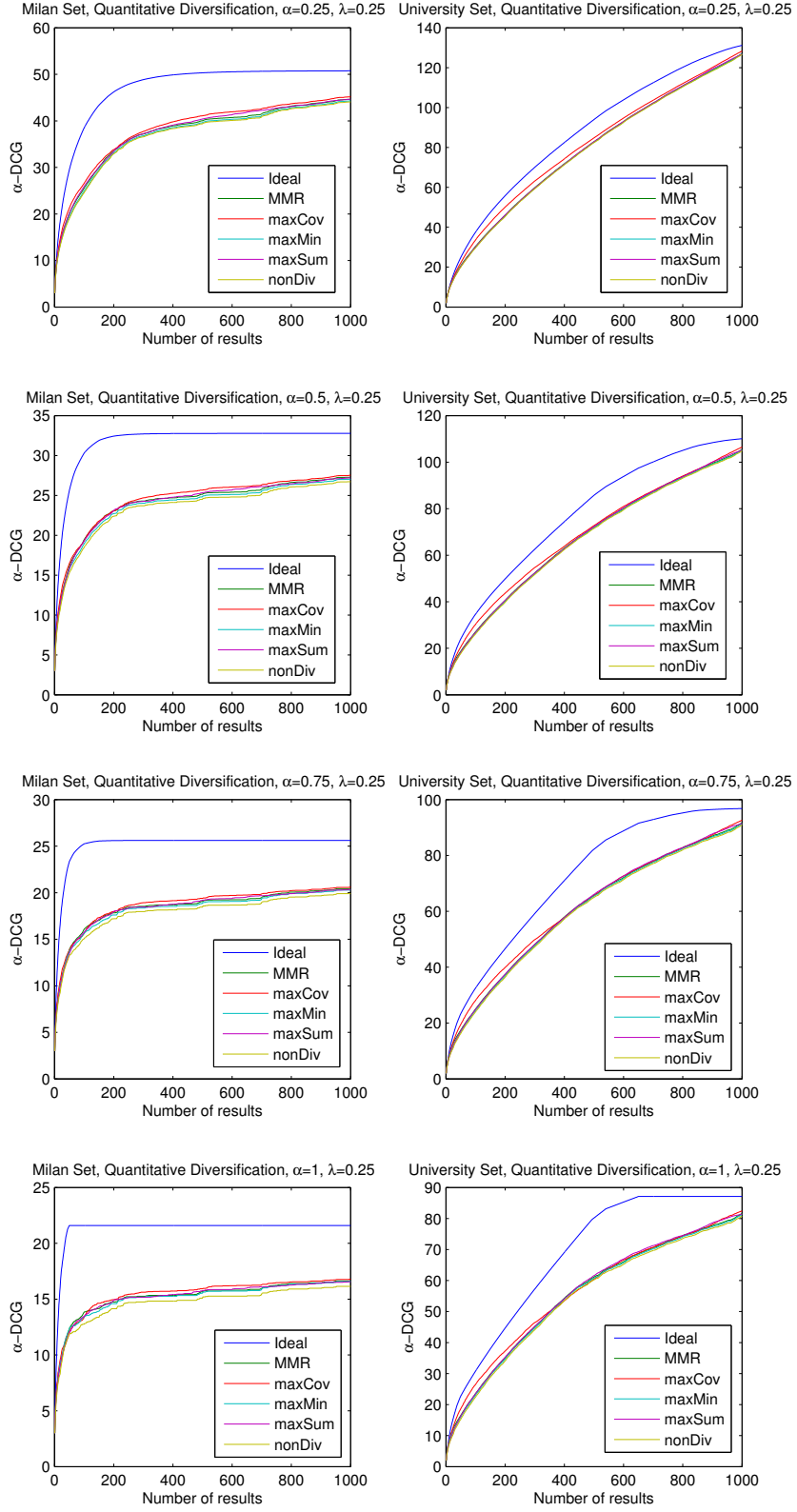


Figure A.7: α -DCG for $\lambda=0.25$, quantitative diversity

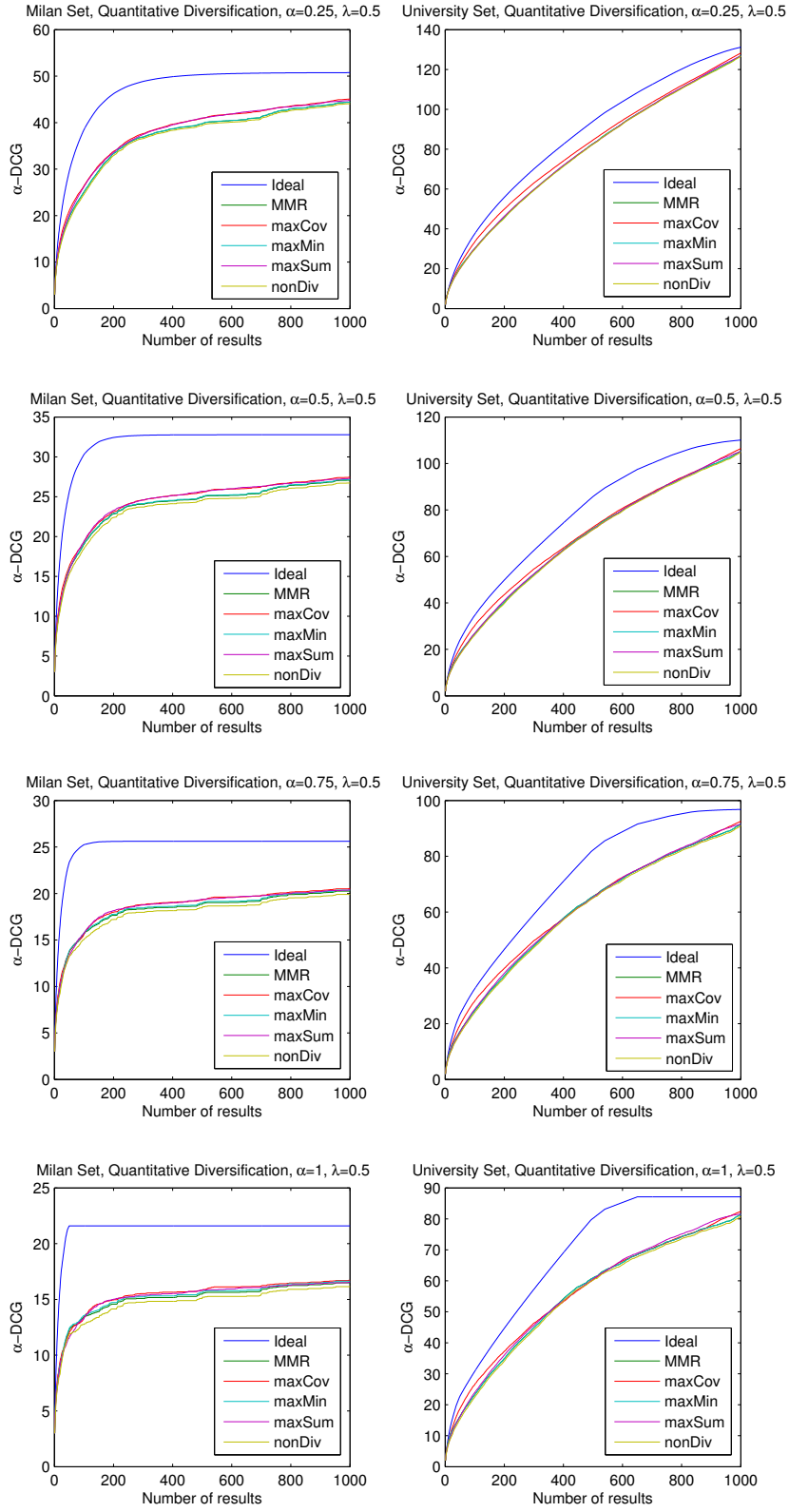


Figure A.8: α -DCG for $\lambda=0.5$, quantitative diversity

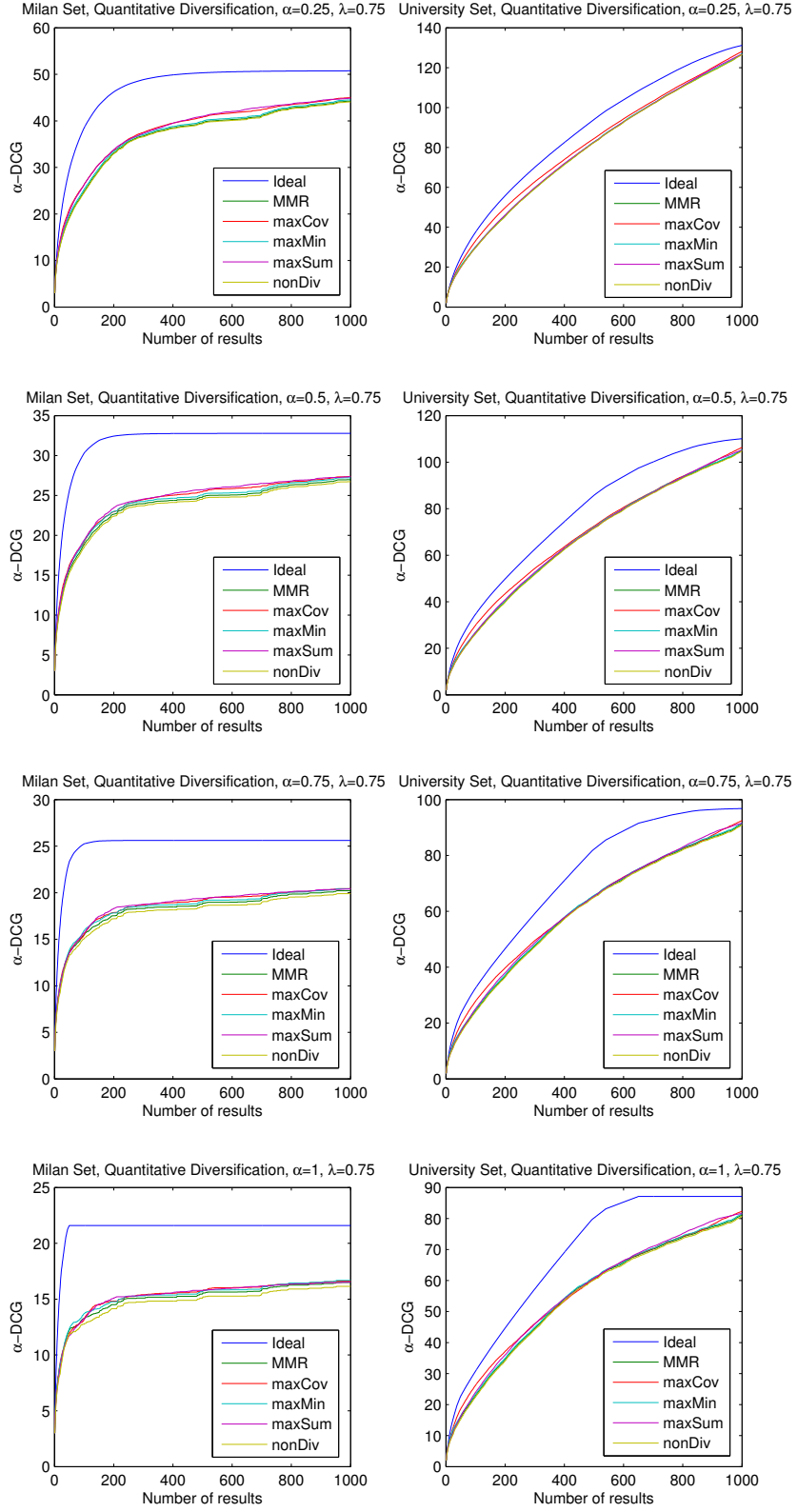


Figure A.9: α -DCG for $\lambda=0.75$, quantitative diversity

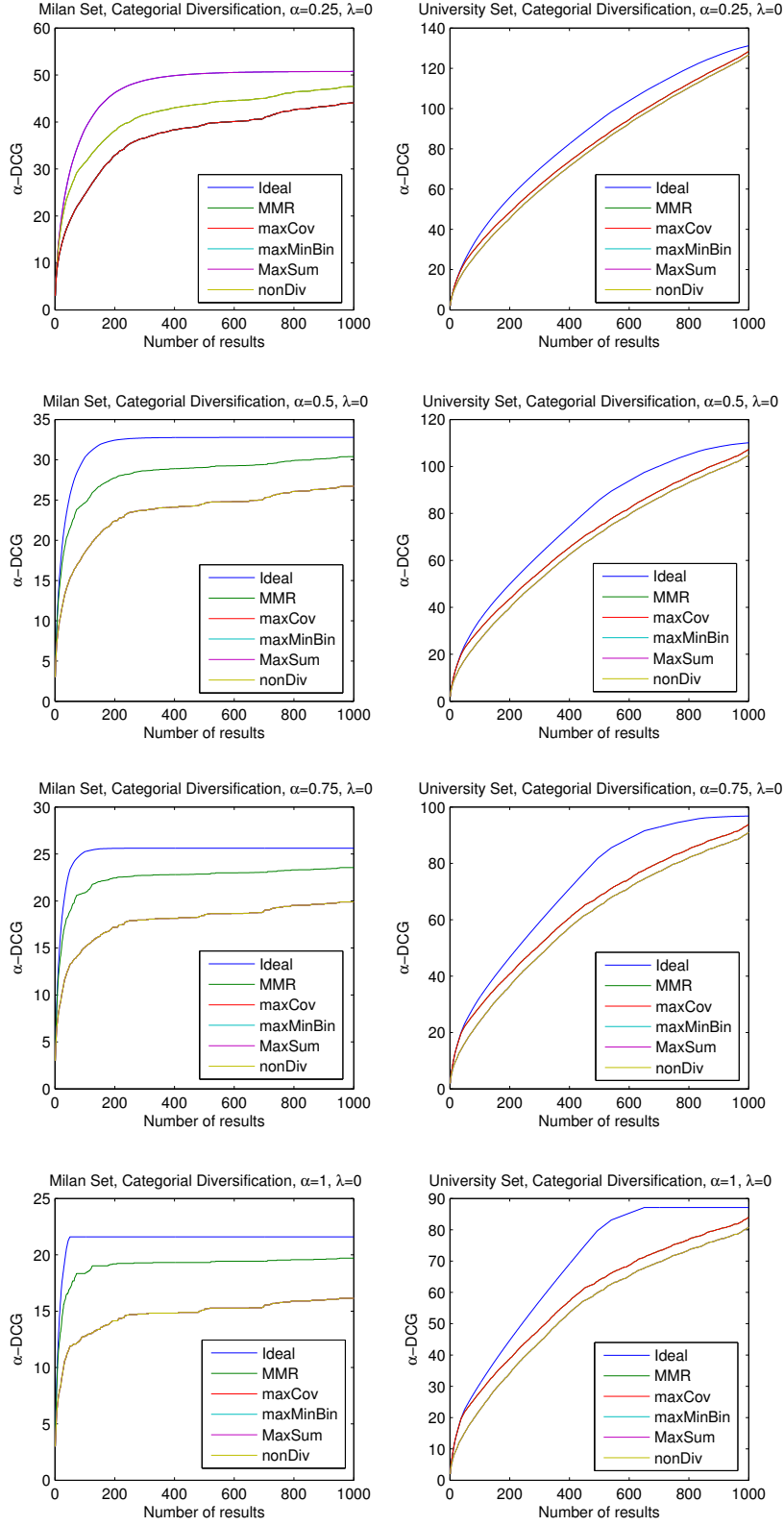


Figure A.10: α -DCG for $\lambda=0$, quantitative diversity

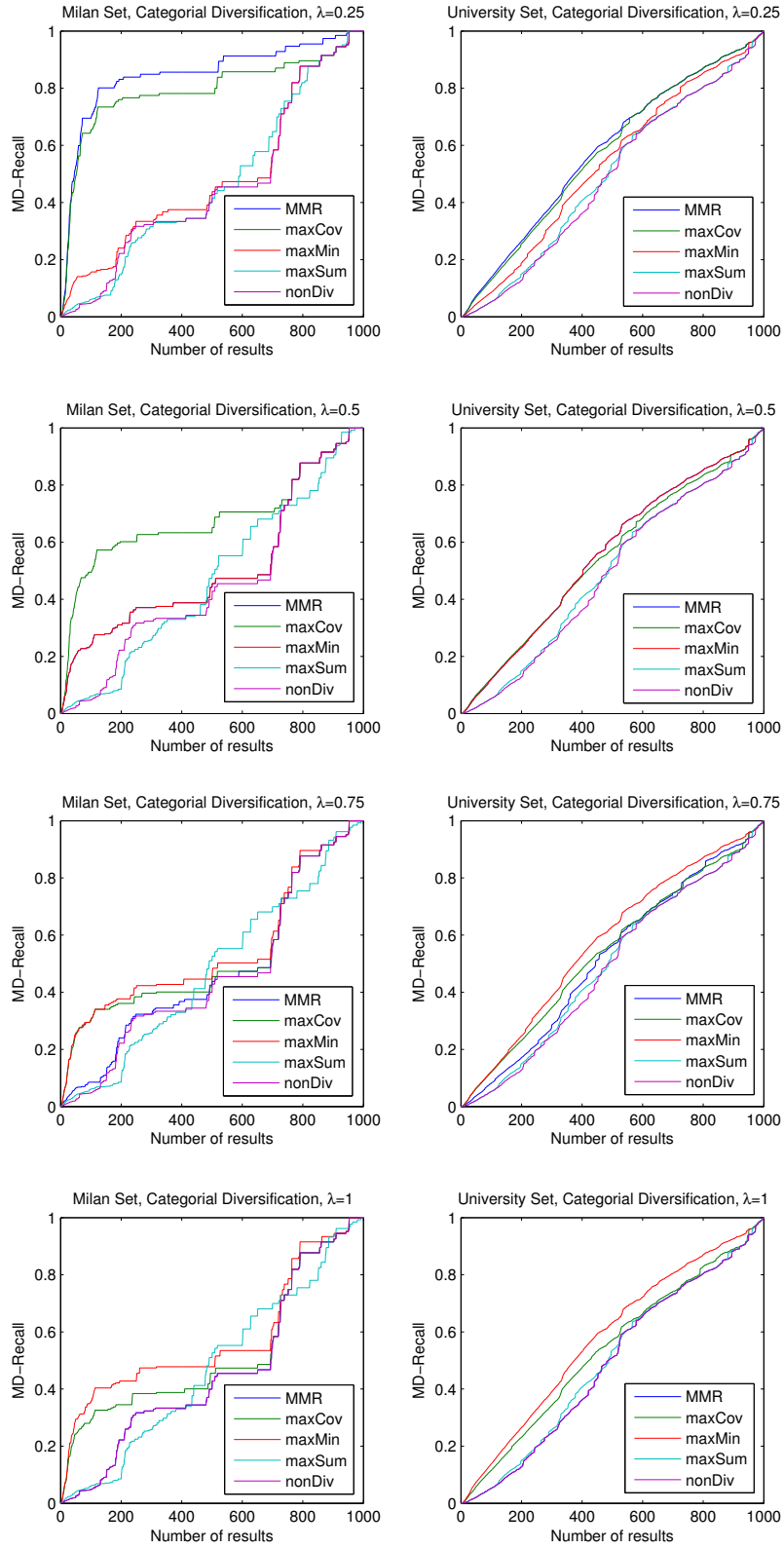


Figure A.11: MD-Recall, categorical diversity

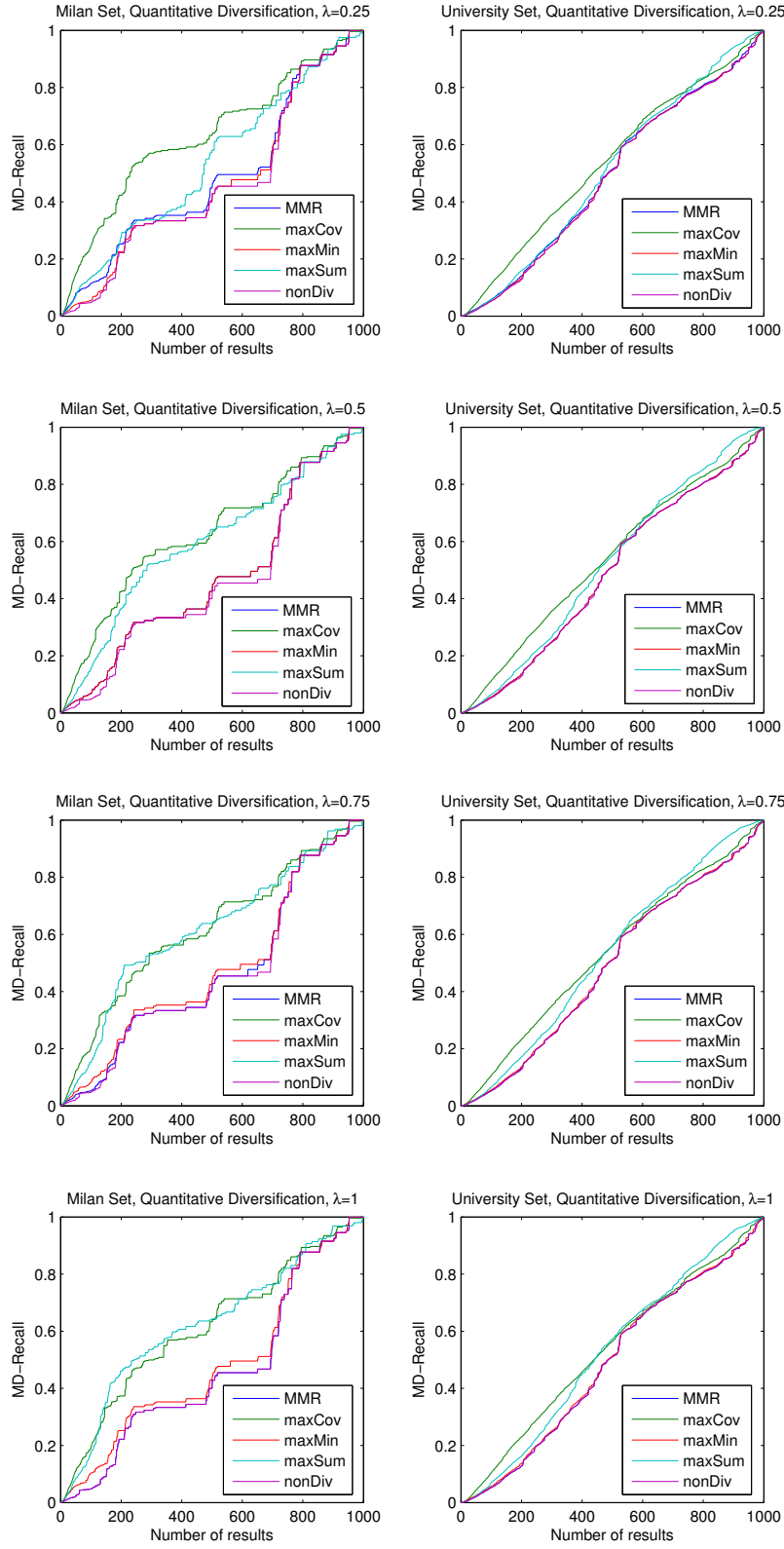


Figure A.12: MD-Recall, quantitative diversity