Future Internet **2015**, *7*, 110-139; doi:10.3390/fi7020110

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

Dynamis: Effective Context-Aware Web Service Selection Using Dynamic Attributes

Atousa Pahlevan *, Jean-Luc Duprat, Alex Thomo and Hausi Müller

Department of Computer Science, University of Victoria, 3800 Finnerty Road, Victoria, BC V8P 5C2, Canada; E-Mails: jld@acm.org (J.-L.D.); thomo@cs.uvic.ca (A.T.); hausi@cs.uvic.ca (H.M.)

* Author to whom correspondence should be addressed; E-Mail: apahlevan@ieee.org; Tel.: +1-250-472-5719.

Academic Editor: Andrew Hudson-Smith

Received: 29 January 2015 / Accepted: 29 April 2015 / Published: 12 May 2015

Abstract: Quality web service discovery requires narrowing the search space from an overwhelming set of services down to the most relevant ones, while matching the consumer's request. Today, the ranking of services only considers static attributes or snapshots of current attribute values, resulting in low-quality search results. To satisfy the user's need for timely, well-chosen web services, we ought to consider quality of service attributes. The problem is that dynamic attributes can be difficult to measure, frequently fluctuate, are context-sensitive and depend on environmental factors, such as network availability at query time. In this paper, we propose the Dynamis algorithm to address these challenges effectively. Dynamis is based on well-established database techniques, such as skyline and aggregation. We illustrate our approach using observatory telescope web services and experimentally evaluate it using stock market data. In our evaluation, we show significant improvement in service selection over existing techniques.

Keywords: service discovery; service selection; quality of service (QoS); Dynamis; aggregation; skyline; histogram; top *k*

---

1. Introduction

With the increasing proliferation of web services, selecting the best one for a consumer's needs can be a challenging and often overwhelming task. Service consumers naturally expect automated and

context-aware service discovery and selection techniques. Current mechanisms based on information retrieval (IR) techniques, particularly string similarity, are of limited use because the service descriptions on which the similarity is calculated are often short and ambiguous.

A promising alternative is to evaluate the suitability of each service using quality of service (QoS) attributes. This is challenging in practice, because QoS attributes can be difficult to measure, frequently fluctuate, are context-sensitive and depend on environmental factors, such as network availability.

We use observatory telescopes as services to illustrate our context-sensitive service discovery techniques. In this domain, geographical position, time of day and weather conditions affect the quality of service selection; these elements contribute to the context in which a picture of the celestial sphere is taken. For instance, one of the attributes specific to this domain is the visibility of the sky, a weather property; another is the relative position of a specific star with respect to the telescope and the darkness of the sky, properties of both geographical position and time. In contrast, the range of brightness in an image is another important QoS attribute for telescopes; the significance of the contrast attribute varies from user to user. In addition, users may request a picture with a specific composition: the number of objects captured in a photograph, and whether the image is to be in color or black and white.

On certain days, particularly those when the telescope is offline due to maintenance, it may be impossible to determine a value for some attributes. In other words, there will be days where photographs are unavailable or the quality of the photographs is too low to recognize celestial properties.

This telescope example illustrates why smart evaluation based on QoS attributes is useful and necessary to improve service selection. One approach is to ignore the fact that attribute values fluctuate and merely assume that all functionally-equivalent services are equally valuable. The user then has to manually sift through the available services, even in cases when one service outperforms or matches another on every measurement. This frustrates users and lowers the adoption rate of a service discovery system.

A real-world case study for dynamic service discovery is the selection of personalized healthcare services: how to identify a relevant and qualitative set of healthcare services among the available ones. The services must be chosen according to context information gathered from bio-sensors and monitoring devices, in spite of large data volumes and transmission losses. Yet, the right information needs to be delivered to the right person, at the right time, in the right place, in the right manner. Service selection has to proceed in spite of missing or incomplete information [1,2].

The best techniques used in practice come in two flavours—those that do not consider QoS attributes [3–6] and those that take a snapshot of the values into account [7–9]. In the former case, the real potential of QoS attributes is squandered. In the latter case, services are misrepresented by temporary fluctuations of QoS values or harshly penalized when the mechanism for obtaining those values fails. Both cases will inevitably arise in practice.

In this paper, we investigate two approaches originating in the database community to deliver a well-chosen subset of services, each with its own advantages: context-sensitive ranking by aggregation [10]; and context-agnostic selection by skyline [11,12]. While both approaches are applicable to our services setting, they require reliable, non-null values for each QoS attribute and, therefore, break down in the presence of dynamic attributes.

New ideas are critical to satisfy the consumer's need for timely, well-chosen web services. We present a novel method based on histogramming for generating reliable values for QoS attributes that can withstand

temporary fluctuations (regardless of their magnitude), as well as null values. Our approach extends the aggregation and skyline techniques to deal with the dynamic attributes effectively. We present the design, implementation, application and extensive evaluation of our QoS-based service selection techniques.

Section 2 provides background information on skyline and aggregation. Section 3 details the problem of web service selection using dynamic attributes. Section 4 discusses related work on web service discovery and, in particular, database research on aggregation query and skyline techniques. Section 5 presents Dynamis, our context-aware web service selection algorithms based on dynamic attributes. Section 6 evaluates our results in the context of a specific example that is easily quantifiable: the stock market. Finally, Section 8 concludes the paper.

2. Background

There are two primary approaches to object selection in databases: aggregation [13], aiming to identify the objects most appropriate to a given context, and skyline [11], aiming to identify in a context-agnostic manner the best objects. Objects can be hotels, movies, restaurants, universities, *etc*. They are described by attributes and typically represented by tuples in a database table. Services are also described by (QoS) attributes and can therefore be represented by tuples, and the aggregation and skyline paradigms can be applied to the service selection problem. However, the aggregation and skyline approaches encounter practical difficulties in the presence of services' dynamic attributes. The remainder of this section describes both approaches (using a services context) and the root causes of those difficulties.

2.1. Aggregation

An aggregation query (*i.e.*, top k or ranked query) ranks each tuple of a database table by combining the attributes with a single, aggregate scoring function and then reports the highest scored tuples (*i.e.*, either the k highest ranked tuples or the ones whose score exceeds a predetermined threshold) [10,13].

Let us assume that services are described by m numeric (QoS) attributes with names $A_1, \ldots, A_i, \ldots, A_m$. Consider a service s, represented by a tuple $\langle a_1, \ldots, a_i, \ldots, a_m \rangle$ of attribute values. The rank of s is computed as follows. First, the user context is modelled by weights for each QoS attribute, $\langle w_1, \ldots, w_i, \ldots, w_m \rangle$, where the magnitude of each w_i reflects the importance of the i-th attribute within the context. Then, the context-aware score for a service is the sum $\sum_{i=1}^{m} a_i \cdot w_i$. The rank of service s is the number of services with a higher score. This affords user customization, since context is built into the ranking mechanism.

2.2. Skyline

A disadvantage of the aggregation query approach is that the reduction of m attributes to a single score often reduces the accuracy of the retrieval process in multi-criteria decision-making [14].

The skyline of such a table is context-agnostic, in contrast to the top k, which proffers the advantage of circumventing the non-trivial task of modelling user context numerically. Conceptually, the skyline is the subset of all tuples that are top-ranked for some selected user context [11,12].

More formally, consider two services s and t represented by tuples $\langle a_1, \ldots, a_i, \ldots, a_m \rangle$ and $\langle b_1, \ldots, b_i, \ldots, b_m \rangle$, respectively. Service s is said to dominate service t, if s has an equal or better value than t for every attribute and a strictly better value for at least one of the attributes; that is, if $(\forall i \in [1,m], a_i \geq b_i) \wedge (\exists i \in [1,m], a_i > b_i)$. The skyline, then, is the set of those services that are not dominated by any other service. Unlike aggregation, skyline results in a set of incomparable services, rather than a total ordering, and unlike aggregation, the cardinality of the resulting set is entirely data dependent as a result of being context agnostic.

3. Motivation

Some practical challenges arise when implementing dynamic service discovery, because the attributes are dynamic. Ignoring these challenges comes at a definite cost. A service that is best suited for a particular user context can be penalized due to an inferior evaluation mechanism. Using our telescope example, we examine how these challenges arise and specifically why they constrain the applicability of the aggregation and skyline techniques. Assume there are hundreds of telescope web services available around the world. To a user interested in photographing a particular star, only a subset of these services is of particular use. Some telescopes are located in the wrong hemisphere. With others, the user may not have a service-level agreement (SLA) that permits her to use that telescope. By considering all such attributes, ideally, one can reduce the set of telescopes to a small subset that is functionally equivalent. This is referred to as functional selection: capturing a set of predefined values. Consider Table 1 depicting attributes and their values captured on four different days for three different telescope services. In this table, the functional attributes are assumed to be satisfied, denoted by a Y for yes, on all four days. After functional selection, the service subset needs to be ranked and possibly further pruned with respect to its dynamic, quantitative QoS attributes [15,16].

The values of all of these attributes fluctuate over time, perhaps even wildly, which complicates the matter of selecting the web service most likely to deliver the highest quality photographs. A knowledge base can maintain an expert-derived threshold value for each attribute, in each domain, to ensure an adequate quality of service.

Because the values are captured or possibly computed at run time, they may not always be available, perhaps they did not arrive on time or were lost due to network problems. Consider what happens when the observatory weather station cannot be reached, the service provider does not provide a value for contrast or a power outage takes out the sensor that measures these values. In these scenarios, the telescope should not necessarily be discounted from consideration, but neither the skyline nor the aggregation function is well-defined over these null values.

Null values are not the only concern. Consider visibility: even on a superb day, there can be patchy cloud cover. A particular measurement of visibility, one that captures the precise value at an exact moment in time, might signify that the visibility is poor, when in fact, there is only a momentary lapse before the cloud passes over. Another issue is latency: perhaps a particular telescope is generally slow to respond, but right after the service has been restarted, there are few connected users; a measurement at this time would overestimate the quality of the service.

Table 1. An example set of telescope web services with multiple dynamic QoS attributes, as measured on four consecutive days. Y, yes.

Telescope	Context/QoS attributes	Days			
		1	**2**	**3**	**4**
T1	functional	Y	Y	Y	Y
	availability (%)	20	40	50	60
	user preference/contrast	8	9	8	5
	time/dayLight	8	9		5
	user preference/composition	8			5
	latency (ms)	6	2		3
	weather/visibility (%)	70	90	90	80
	location/viewing angle	8	8	4	5
T2	functional	Y	Y	Y	Y
	availability	85	93	65	89
	user preference/contrast	8	9	4	5
	time/daylight	8	9	5	5
	composition	8	9	9	5
	latency	9	8		9
	weather/visibility	90	80	80	90
	location/viewing angle	8	6		8
T3	functional	Y	Y	Y	Y
	availability	30	60	30	60
	user preference/contrast	8	9	4	1
	time/daylight	8	9		1
	composition	8			3
	latency	5	7	4	1
	weather/visibility	70		70	50
	location/viewing angle	8	9	9	9

3.1. Manifestation of the Challenges

Both aggregation and skyline have practical issues when applied to dynamic web service discovery.

Null values are perhaps the most debilitating, because they introduce increased incomparability. The summation on which aggregation depends is ill-defined in the presence of null values. The biggest

drawback of skyline is that it often is insufficiently selective, especially as the dimensionality increases; introducing more incomparability via null values only serves to exacerbate this problem. A conceivable approach is to replace null values with zeros or estimates, but zeroing values can be unnecessarily harsh, especially when the null arises from network issues with an external data source.

Temporary fluctuations and unreliable measurements can significantly affect the skyline. A service that is consistently part of the skyline because of a maximal value on one attribute could be dropped at run-time.

The context should affect the relevancy of dynamic attributes. For instance, availability is usually extremely important for quality-driven service selection. However, the importance of this attribute can be tempered if the user decides to wait for a higher quality service that is temporarily unavailable, rather than settle for a lower-quality service that is immediately available.

Ideally, one could apply both well-established selection approaches—aggregation and skyline—to dynamic web service selection; however, the challenges detailed above obfuscate QoS comparisons between services. The definition of skyline (or, more precisely, of dominance) requires a well-defined, ordinal relationship (*i.e.*, the comparison operators $<$ and $>$ must be well defined for the attribute values of any two services). Null values invalidate this, because they are incomparable to non-null values. Snapshot values distort this, because they cause the ordinal relationship among services to fluctuate wildly. Thus, we need to restore the relationship somehow when null and fluctuating values arise.

For aggregation, the problem is even trickier, because the scoring function relies on attributes that are real valued, not just ordinal. Thus, it is important that, for example, the difference between three and one is larger than the difference between three and two, since this will affect the aggregate score.

Naturally, any technique that supports the case of aggregation can be used to re-establish the concept of dominance, because the ordinal operators $<$ and $>$ can be applied to any real-valued attribute. The goal is then to derive a reasonable value for null and fluctuating attributes to assess the magnitude by which the values, if known, would differ from the other known values.

4. Related Work

In the past, the ranking of services by QoS properties has been addressed by researchers in two different ways: (1) by considering static attributes [9,17]; and (2) by taking a snapshot of the current values of the attributes [14,18]. The former squanders the opportunity to improve the user experience dynamically at the time the user is actually selecting the service. In our telescope example in Table 1, capturing a measure of the availability of telescope service T3 on Day 1 and using that information to evaluate the service on Day 4, when the value has changed substantially, are misrepresenting reality. Nonetheless, the work done here is still important, because it introduces the idea of using aggregation querying in the context of web service discovery.

Regarding the second option, capturing instantaneous measurements of QoS properties ignores the practical issues that arise naturally when using dynamic attributes. For example, on Day 3, the latency measurement for T2 is null, but one should be lenient, because, historically, this attribute performed well.

Skoutas *et al*. [14] introduced the concept of dominance and skyline into the field of web service discovery. They examine every subset of the QoS attributes and count the number of subsets in which

a given service is dominated. The services can then be ranked in ascending order of score. In this way, they combine the notions of skyline and aggregation without having to model user context. They also present several algorithms to determine the top k web services based on several scoring functions: dominated score (TKDD), dominating score (TKDg) and dominance score (TKM). They improved the performance by having boundaries for each object, consisting of the lowest and highest values of the dominating scores. Therefore, instances (*i.e.*, attributes of a service) are maintained in three lists: maximum, minimum and current values of instances for a service. This approach helps with static service discovery, where values of attributes are available, but cannot support missing values.

Hadad *et al.* [18] proposed an algorithm, Transactional and Quality-of-Service (TQoS) driven selection, that is embedded in the transactional service selection for web service composition. The algorithm is designed for local QoS optimization; that is, the selection of the qualifying services for each activity that fulfils the transactional requirements. The algorithm's input is a workflow that includes a set of activities to fulfil a request. The output is a set of the services for each activity in the workflow with similar transactional properties, but dissimilar non-functional properties. The TQoS selection uses user preferences as a weight assigned to each QoS criterion. Then, the web service is scored through a conventional scoring function to rank services and to find the one with the maximum score. If there are multiple services with the same score, one of them is chosen randomly. Similar techniques are applied to select the maximum score path for web service composition.

Possibility theory is a mathematical theory for dealing with certain types of uncertainty, which has been used to tackle the issues of skyline for uncertain QoS attributes in web service discovery [19]. This work is based on a possibility distribution that contains all possible values of QoS attributes for each service. The authors propose two skyline algorithm extensions—pos-dominates and nec-dominates—to compute whether service s_i dominates service s_j in accordance with the threshold values computed from the possibility distribution. As far as we could tell, this work has not been applied to dynamic web service discovery.

Dustdar *et al.* [20,21] proposed the QoS-aware Vienna Runtime Environment for Service-oriented Computing (VRESCO) run-time environment for dynamic binding, invocation and mediation. They describe functional attributes in their service meta-data model. These QoS attributes can be specified manually using a management service or measured automatically and integrated into VRESCO. For each service, a schedule can be defined that specifies when the monitor should trigger the measurement and publish their values. The average QoS values can be aggregated based on the information stored in the QoS events. This approach cannot support fluctuating or unavailable values. The challenge for this algorithm lies in capturing and computing dynamic attributes at query time.

Several approaches in web service selection share their conceptual underpinning with the trust and reputation evaluation techniques by Vu *et al.* [22] and Doshi *et al.* [23]. In these approaches, user feedback and provider reports regarding generic QoS attributes are the principal evaluation and selection criteria. Then, the QoS data are the trustworthiness of the reported quality of services from both users and providers. Vu *et al.* measure the credibility of the user and the provider claims to improve the service ranking by a trusted third party. They also employed a time-series forecasting techniques to predict the future quality of the service based on these data. Doshi *et al.* introduce the Wisp framework to integrate the trust models

into service composition. Then, based on the users' experiences with services, the certainty is updated. They also measure the service honesty, competency and reliability.

SDDS (static discovery and dynamic selection) by Pahlevan and Müller is our initial framework for integrating static and dynamic attributes during web service discovery [24,25]. We examined context-sensitive QoS attributes [26] and their use in selecting services using aggregation and skyline-based algorithms [27]. Table 1 shows an example with SDDS-determined attributes, where the values are computed in a variety of ways; for example, visibility is obtained from weather services, such as AccuWeather http://www.accuweather.com/. Other attributes that refer to context, such as environmental conditions or user preferences, can be obtained via sensors or directly from the user. In her PhD thesis, Pahlevan extended this work by providing a full formalization and description of our aggregation and skyline-based algorithms and then presented a thorough experimental evaluation [28].

5. Dynamis Service Selection

Our Dynamis service selection approach to handling dynamic attributes is complementary to existing web service selection techniques; Dynamis is an ancient Greek word (δύναμις) meaning "power" or "force." This is because any static selection mechanism for evaluating which services are functionally equivalent is in fact a prerequisite step of our solution. Moreover, existing methods for dynamic attributes that are based on snapshot measurements are a special case for our algorithm.

5.1. Dynamis: Histogram, ART, Top k and Skyline

Dynamis addresses the aforementioned problems of null and fluctuating values effectively. Rather than taking an instantaneous measurement of each attribute, we collect the ν most recent measurements in a histogram. In this case, null values only become a concern when all of the last ν measurements are null, at which point, it can safely be assumed that there is something significantly wrong with the property.

Neither aggregation nor skyline are defined for histograms (*i.e.*, collections of measurements). A natural approach is to define a function on a histogram to convert it into a sensible value. We use the area-right-of-threshold (*ART*) function, for this purpose. Specifically,

$$Area = ART_s(A_i, \tau_i)$$

is the number of measurements for service s for attribute A_i that are greater than threshold τ_i. The intuition for the area-right-of-threshold comes from the idea of integrating in service selection the most important part of the histogram.

Now, we specify both the aggregation and the skyline problems on histograms.

The top k histogram aggregation problem is to retrieve from the set of services $\mathbb{S}$, with attributes:

$$\mathbb{A} = \langle A_1, \ldots, A_m \rangle$$

given a weight vector:

$$\vec{w} = \langle w_1, \ldots, w_m \rangle$$

Algorithm 1 Dynamis aggregation.

Inputs *a set S of functionally equivalent web services;*
a set $\mathbb{A}$ *of dynamic attributes for each service;*
a threshold vector $\vec{\tau}$ *where* τ_i *is a threshold for attribute* A_i*;*
a weight vector $\vec{w}$ *where* w_i *is a weight for attribute* A_i *(i.e.,context);*
a time t where the evaluation takes place;
a time window ν *over which attributes are considered;*
a scalar k to indicate the number of services to return.
Output *The top k services with respect to context* $\vec{w}$
Procedure DYNAMISAGGREGATION$(S, \mathbb{A}, \vec{\tau}, \vec{w}, t, \nu, k)$
  for each $s \in S$ **do**
    $Score[s] \leftarrow 0$
    for each $A_i \in \mathbb{A}$ **do**
      $h \leftarrow \text{Histogram}(A_i, t-\nu, t)$
      $Area \leftarrow ART(h, \tau_i)$
      $Score[s] \leftarrow Score[s] + w_i \times Area$
    end for
  end for
  $Dynamis \leftarrow \text{Sort}(Score, >)$
  return $Dynamis[1:k]$ ▷ return the top k tuples
end Procedure

and a threshold vector:

$$\vec{\tau} = \langle \tau_1, \ldots, \tau_m \rangle$$

the top k ranked services with respect to aggregation function:

$$\sum_{i=1}^{m} w_i \cdot ART_s(A_i, \tau_i)$$

Instead of considering snapshot values for an attribute A_i, we compute $ART_s(A_i, \tau_i)$ from A_i's histogram (*cf.*, Algorithm 1).

Similarly, we define the skyline of histograms

where a service $s \in \mathbb{S}$ dominates another service $t \in \mathbb{S}$ if:

$$ART_s(A_i, \tau_i) \geq ART_t(A_i, \tau_i)$$

for each $i \in [1, m]$, there exists an $i \in [1, m]$ for which the above inequality holds strictly for ($>$). The skyline is the set of those services not dominated by any other functionally-equivalent web service.

Since the result of the skyline is not ranked, we employ the notion of k-dominance introduced by Huang *et al.* [29]. Thus, we compute a score for each service as follows. Suppose $\langle a_1, \ldots, a_n \rangle$ is the tuple of attribute values of service s in its skyline. We now consider the sorted lists of values for each of these attributes considering the services in its skyline.

Algorithm 2 Dynamis skyline.

Inputs *a set S of functionally equivalent web services;*
a set $\mathbb{A}$ of dynamic attributes for each service;
a threshold vector $\vec{\tau}$ where τ_i is a threshold for attribute A_i;
a time t where the evaluation takes place;
a time window ν over which attributes are considered;
a scalar k to indicate the number of services to return.
Output *The top k services that best fit the request*
Procedure DYNAMISSKYLINE($S, \mathbb{A}, \vec{\tau}, t, \nu, k$)
    $H \leftarrow \emptyset$
    for each $s \in S$ **do**
        Create a tuple h of m elements.
        for each $A_i \in \mathbb{A}$ **do**
            $histogram \leftarrow \text{Histogram}(A_i, t-\nu, t)$
            $h[i] \leftarrow ART(histogram, \tau_i)$
        end for
        Add h vector to H.
    end for
    Compute skyline SkyL_H of H.
    for each $h \in \text{SkyL}_H$ **do**
        $Score[h] \leftarrow \sum_i^m Rank_i(h)$
    end for
    $Dynamis \leftarrow \text{Sort}(Score, <)$
    return $Dynamis[1:k]$ ▷ return the top k tuples
end Procedure

For each value a_i, we check to see what its rank is in the sorted list. By r_i, we denote the rank of a_i in the list. Let $r_s = \langle r_1, \ldots, r_m \rangle$ be the vector of the ranks for s. Then, the scoring function simply sums up these ranks as follows.

$$Score(h) = \sum_{i=1}^{m} Rank_i(h) = \sum_{i=1}^{m} r_i.$$

Finally, we retrieve the top k services from the skyline according to this scoring function (*cf.*, Algorithm 2).

6. Evaluation

This area of research typically uses information retrieval systems evaluation methods [30–35]. A test collection includes a set of service offers $\mathbb{S}$, a number of service requests $\mathbb{Q}$ and relevance judgements (RJ). The relevance judgements are identified by how, and to which degree, the offer is relevant to a request by human experts. Evaluation involves the comparison of RJ with a system under evaluation; we simply compare the computed rank with the one induced by the relevant judgement. Therefore, a reliable RJ is critical for a precise and effective evaluation. As these evaluation techniques become ubiquitous, the accuracy and meaningfulness of the result is subjective [36]. Due to conflicts in human judgement,

research has been conducted to determine how to obtain reliable relevance judgements [37,38]. In this section, we first recall some of the ubiquitous IR measurements and then explain the extended version of these metrics that have been adopted in the service discovery domain and, finally, describe our evaluation methodologies towards a trustable evaluation measurement.

The existing evaluation approaches are based on binary relevance: typically precision and recall. Binary relevance is limited to identifying whether the result is relevant or not. Recall is the proportion of the relevant items that are in the set of returned results, and precision is the proportion of items in the returned results that are relevant [38].

Tsetsos *et al*. [39] use a more flexible rubric in the form of a fuzzy linguistic variable where matchmaking is the degree of match. The values of this variable are fuzzy terms, such as irrelevant, slightly relevant, somewhat relevant, relevant and very relevant. Jervlin *et al*. [40] proposed a cumulative gain (CG), at rank i, CG_i, to generalize these binary metrics to support the notion of graded relevance. It measures the gain g the user receives by scanning the top i items of ranked list j. Gain g depends on the application. Specifically, CG_i is defined as:

$$CG_i = \sum_{j=1}^{i} g(r_j)$$

As shown below, we calculate the gain in our setting based on real quality values rather than on RJ, as in classical IR.

6.1. Evaluation Setup

To evaluate our algorithm, we need a high-quality web service application. Gathering weather data from a collection of weather stations, matching the location of real-world telescopes has several issues. First, telescopes tend to be located at a few high-altitude sites with exceptional visibility; this makes for a small number of weather stations. Second, there are large gaps where the automated weather observation station (AWOS) does not record the weather conditions, because they are unserviceable. This results in a dataset that is too small to be used for validating our algorithms effectively.

A better strategy is to use data for stocks tracked, for example, by the S&P 500 index. In particular, we used stock tracking data for the period of 1 January 2009 to 22 November 2011, almost two years' worth of data. A stock can be considered as "services to your money." The data collected includes the low, high and closing price for the stock, as well as transaction volumes for each trading day; we use these as proxies for dynamic non-functional telescope attributes. We also tracked the Dow Jones Industrial Average (DJI) over the same period, which we use as an additional attribute when evaluating service quality (stock desirability). The full dataset is available on GitHub at https://github.com/Atousa/DYNAMIS.

Our evaluation targets the use-case of a consumer searching for a web service that provides functionality needed in some application being developed. The individual services that could be chosen are in fact proxied by the various stocks that we tracked. This allows us to compare some 500 services (stocks) by evaluating several attributes varying with high frequency, such as price (P) and trading volume (V), and to compare service selection strategies (stock selection). Note that the analysis should not be considered investment advice.

After gathering data, we compute the following additional attributes to be used during evaluation: change in price (ΔP), change in volume (ΔV) and relative strength *vs*. Dow Jones index (RSD); d and $d-1$ are a specific trading date and its immediate predecessor respectively, while s represents a specific stock in our dataset. Formally, we define:

$$\Delta \mathrm{P_s} = \frac{\mathrm{P}_d - \mathrm{P}_{d-1}}{\mathrm{P}_{d-1}}$$

$$\Delta \mathrm{V_s} = \frac{\mathrm{V}_d - \mathrm{V}_{d-1}}{\mathrm{V}_{d-1}}$$

$$\mathrm{RSD_s} = \Delta \mathrm{P}_s - \Delta \mathrm{P_{DJI}}$$

We now demonstrate how our Dynamis approach increases the effectiveness of selection by providing finer filtering and ranking of the services in the registries.

The evaluation task is to rank a list of web services according to their relevance for a given consumer request (identify stocks that could generate profits). Using the data collected, we evaluate the quality of our picks by looking at subsequent profits or losses of the stocks selected, thus avoiding human judgement conflicts and their consequences in the evaluation result (*cf*., Section 6) [37].

Most of the analysis is illustrated using one stock from the technology sector: INTC (Intel Corp.); but we also look at AAPL (Apple Inc.) and GOOG (Google Inc.) when we need to compare stock performance. For clarity, the data for a given stock is shown consistently in the same color throughout this section. Detailed results for all of the stocks tracked are available at GitHub https://github.com/Atousa/DYNAMIS/.

Figure 1 shows significant variability in the trading volume data from one day to the next; reported here is the percentage change with respect to the previous trading day's transaction volume (ΔV). However, as shown in Figure 2, by computing the histogram of the data over a period of time, some of the information contained therein becomes more readily visible. For example, we can see that the ΔV fluctuation is usually within ±50% of the previous day's volume. We rely on the fact that histograms expose high-level information contained in a dense dataset, particularly the various quantiles.

The area-right-of-threshold (ART) function counts the data in the histogram whose value is greater than the specified threshold; in Figure 2 to the right of the dashed line. That is, the ART function counts the number of points in a given quantile. Figure 3 shows the evolution of $\mathrm{ART}(\Delta \mathrm{V}60, 10\%)$ over time *(i.e.*, for every day we compute ΔV over the following 60 trading days), which then counts the number of elements to the right of the 10% threshold. Note that ART() is an integer-valued function. On a given date, the daily change is either greater than the threshold or not; respectively contributing the value one or zero towards the value of ART(), respectively.

6.1.1. Scoring and Ranking

Now that we have collected data and computed various metrics, we aggregate attributes to determine the desirability of a specific service.

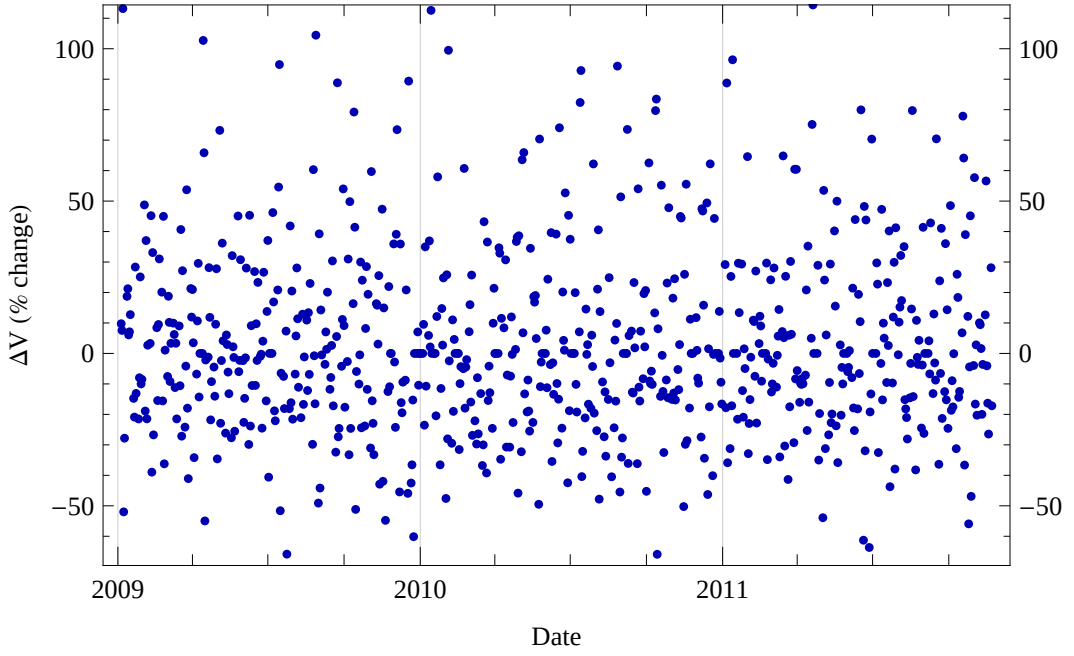

Figure 1. INTC (Intel Corp.) transaction volume change with respect to previous trading day.

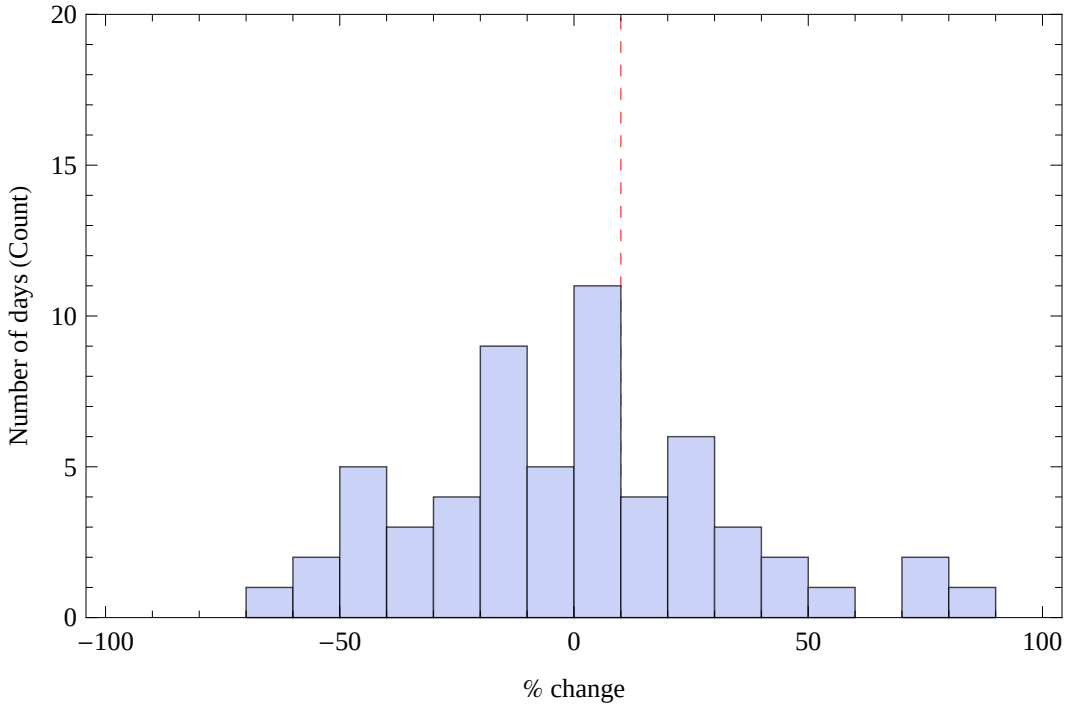

Figure 2. INTC histogram of trading volume daily change, starting from 13 October 2009 over 60 trading days. The dashed vertical line is the 10% threshold. $\mathrm{ART}(\Delta \mathrm{V}60, 10\%) = 19$.

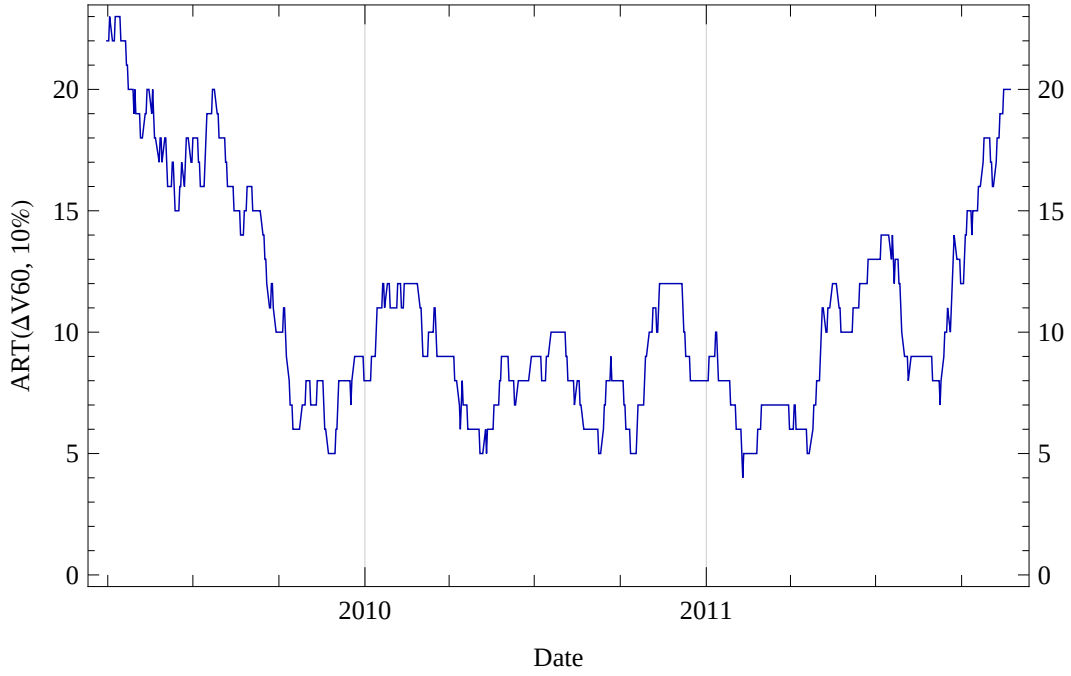

Figure 3. Evolution of the ART$(\Delta V60, 10\%)$ integer-valued function over time for INTC.

The scoring function computes a score for each stock: for each day in the snapshot aggregation approach and for a range of days in the Dynamis aggregation. The scoring function employs a weight vector (user preferences) in the snapshot aggregation approach and a weight vector, as well as a threshold vector in the Dynamis aggregation approach.

Let $\vec{w} = (w_1, \ldots, w_n)$ be the weight vector and $\vec{\tau} = (\tau_1, \ldots, \tau_n)$ be the threshold vector. Let $s = \langle a_1, \ldots, a_n \rangle$ be some service. For snapshot aggregation, the scoring function ($SAgg$) is:

$$SAgg(s) = \sum_{i=1}^{n} w_i \cdot a_i$$

For Dynamis aggregation, the scoring function ($DAgg$) is:

$$DAgg(s) = \sum_{i=1}^{n} w_i \cdot ART(a_i, \tau_i)$$

In both the snapshot and Dynamis aggregation, we use a linear combination of the attributes available to score the services. For our stock setting, we picked:

$$\vec{w} = (2.0, 0.1, 3.0)$$

These three weights correspond to ΔP, ΔV and RSD. The values reflect our opinion that the relative performance of a stock against a benchmark is slightly more important than the daily percentage increase, and the volume percentage change is least important.

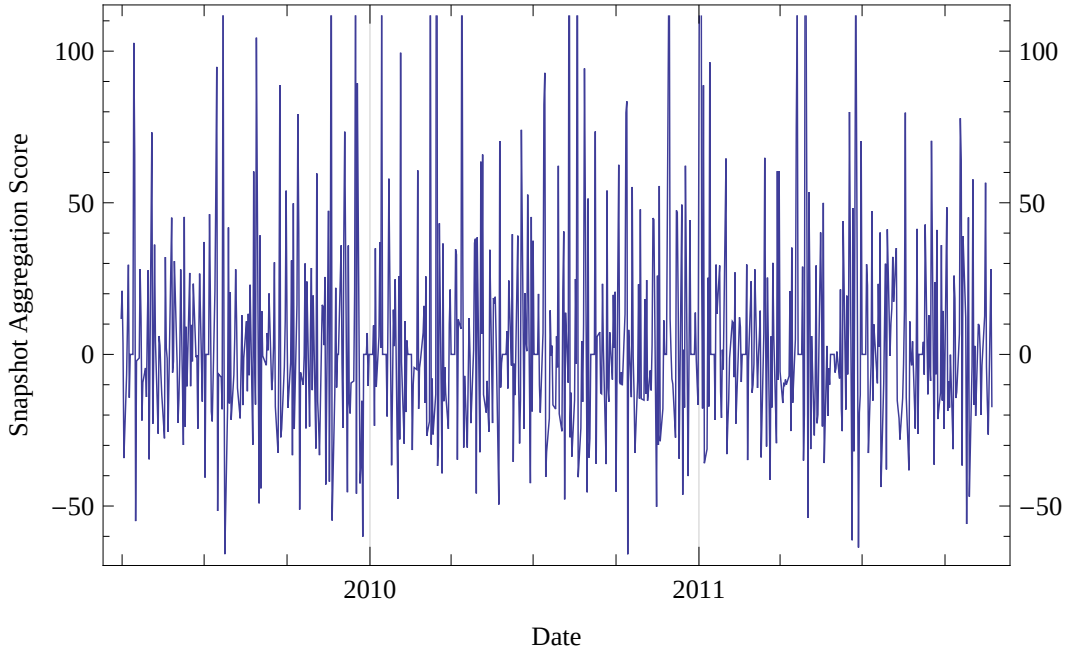

Figure 4. INTC score computed daily using snapshotting with the scoring function: $SAgg(\text{INTC}) = 3 * \text{RSD} + 2 * \Delta\text{P} + 0.1 * \Delta\text{V}$.

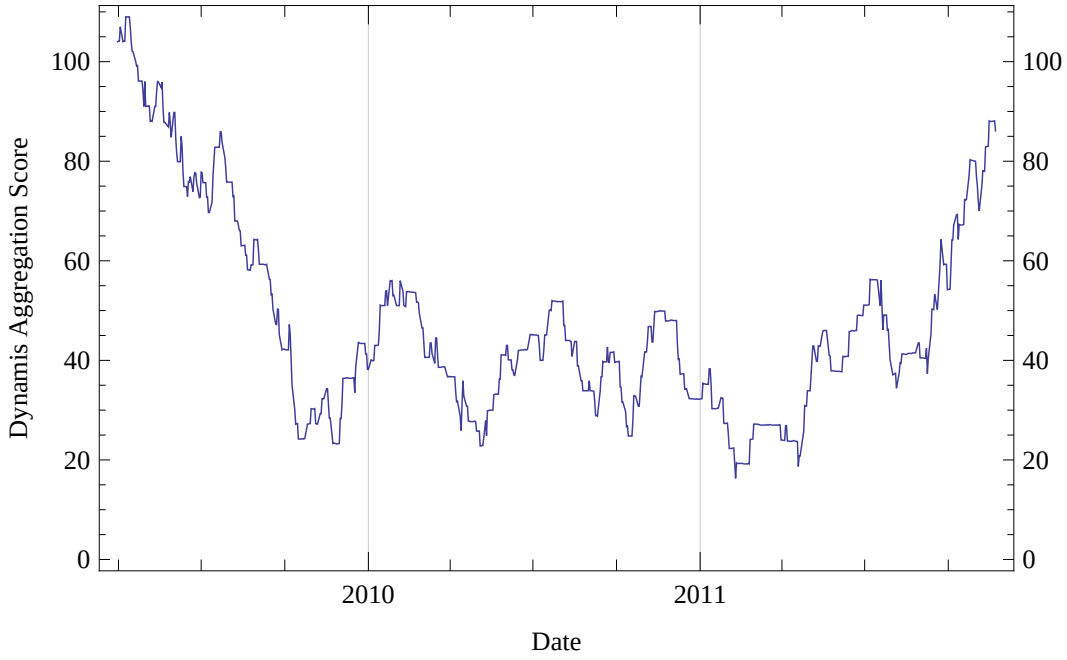

Figure 5. INTC score computed daily using Dynamis with the following scoring function: $DAgg(\text{INTC}) = 3 * \text{ART}(\text{RSD60}, 1\%) + 2 * \text{ART}(\Delta\text{P60}, 2\%) + 0.1 * \text{ART}(\Delta\text{V60}, 10\%)$.

The score computed with (*SAgg*) for INTC as it varies over time is depicted in Figure 4. For the Dynamis aggregation approach, the resulting scores vary more slowly over time (*cf.*, Figure 5).

In our dataset, we have scored every stock on every day. So far, we have looked at the score of a single stock as it changes over time. We can examine the scores for all of the stocks on a specific day, sorted

from best to worst. The position of a stock in this sorted list is its rank for that day; therefore, a high score results in a low rank: rank 1 is the best stock pick. On a specific date, say 3 October 2011, the curve depicted in Figure 6 represents the sorted stocks, and Table 2 lists the top 10 ranked stock tickers for that day. These present the ranking of tickers on a specific date, but the evolution of a given ticker's rank over time can also be graphed.

Table 2. Top 10 stocks on 3 October 2011, ranked with Dynamis aggregation. These correspond to the highest-scored tickers in Figure 6.

Rank	Ticker	Company
1	CF	CF Industries Holdings, Inc.
2	RRC	Range Resources Corp.
3	RHT	Red Hat, Inc.
4	TLAB	Tellabs, Inc.
5	NVDA	NVIDIA Corporation
6	GT	The Goodyear Tire & Rubber Company
7	JWN	Nordstrom, Inc.
8	SNDK	SanDisk Corporation
9	CTXS	Citrix Systems, Inc.
10	PFG	Principal Financial Group Inc.

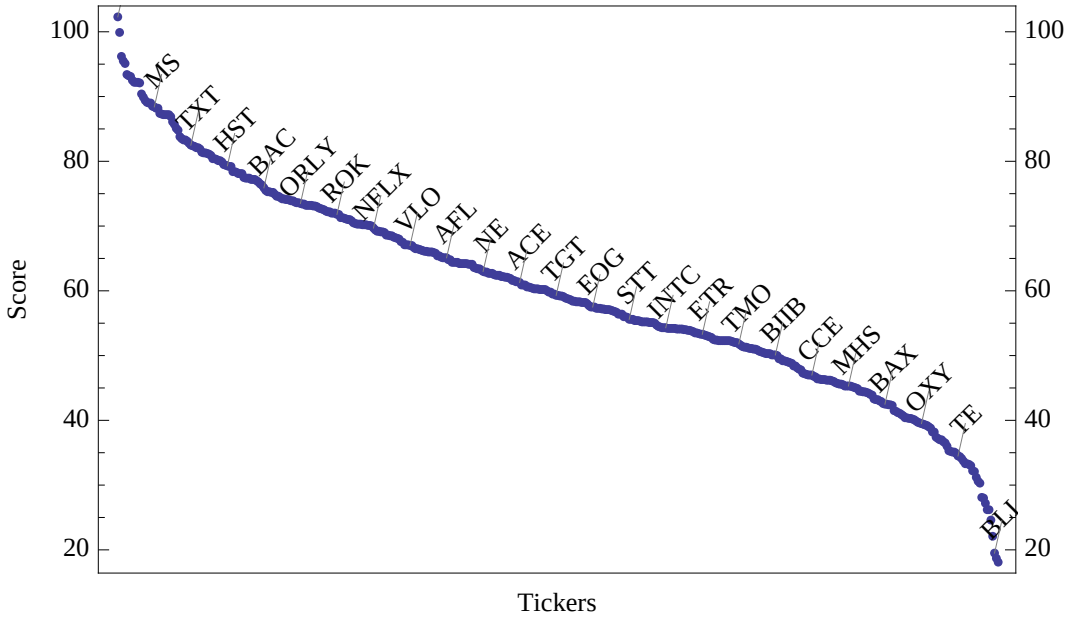

Figure 6. Sorted scores computed with Dynamis aggregation on 3 October 2011.

Figure 1 and Figures 3–5 are time-based representations of the data. In contrast, Figure 6 represents the stock tickers on a fixed date. The following figures are again plotted against time. In the case of INTC, the results, which are based on the snapshot and Dynamis aggregation techniques, are shown in

Figures 7 and 8, respectively. Note that for clarity, when graphing ranks, we shade the area under the curve; this allows us to visually distinguish ranks from other quantities graphed.

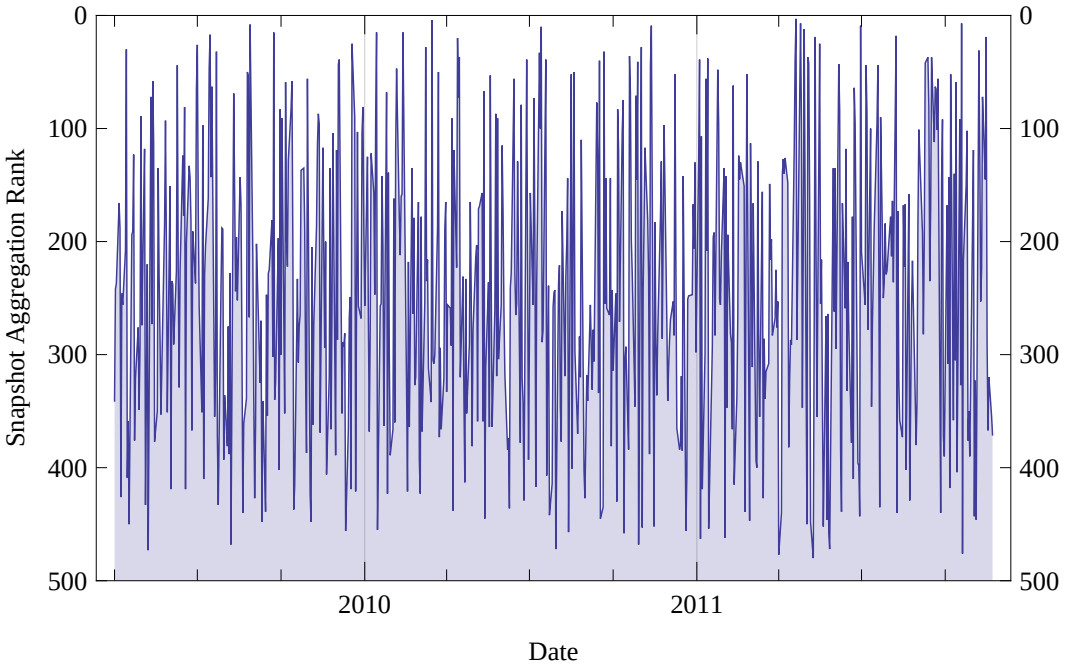

Figure 7. INTC rank as its score (snapshot aggregation) changes over time in a pool of 500 tickers. A lower-numbered rank (visually higher in the graph) means a better stock pick.

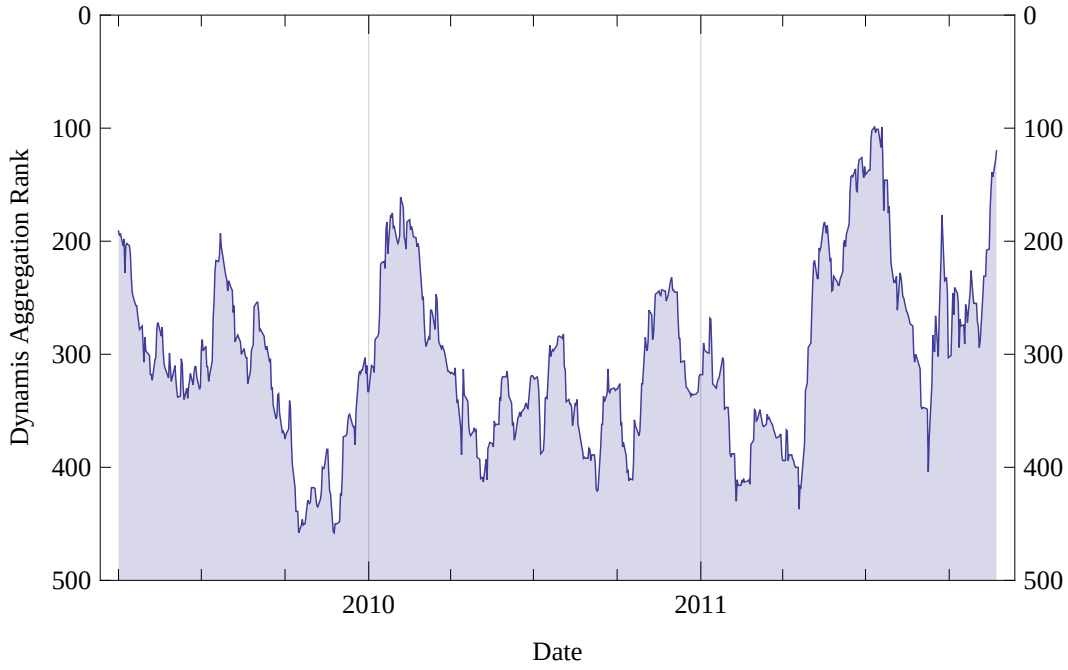

Figure 8. INTC rank as its score (Dynamis aggregation) changes over time in a pool of 500 tickers. A lower-numbered rank (visually higher in the graph) means a higher score.

6.1.2. Skyline

The daily evolution of the snapshot skyline rank for INTC looks much like Figure 7 and is not explicitly depicted here, because there is not much that can be understood from a visual depiction of such noisy data. The evolution of the Dynamis skyline rank is presented in Figure 9.

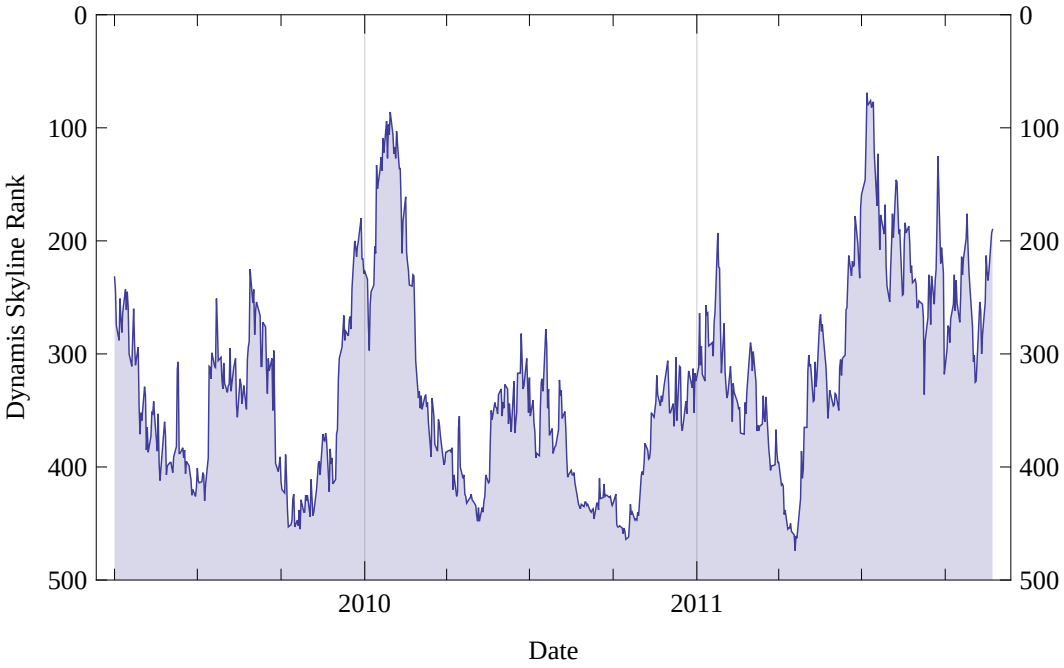

Figure 9. INTC Dynamis skyline rank as its score changes over time in a pool of 500 tickers.

6.2. Results

In order to compare and evaluate the rankings computed, we looked at three tickers from the technology sector: AAPL, INTC and GOOG. Figure 10 shows the relative price change of these stocks over the entire time period, normalized to start from the same relative value and overlaid on the same figure. This helps visualize the performance of the three stocks relative to each other.

Figure 11 shows the computed ranks using snapshot aggregation for our three stocks. The data are noisy enough that we chose not to connect the points; otherwise, we would have been faced with a solid block of color. Much like every graph seen so far, snapshotting gives us noisy data: the rank for a given stock will vary significantly from one day to the next. This may or may not be a disadvantage; there may be a fixed cost to switching services (e.g., a transaction fee), which could overwhelm the benefits of frequent change. However, there may be an advantage to changing the selected service frequently, since switching the provider load balances the services nicely. Therefore, the high frequency of change in ranks in the snapshotting technique cannot, in and of itself, be counted against the technique. However, in Figure 12, by computing the histogram of the data over a period of time, it becomes more readily visible. The snapshot skyline ranks are similar to Figure 11 in that they are scattered with no obvious trend. Figure 13 instead shows the ranks computed using the Dynamis skyline techniques.

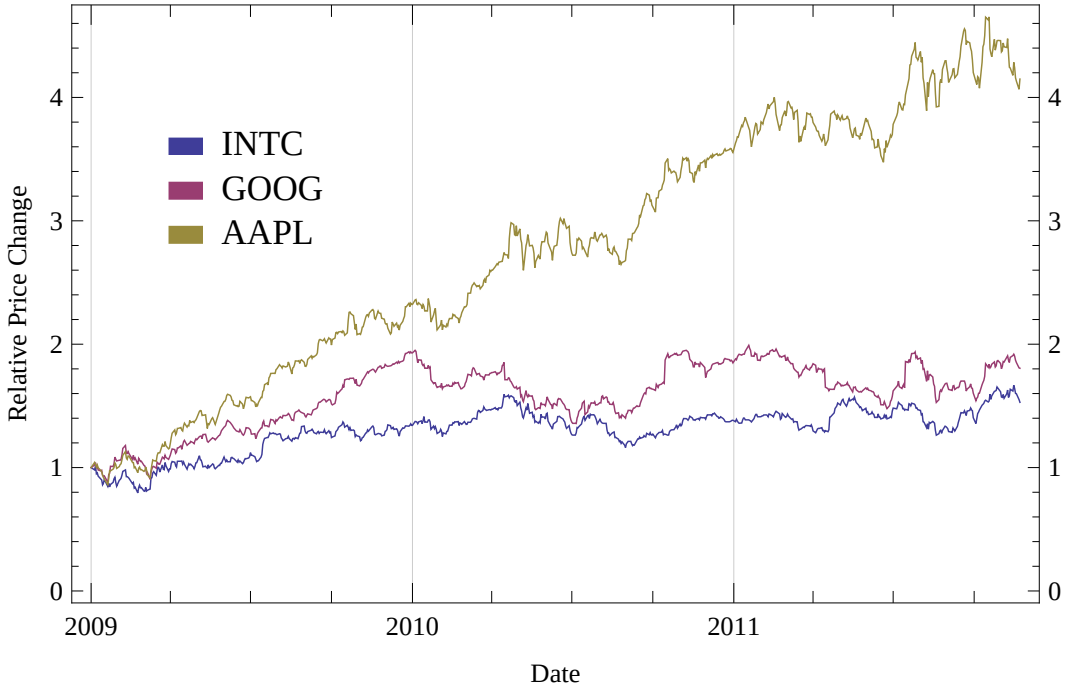

Figure 10. Relative price change of three stock tickers over the time period considered, normalized to start from the same relative value.

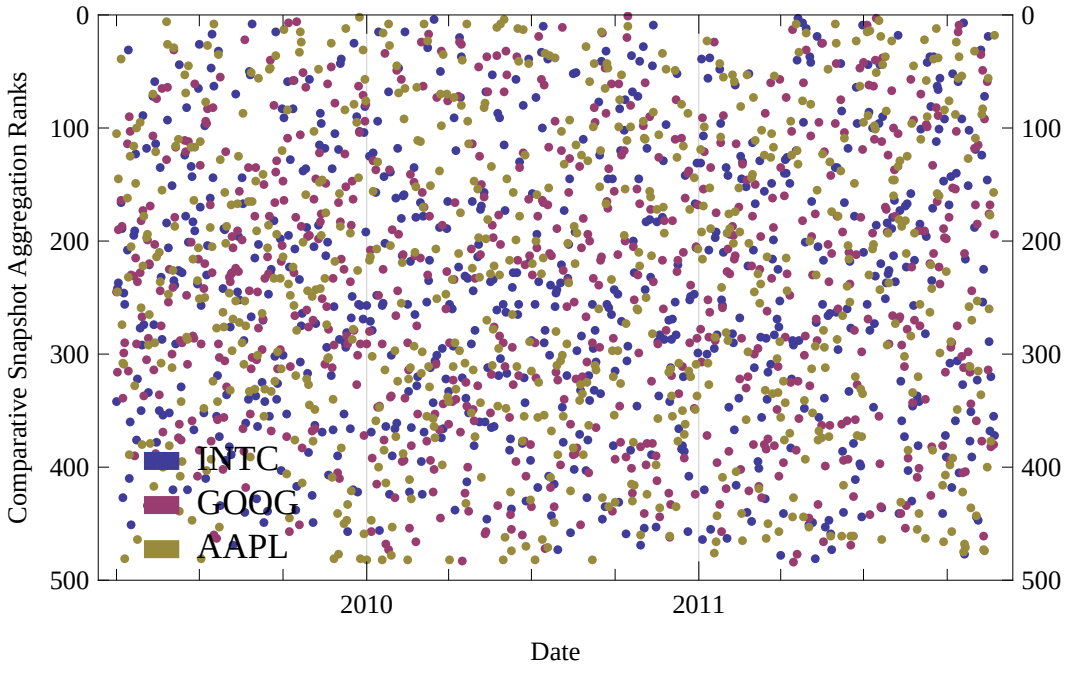

Figure 11. Snapshot aggregation rank of our three stocks over the time period considered.

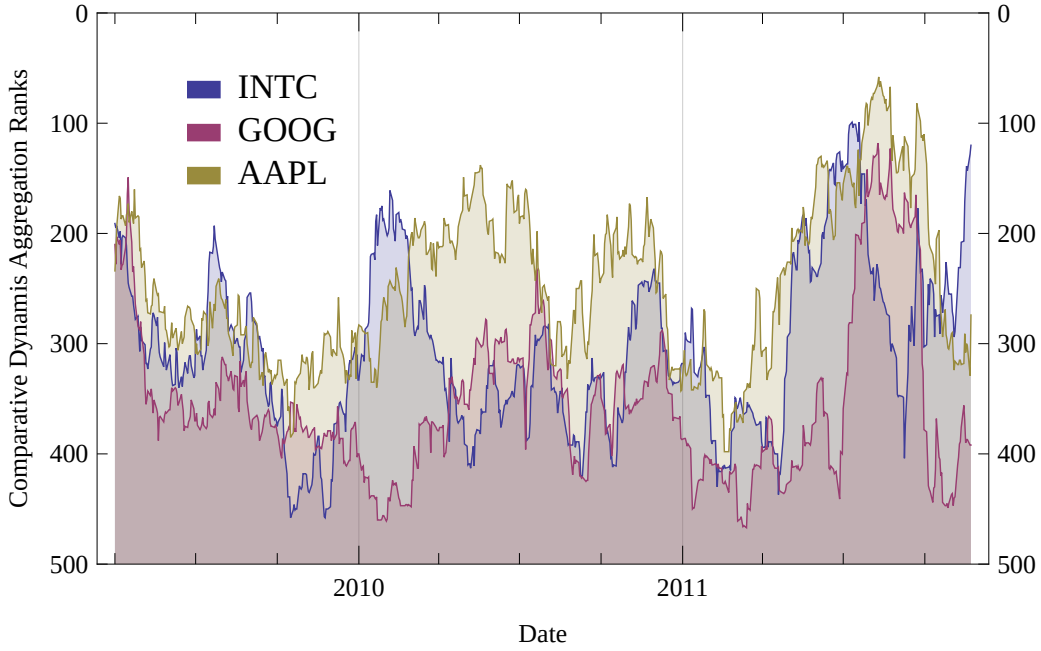

Figure 12. Dynamis aggregation rank of our three stocks over the time period considered.

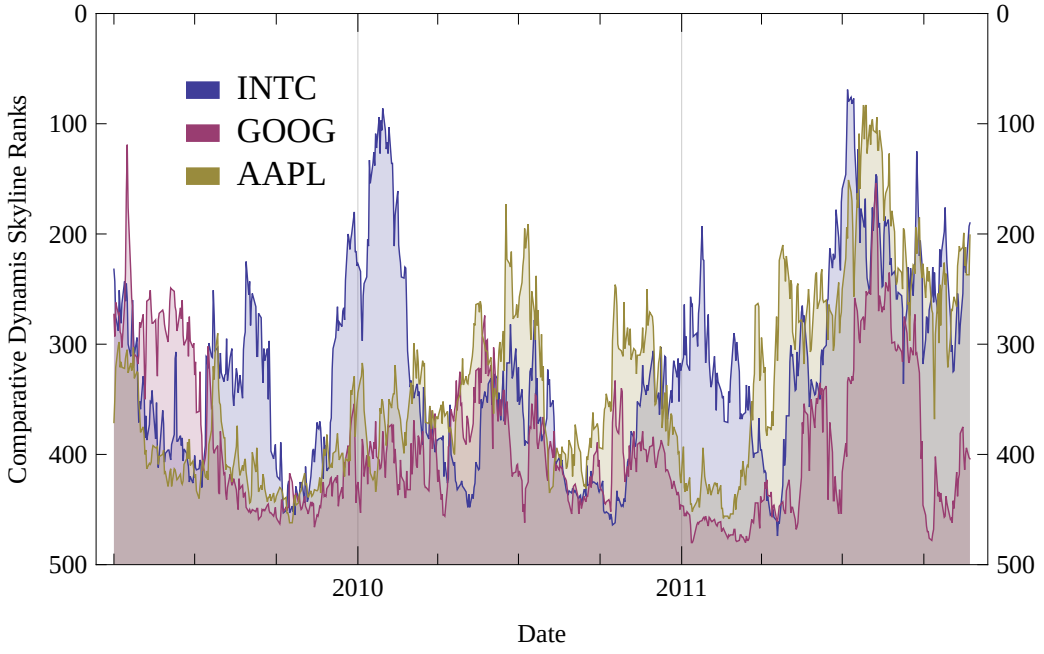

Figure 13. Dynamis skyline rank of our three stocks over the time period considered.

On a specific day d we select the top k stocks using our four methods. Specifically, we obtain the following top k rankings:

- L-SAgg using *SAgg*
- L-SSkyline using *SSkyline*
- L-DAgg using *DAgg*
- L-DSkyline using *DSkyline*

$$\begin{cases} L-SAgg = (s_1, s_2, \cdots, s_k) \\ L-SSkyline = (s'_1, s'_2, \cdots, s'_k) \\ L-DAgg = (s''_1, s''_2, \cdots, s''_k) \\ L-DSkyline = (s'''_1, s'''_2, \cdots, s'''_k) \end{cases}$$

In order to evaluate the success of these four methods, we now evaluate the cumulative gain obtained from purchasing the top k ranked stocks.

Using the historical data gathered for stock prices, we are able to follow the evolution of a hypothetical investment on day d for each of our top k picks; the period over which we follow that investment we call window w. We can compute the gain for stock s purchased on day d over the investment period w as:

$$\text{Gain}_d^s = \frac{\text{P}_{d+w}^s - \text{P}_d^s}{\text{P}_d^s}$$

We divide the gain by the initial price in order for our computations to remain independent of the magnitude of the price. This quantity is known as the arithmetic return or yield.

We assume a sale on the last day of the window, when the gains or losses are realized. At that time, we add the gains for each one of the stocks selected as part of the top k pick. In order to keep our results independent of the number of stocks selected, we divide the total gain at the end of the investment period by k. The cumulative gain (CG) is therefore:

$$CG_d = \frac{1}{k} \sum_{s=s_1 \ldots s_k} \text{Gain}_d^s$$

This quantity is technically known as the arithmetic average rate of return. Thus, a result greater than zero represents a net gain, and a result less than zero represents a net loss over the period. To illustrate this, assume we pick the top 10 stocks ($k = 10$) and track them over the course of one month, *i.e.*, 20 trading days ($w = 20$). This is illustrated in Figure 14, using the highest ranked stocks selected by the snapshot aggregation technique. Figure 15 shows these same gains depicted in a histogram. The mean gain is a modest 1.7%.

Since each day is an independent investment, we can show the evolution of the portfolio over time by compounding the cumulative gains. It is critical to realize that the CG_d only affects that fraction of the overall investment that was invested on that day, that is $1/w$. Let us explain this fact with an example: we shall invest a fixed amount of money x in each of the top k stocks selected by one of our algorithms, and keep the funds invested for w trading days. At the end of the period, each stock has a rate of return Gain_d^s, and the portfolio of k stocks has a cumulative gain $CG_d = 1/k \sum_s \text{Gain}_d^s$. That is, we invested $x \times k$ on day d and recovered $x \times k \times CG_d = x \times \sum_s \text{Gain}_d^s$ on day $d + w$.

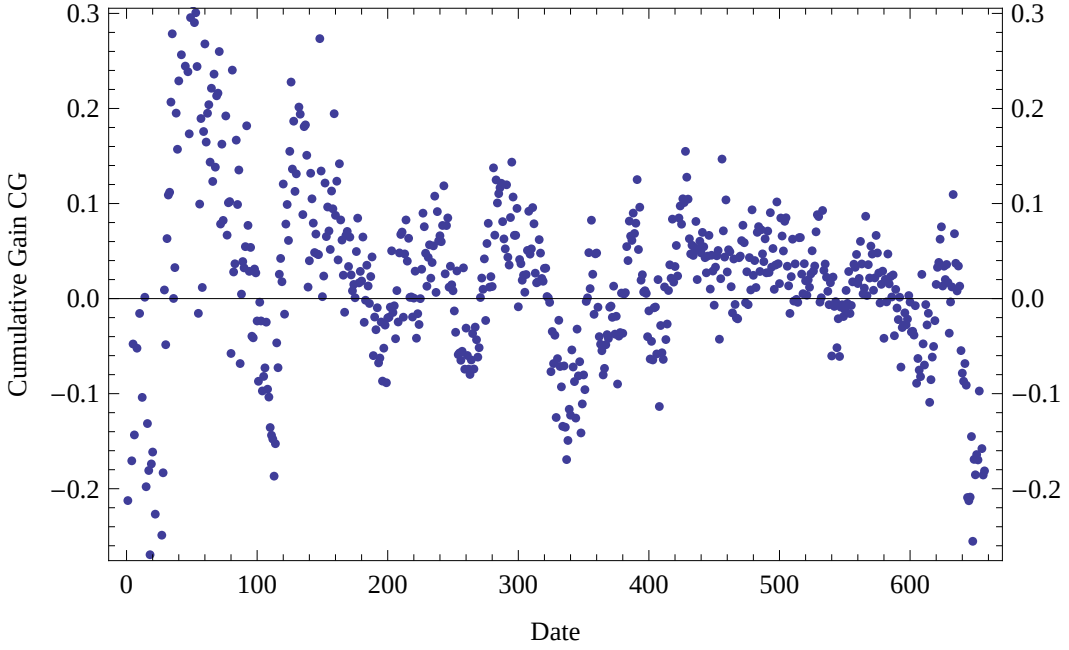

Figure 14. Gain realized by picking the top 10 tickers chosen from the snapshot aggregation algorithm and selling after 20 days.

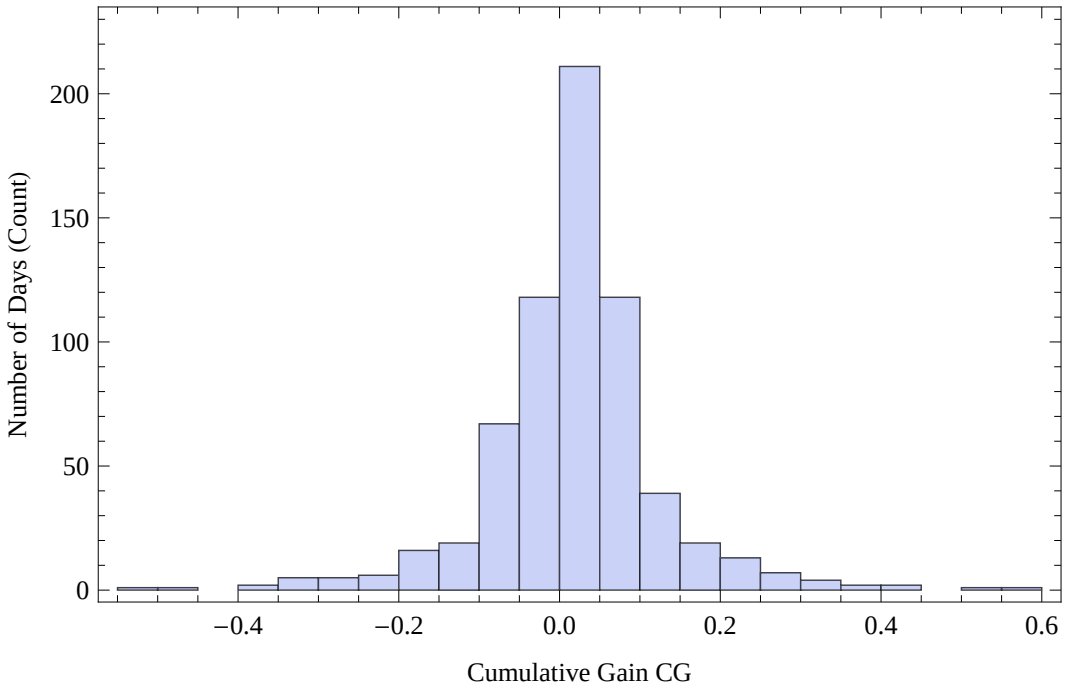

Figure 15. Histogram of the gains realized by picking the top 10 tickers chosen from the snapshot aggregation algorithm and selling after 20 days.

Now, assume that we repeat this process every day, over a long period of time. Since we are investing a fixed amount per day, the total amount of money that needs to be available at the beginning of the process is Funds $= x \times k \times w$. The cumulative gain CG_d on any given day only affects $1/w$ of the total sum invested. A little algebra will show that:

$$\frac{1}{w} \times \text{Funds} \times CG_d = \text{Funds} \times \frac{1}{k} \sum_s \frac{\text{Price}^s_{d+w} - \text{Price}^s_d}{w \times \text{Price}^s_d}$$

That is, the price difference is scaled by $1/w$. To compute the evolution of the overall gain (OG) over time, we compound the cumulative gains as follows:

$$OG = \prod_d \left(1 + \frac{CG_d}{w}\right)$$

The gain Gain^s_d and cumulative gain CG_d are both zero-based quantities; compounding these would converge to zero very quickly. Adding one to the quantity CG_d/w allows us to multiply them meaningfully. This is the same as saying that a 20% yield is equivalent to a factor of 1.2.

Figure 16 shows our four algorithms side by side. We observe that Dynamis aggregation performs best. Snapshot aggregation and snapshot skyline basically performed the same; they are indistinguishable in Figure 16; but in fact, they are not identical.

Finally, we compute the geometric average rate of return, or average yield Y, as:

$$Y = \sqrt[n]{\prod_{d=1}^{n} \left(1 + \frac{CG_d}{w}\right)} - 1 = \sqrt[n]{OG} - 1$$

Comparing the average yield Y at the end of the evaluation period in Figure 17, we see that Dynamis performs better than snapshotting, with Dynamis aggregation being the clear winner.

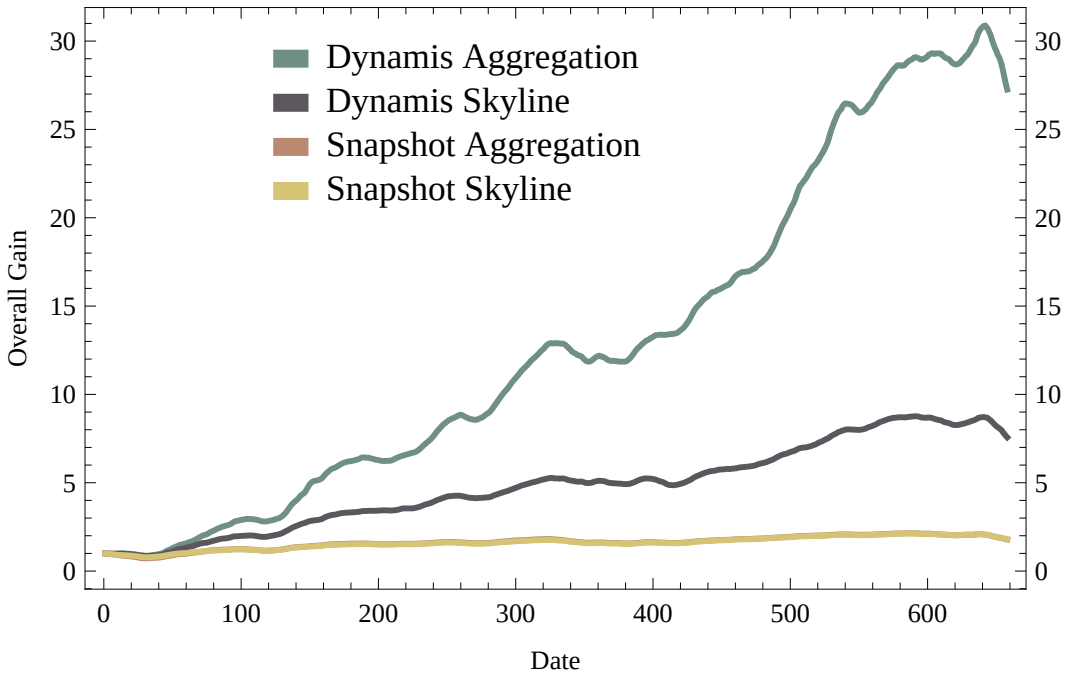

Figure 16. Overall gain (OG) with k = 10 and w = 20.

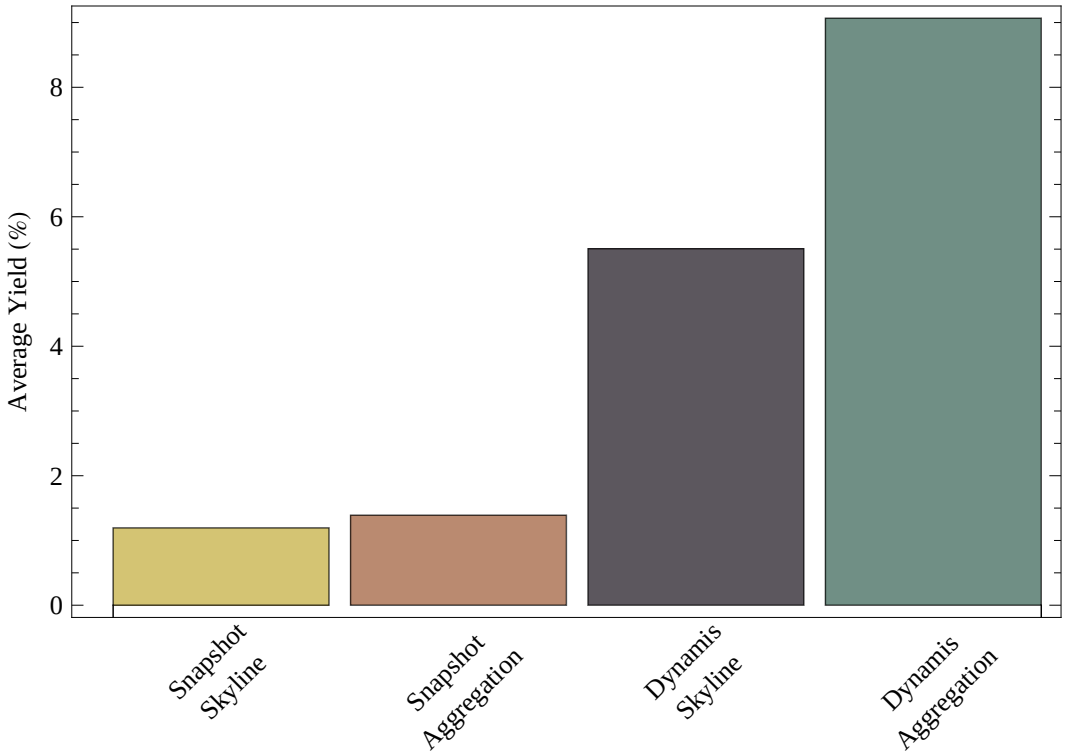

Figure 17. Average yield at the end of the evaluation period for our four algorithms with $k = 10$ and $w = 20$.

6.3. Generalization of Results

The previous section established that Dynamis performs better than snapshotting, and Dynamis aggregation performs better than Dynamis skyline. We now show that these results hold for other values of k and w. Figure 18 depicts the average yield Y for a 20-day trading window ($w = 20$) as we pick an increasing number of top k picks along the X-axis. The actual gain does not matter; the intent is to show that the relative performance of the algorithms remains the same regardless of the value of k. While the skyline and aggregation trend lines do converge, Dynamis remains superior compared to the snapshotting technique. Similarly, for a fixed value of $k = 10$, the relative performance of the algorithms remains the same as we vary w as shown in Figure 19.

While the yields may look surprisingly high, one must consider the fact that the overall market did well in the period under consideration. As a result, picking the top 10 performing stocks every day and holding these for a period of several weeks is expected to provide good returns. Consider however that the amounts invested are substantial as w increases and that the risk is high. Thus, we have shown that the relative performance of our algorithms remains the same as k and w vary, with Dynamis outperforming snapshotting in all cases investigated.

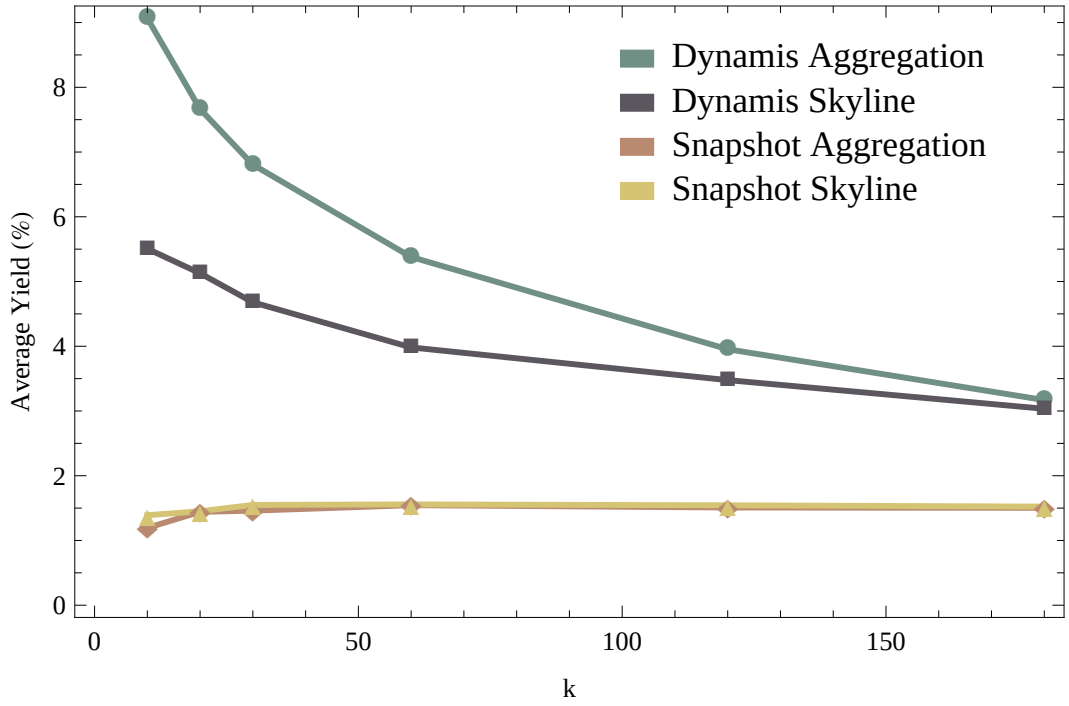

Figure 18. Average yield (Y) for a 20-trading day window ($w = 20$) as we pick an increasing number of top k picks along the X-axis.

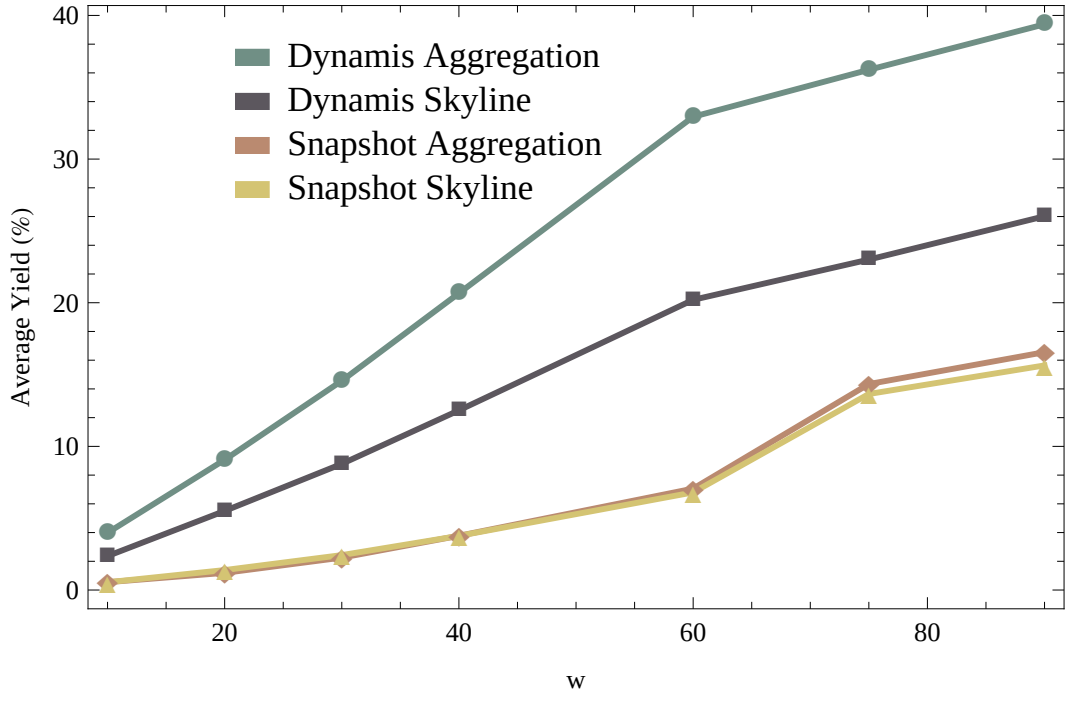

Figure 19. Average yield (Y) for a w-trading day window along the X-axis, as we pick the top 10 stocks.

The values reflect our opinion that the relative performance of a stock against a benchmark is slightly more important than the daily percentage increase, whereas the volume percentage change is not as important. Nonetheless, we obtain similar results by setting these weights to other reasonable values.

7. Discussion

We selected stock market data as a proxy for dynamic non-functional attributes, primarily due to the paucity of actual data on telescopes, since it represents a large source of high-frequency information. More importantly, the stock market allows us to evaluate the performance of our algorithms by comparing the performance of the stocks picked with respect to the market as a whole. This is something that cannot be done with weather data, because we do not have a way of evaluating the quality of a picture that could have been taken by a telescope, but, in fact, was not. The ability to run our algorithms over an unbiased dataset allows us to perform a quantitative performance analysis that would not be possible in a subjective context.

The implementation of our Dynamis algorithm is relatively straightforward: the algorithm lends itself to the stream processing paradigm, since it only uses local data, *i.e.*, data that are closely related in time or by stock. Several passes are needed over the data, with a small number of sorting steps as identified in the algorithm. We implemented our algorithm and evaluation system in Python including a Makefile to track data dependencies. The goal was flexibility, rather than run-time efficiency of the implementation. The latter was not an issue in our experiments.

8. Conclusions

A large number of web services performs the same function, but in different ways or with different qualities, that is they have different non-functional properties, such as quality of service properties. This complicates the web service search and discovery process and gives rise to the need for an effective, automated, quality-driven service retrieval and dynamic selection algorithm that considers only a qualified subset of the services. With our approach, we strive to serve consumers better and achieve effective service selection from the vast number of services available by narrowing the search to include only qualified services: the top k services.

Today, service selection by QoS attributes only considers static attributes or a snapshot of current values, resulting in low-quality, low-accuracy results. To address this challenge, we focused on capturing snapshot measurements of QoS attributes at the time of the query, which necessitates considering dynamic attributes, effectively selecting services and confronting the challenges posed by fluctuating and missing attributes and the task of monitoring and storing these values. After considering these aspects, we were able to derive a more effective algorithm without relying on stale dynamic attributes. Our histogram-based approach addresses these challenges, which hamper other approaches. Our methods and approach both simplify and enhance the web service search and discovery process. This enhancement is accentuated by the ease with which our approach can be integrated with existing technology and systems, both for users and administrators.

Acknowledgments

This work was funded in part by the National Sciences and Engineering Research Council (NSERC) of Canada under the NSERC Strategic Research Network for Smart Applications on Virtual Infrastructures

(SAVI-NETGP 397724-10) and the Collaborative Research and Development program (CRDPJ 320529-04 and CRDPJ 356154-07), IBM Corporation, Intel Corporation, and the University of Victoria (Canada).

Author Contributions

The authors invented the Dynamis algorithm for dynamic service selection, and described it in this paper. In particular, Atousa Pahlevan, Jean-Luc Duprat, Alex Thomo and Hausi Müller designed the algorithm; Atousa Pahlevan and Jean-Luc Duprat did the data wrangling, the Mathematica implementation, the Evaluation and wrote the initial manuscript; Alex Thomo and Hausi Müller reviewed and edited the paper.

Conflicts of Interest

One of the authors (Jean-Luc Duprat) is or has been employed by several of the companies whose stock tickers are analyzed in the paper. This doesn't present a conflict, given that we are using this as proxy data for service selection. Nonetheless we prefer to acknowledge this up front.

References

1. Fenza, G.; Furno, D.; Loia, V. Hybrid Approach for Context-Aware Service Discovery in Healthcare Domain. *J. Comput. Syst. Sci.* **2012**, *78*, 1232–1247.
2. Sousa, M.; Lopes, W.; Madeiro, F.; Alencar, M. Cognitive LF-Ant: A Novel Protocol for Healthcare Wireless Sensor Networks. *Sensors* **2012**, *12*, 10463–10486.
3. Castells, P.; Fernandez, M.; Vallet, D. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE Trans. Knowl. Data Eng. (KDE)* **2007**, *19*, 261–272.
4. Crasso, M.; Zunino, A.; Campo, M. Easy Web Service Discovery: A Query-by-Example Approach. *Sci. Comput. Program.* **2008**, *71*, 144–164.
5. Lee, D.L.; Chuang, H.; Seamons, K.E. Document Ranking and the Vector Space Model. *IEEE Softw.* **1997**, *14*, 67–75.
6. Verma, K.; Sivashanmugam, K.; Sheth, A.; Patil, A.; Oundhakar, S.; Miller, J. METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *J. Inf. Technol. Manag. Spec. Issue Univers. Glob. Integr. (IJITM)* **2005**, *6*, 17–39.
7. Chen, I.; Yang, S.; Zhang, J. Ubiquitous Provision of Context Aware Web Services. In Proceedings of the 2006 IEEE International Conference on Services Computing (SCC '06), Chicago, IL, USA, 18–22 September 2006; pp. 60–68.
8. De Almeida, D.; de Souza Baptista, C.; da Silva, E.; Campelo, C.; de Figueiredo, H.; Lacerda, Y. A Context-Aware System Based on Service-Oriented Architecture. In Proceedings of the 20th International Conference on Advanced Information Networking and Applications, 2006 (AINA 2006), Vienna, Austria, 18–20 April 2006; Volume 1.
9. Truong, H.L.; Dustdar, S. A Survey on Context-Aware Web Service Systems. *Int. J. Ad Hoc Ubiquitous Comput. (IJAHUC)* **2009**, *5*, 5–31.

10. Fagin, R.; Lotem, A.; Naor, M. Optimal Aggregation Algorithms for Middleware. In *Proceedings of the 20th ACM Symposium on Principles of Database Systems (PODS)*; ACM: New York, NY, USA, 2001; pp. 102–113.
11. Tian, X.; Zhang, D.; Tao, Y. On Skyline with Flexible Dominance Relation. In Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico, 7–12 April 2008; pp. 1397–1399.
12. Borzonyi, S.; Kossmann, D.; Stocker, K. The Skyline Operator. In Proceedings of the 17th International Conference on Data Engineering (ICDE), Heidelberg, Germany, 2–6 April 2001; Volume 3, pp. 421–430.
13. Ilyas, I.F.; Beskales, G.; Soliman, M.A. A Survey of Top-k Query Processing Techniques in Relational Database Systems. *ACM Comput. Surv.* **2008**, *40*, 11.
14. Skoutas, D.; Sacharidis, D.; Simitsis, A.; Kantere, V.; Sellis, T. Top-k Dominant Web Services under Multi-Criteria Matching. In Proceedings of the 12th International Conference on Extending Database Technology (EDBT), Saint-Petersburg, Russia, 24–26 March 2009; pp. 898–909.
15. Kritikos, K.; Pernici, B.; Plebani, P.; Cappiello, C.; Comuzzi, M.; Benbernou, S.; Brandic, I.; Kertész, A.; Parkin, M.; Carro, M. A Survey on Service Quality Description. *ACM Comput. Surv.* **2013**, *46*, 1:1–1:58.
16. Kritikos, K.; Plexousakis, D. Towards Optimal and Scalable Non-Functional Service Matchmaking Techniques. In Proceedings of the 2012 IEEE 19th International Conference on Web Services (ICWS), Honolulu, HI, USA, 24–29 June 2012; pp. 327–335.
17. Kyriakos, K.; Dimitris, D. Requirements for QoS-based Web Service Description and Discovery. In *IEEE Transactions on Services Computing (TSC)*; IEEE Computer Society: Los Alamitos, CA, USA, 2009; Volume 2, pp. 320–337.
18. El Hadad, J.; Manouvrier, M.; Rukoz, M. TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition. *Trans. Serv. Comput. (TSC)* **2010**, *3*, 73–85.
19. Benouaret, K.; Benslimane, D.; Hadjali, A. Selecting Skyline Web Services from Uncertain QoS. In Proceedings of the 9th International Conference on Services Computing (SCC), Honolulu, HI, USA, 24–29 June 2012; pp. 523–530.
20. Leitner, P.; Rosenberg, F.; Michlmayr, A.; Dustdar, S. Towards a Flexible Mediation Framework for Dynamic Service Invocations. In Proceedings of the 6th IEEE European Conference on Web Services (ECOWS), Dublin, Ireland, 12 November 2008; pp. 45–59.
21. Michlmayr, A.; Rosenberg, F.; Leitner, P.; Dustdar, S. End-to-End Support for QoS-Aware Service Selection, Binding, and Mediation in VRESCo. *IEEE Trans. Serv. Comput. (TSC)* **2010**, *3*, 193–205.
22. Vu, L.H.; Aberer, K. Towards Probabilistic Estimation of Quality of Online Services. In Proceedings of the 2009 IEEE International Conference on Web Services (ICWS), Los Angeles, CA, USA, 6–10 July 2009; pp. 99–106.
23. Doshi, P.; Paradesi, S.; Swaika, S. Integrating Behavioral Trust in Web Service Compositions. In Proceedings of the 2009 IEEE International Conference on Web Services (ICWS), Los Angeles, CA, USA, 6–10 July 2009; pp. 453–460.

24. Pahlevan, A.; Müller, H.A.; Cheng, M. A Dynamic Framework for Quality Web Service Discovery. In *Proceedings of the 4th International Workshop on a Research Agenda for Maintenance and Evolution of Service-Oriented Systems (MESOA)*; SEI Press: Pittsburgh, PA, USA, 2010.
25. Pahlevan, A.; Müller, H.A. Static-Discovery Dynamic-Selection (SDDS) Approach to Web Service Discovery. In Proceedings of the 3rd International Workshop on a Research Agenda for Maintenance and Evolution of Service-Oriented Systems (MESOA), Edmonton, BC, Canada, 21 September 2009; pp. 769–772.
26. Pahlevan, A.; Müller, H.A. Self-Adaptive Management of Web Service Discovery. In Proceedings of the Doctoral Symposium 8th IEEE European Conference on Web Services (ECOWS), Ayia Napa, Cyprus, 1–3 December 2010; pp. 21–24.
27. Pahlevan, A.; Chester, S.; Thomo, A.; Müller, H.A. On Supporting Dynamic Web Service Selection with Histogramming. In Proceedings of the International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-based Systems (MESOCA), Williamsburg, VA, USA, 26–26 September 2011; pp. 1–8.
28. Pahlevan, A. Dynamic Web Service Discovery. Ph.D. Thesis, Department of Computer Science, University of Victoria, Victoria, BC, Canada, January 2013.
29. Huang, J.; Ding, D.; Wang, G.; Xin, J. Tuning the Cardinality of Skyline. In *Advanced Web and Network Technologies and Applications*; Ishikawa, Y., He, J., Eds.; Springer-Verlag: Berlin, Germany, 2008; pp. 220–231.
30. Tsetsos, V.; Anagnostopoulos, C.; Hadjiefthymiades, S. On the Evaluation of Semantic Web Service Matchmaking Systems. In Proceedings of the 4th European Conference on Web Services (ECOWS), Zurich, Switzerland, 4–6 December 2006; pp. 255–264.
31. Lee, D.; Kwon, J.; Yang, S.; Lee, S. Improvement of the Recall and the Precision for Semantic Web Services Search. In Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS), Melbourne, QLD, Australia, 11–13 July 2007; pp. 763–768.
32. Hao, Y.; Zhang, Y. Web Services Discovery based on Schema Matching. In *Proceedings of the 30th Australasian Computer Science Conference (ACSC)*; Australian Computer Society, Inc.: Sydney, Australia, 2007; Volume 62, pp. 107–113.
33. Segev, A.; Toch, E. Context-based Matching and Ranking of Web Services for Composition. *IEEE Trans. Serv. Comput. (TSC)* **2009**, *2*, 210–222.
34. Wang, Y.; Stroulia, E. Flexible Interface Matching for Web-Service Discovery. In Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE), Rome, Italy, 13 December 2003; pp. 147–156.
35. Stroulia, E.; Wang, Y. Structural and Semantic Matching for Assessing Web-Service Similarity. *Int. J. Coop. Inf. Syst. (IJCIS)* **2005**, *14*, 407–437.
36. Saracevic, T. Effects of Inc.onsistent Relevance Judgments on Information Retrieval Test Results: A Historical Perspective. *Libr. Trends* **2008**, *56*, 763–783.
37. Küster, U.; König-Ries, B. Relevance Judgments for Web Services Retrieval—A Methodology and Test Collection for SWS Discovery Evaluation. In Proceedings of the 7th IEEE European Conference on Web Services (ECOWS), Eindhoven, The Netherlands, 9–11 November 2009; pp. 17–26.

38. Küster, U.; König-Ries, B. Evaluating Semantic Web Service Matchmaking Effectiveness Based on Graded Relevance. In Proceedings of the 2nd International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web (SMR) at the 7th International Semantic Web Conference (ISWC), Congress Center, Karlsruhe, Germany, 26–30 October 2008.
39. Tsetsos, V.; Anagnostopoulos, C.; Hadjiefthymiades, S. On the Evaluation of Semantic Web Service Matchmaking Systems. In Proceedings of the 4th European Conference on Web Services (ECOWS), Zurich, Switzerland, 4–6 December 2006; pp. 255–264.
40. Järvelin, K.; Kekäläinen, J. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst. (TOIS)* **2002**, *20*, 422–446.