Long-term Optimization of Update Frequencies for Decaying Information

Simon Razniewski
Free University of Bozen-Bolzano
razniewski@inf.unibz.it

Werner Nutt
Free University of Bozen-Bolzano
nutt@inf.unibz.it

ABSTRACT

Many kinds of information, such as addresses, crawls of webpages, or academic affiliations, are prone to becoming outdated over time. Therefore, in some applications, updates are performed periodically in order to keep the correctness and usefulness of such information high. As refreshing information usually has a cost, e.g. computation time, network bandwidth or human work time, a problem is to find the right update frequency depending on the benefit gained from the information and on the speed with which the information is expected to get outdated.

This is especially important since often entities exhibit a different speed of getting outdated, as, e.g., addresses of students change more frequently than addresses of pensionists, or news portals change more frequently than personal homepages. Thus, there is no uniform best update frequency for all entities.

Previous work [5] on data freshness has focused on the question of how to best distribute a fixed budget for updates among various entities, which is of interest in the *short-term*, when resources are fixed and cannot be adjusted.

In the *long-term*, many businesses are able to adjust their resources in order to optimize their gain. Then, the problem is not one of distributing a fixed number of updates but one of determining the frequency of updates that optimizes the overall gain from the information.

In this paper, we investigate how the optimal update frequency for decaying information can be determined. We show that the optimal update frequency is independent for each entity, and how simple iteration can be used to find the optimal update frequency. An implementation of our solution for exponential decay is available online.

1. INTRODUCTION

In many applications such as address management or website crawling, information gets outdated over time and periodical refreshes are needed in order to ensure that the information remains useful. Refreshing information usually has a cost, e.g., computation time, network bandwidth or human work time. For instance, a company doing web indexing wants to revisit websites neither too seldom, as this leads to the stored information being outdated and thus to unsatisfied customers, nor too often, as this leads to too high computation and networking costs. The same holds for an advertising company that has a database of addresses: If the addresses are updated too seldom, much advertisement will not reach its destination. If they are updated too often, the cost of the updates will outweigh the revenue from the advertisement.

Therefore, it is important to neither refresh information too often nor too seldom, but to find the optimal update frequency that maximizes the ratio between information usefulness and update expenses. We identify two particular problems.

Short-term Optimization. The first problem, which we call the *short-term optimization problem*, is, how to distribute the available update resources best, that is, how to optimize the usefulness (weighted freshness) of the information given a fixed amount of update resources. In its core, it is a question of resource distribution ("Should I better use an update for website A or website B?").

While in the short-term, companies have to deal with the available resources, in the long term, resources often can be adjusted to needs. This holds especially for computational resources given today's cloud technologies. A problem is therefore to determine the optimal number of resources to use.

Long-term Optimization. As updates usually have a cost associated, while correct information brings a benefit, the problem of *long-term optimization* asks for the update frequency that maximizes the net income, that is, the benefit from correct information minus the cost for the updates. In its core it is a problem of optimization ("How often should I update website A?")

In this problem, no interaction between entities occurs, as the optimal update frequency of an entity only depends on its decay behaviour, the cost of an update, and the benefit of being correct. The resources that the company should ideally have available are then computed as the sum of the resources needed for achieving the optimal update frequency for each individual entity.

Example. Let us make the distinction concrete. Consider a web crawling company that indexes two webpages, A and B. Suppose that A and B have a linear probability of getting outdated within 0 and 10, and within 0 and 20 days, respectively (this implies e.g. that after 5 days, the chance for A to still be correct is 50% and for B 75%, and after 8 days, 20% and 60%, and so on).

Regarding the problem of short-term optimization, suppose that the company has resources to perform 3 updates per 10 days. Then the best distribution of these updates is to update A twice (every 5 days), and B once (every 10 days), because this implies that each of the pages has an average probability of 0.75 to be up-to date, which

WebDB'15, May 31 - June 04 2015, Melbourne, VIC, Australia.

ACM 978-1-4503-3627-7/15/05 $15.00.
http://dx.doi.org/10.1145/2767109.2767113.

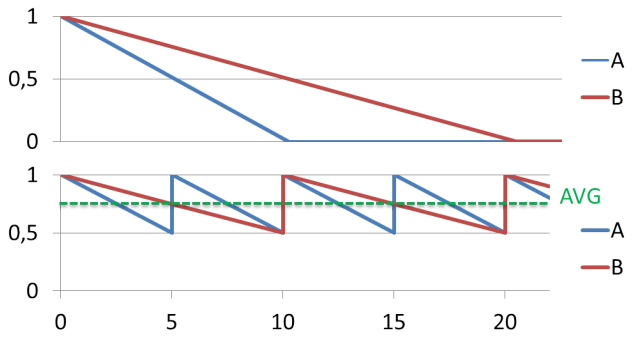

Figure 1: Correctness probability as a function of time, upper graph without updates, lower graph with updates for A every 5 days, for B every 10 days.

also gives the best average for both pages (0.75). The probability of correctness as function of time of each entity is shown in Figure 1. If e.g. we would invert the frequencies and update A only once but B twice, this would give the average correctness of 0.5 and 0.875, respectively, and thus a lower average of 0.6875.

On the other hand, suppose that each update has a cost of 1 unit, and that the daily gain from a correct entity is 1 unit, too. Then we can ask for the optimal update resource allocation for A and B, that is, the number of updates that maximizes the gain minus the update costs. It turns out that for A, the optimal update frequency is every 4.5 days (2.2 updates per 10 days), which yield an average correctness of 0.78, an update cost of 2.2, a gain of 7.8 and a net income of $7.8 - 2.2 = 5.6$ per 10 days. Making available more resources, e.g. 3 updates per 10 days, leads to a higher average correctness of 0.83, and thus also to a higher gain of 8.3, however, the higher update cost outweighs this, leading to an overall income of $8.3 - 3 = 5.3$. Similarly, fewer resources, e.g., 2 updates per 10 days, are not optimal, as this leads to a net income of only 5.5.

Using the same analysis, we find that the optimal update frequency for B is every 6.3 days (1.6 updates per 10 days), which leads to an average correctness of 0.83 and a net income of 6.8, compared with a net income of 6.5 when updating once every 10 days. We conclude that in the long term, in order to maximize its income, the company should adjust their resources to be able to do $1.6 + 2.2 = 3.8$ updates instead of 3 per 10 days, as this improves the net income by 3.3% from 12 to 12.4.

Previous work by Cho and Garcia-Molina [5] has provided a solution to the problem of short-term optimization by reducing it to a problem of linear optimization.

We believe that the problem of long-term optimization has similar relevance, as, in order to stay competitive, companies in the service sector are usually quick in adapting to the optimal market needs, and because this holds in particular for computational resources due to the scalable cloud resources available today.

Contribution. Our contribution in this paper is to give a *general solution to the problem of long-term optimization of update frequencies* for entities when there is a cost associated to an update and a benefit associated to up-to-date information.

Our solution is independent of the chosen decay function and applies to any domain where the cost of checking whether an entity got updated is the same as the cost of pulling the update.

We illustrate the benefit of our solution in two use cases, namely address data maintenance [16] and web crawling [5].

2. MOTIVATING SCENARIOS

We next discuss two concrete scenarios where short-term and long-term optimization play a role.

Medical Advertisement. Consider an address reselling company in the medical domain. Its business model consists of maintaining a high-quality database of specialist doctors and selling the contact information to suppliers of medical technology and to pharmaceutical companies. As medical equipment can be expensive, suppliers are willing to pay considerable amounts for addresses of specialists that they intend to target with advertisement.

The main activity of the address reselling company is therefore the acquisition of information about new specialists, and the maintenance of existing information. For both tasks, it uses mainly two techniques, namely web search on hospital homepages, on doctor homepages and in social networks, and phone calls to hospitals.

Information about doctors shows different decay rates. Younger doctors are more likely to move than older doctors, and similar differences exist also between doctors in the countryside versus doctors in big cities.

If the company has a fixed set of employees, its problem is one of short-term optimization. Given fixed resources, it has to decide in which order to check/update the information about each doctor. In the long-term, the company is however interested in adjusting its resources in such a way that it optimizes its net income. Thus, the company also faces the problem of long-term optimization, where it has to decide for each record the update frequency that optimizes the benefit from the achieved currency minus the update cost to achieve that currency.

Web Crawling. In web crawling, companies are providing large-scale service based on web information. Due to its large scale, for such services, the web cannot be visited live when serving user requests, but instead crawled versions of the web are used. Example services are search, price comparison sites or news aggregators. A commonality is that the quality of the provided service is highly dependent on the crawling frequency.

A current estimate for the cost of a crawl is 0.0002 Cents, based on the pricing of $2 per 1 Million crawls of the crawl provider 80legs[1][2].

Webpages may exhibit very different frequencies with which they get outdated. While e.g. newspages can change minutely, personal homepages often change even less frequently than once per month. From past crawls, website attributes and website content, crawlers can well estimate the likelihood of change of a given webpage. Given this, they have to decide how often to recrawl a given webpage.

If their resources are fixed (i.e., they have a fixed number of servers and no or flat network fees), the problem is one of short-term optimization. They have to determine how to best distribute their available crawls among webpages. This problem has been extensively investigated (see e.g. [5, 14, 17]).

If their resources are not fixed however, e.g. if they are able to use more computing capacity from cloud providers, are able to buy more servers for future use, or are charged depending on their network usage, the problem becomes one of long-term optimization.

[1]http://80legs.com

[2]For historic values, see e.g. http://research.microsoft.com/en-us/um/people/nickcr/pubs/craswell_adc04.pdf, which reported 0.05 ct/crawl (network cost) in 2004, and http://www.michaelnielsen.org/ddi/how-to-crawl-a-quarter-billion-webpages-in-40-hours/, which reported 0.0023 ct/crawl in 2012.

They have to decide on the best update frequency for each website, that is, the frequency that maximizes the net income from the freshness of that site minus the update costs. The individual update frequencies for the webpages then constitute the resources that the company should acquire for future use.

3. RELATED WORK

In many domains, information is subject to change over time. Currency (or freshness) as a dimension of data quality is used to describe the extent to which information captures the current state of the modelled domain. Analyses of currency can be found e.g. in [10, 7] and [2], with the latter also investigating how to measure currency.

Cho and Garcia-Molina [5] investigated the problem of short-term optimization for information subject to decay. They show that an optimal policy for update distribution can significantly outperform random, uniform and proportional update distribution, and described a methodology to compute the optimal update distribution. There has been considerable follow-up work regarding optimal crawling strategies under various constraints, e.g. [14], or [17]. Other work on crawling has focused on how to determine the change frequency of web pages [6].

Similar problems with decaying information occur also in data delivery on the web, as discussed in [12, 3, 10], for stream data warehouses [9], or for distributed databases [4].

An overview of decay functions can be found in [10], where the authors discuss linear, exponential, geometric, Weibull, and Gamma distributions for describing decay rates, and describe a methodology to measure the currency of information.

Information decay is also known in other domains such as recommender systems [11], where it is called drift in user interest, and describes the observation that the longer ago a user was found to have an interest in a certain record, the higher the chance that that interest may have changed.

Decay is also known in entity matching and data cleaning [13, 8], where newer attribute correspondences are better indicators for the equivalence between entities than older correspondences.

Finally, decay is also a frequent issue for storage media. There decay, often called data degradation or data rot, refers to the physical processes that make data unreadable over time [15, 1]. Since different storage media show different decay rates, our analysis may also be relevant for refresh policies for degrading storage media.

4. INFORMATION DECAY

Decay is a well-known concept in physics and chemistry, for instance in radioactive decay or chemical reactions, where it describes the quantitative degradation of substances over time. In IT, data itself, when properly stored, is not subject to decay, however, as the real-world changes, the information value of the stored data may degrade. We therefore label this phenomenon as *information decay*.

Decay Function. To describe decay mathematically, one needs a function over time which describes the probability that an entity is up-to-date. We call this function the *decay function* z. In general, any function z with $z(0) = 1$ and which is monotonically decreasing can occur as decay function, which describes the following:

$$P(\text{correctness at time } t) = z(t).$$

Various classes of decay functions such as linear, exponential and geometric decay are discussed in [10]. Linear decay, which

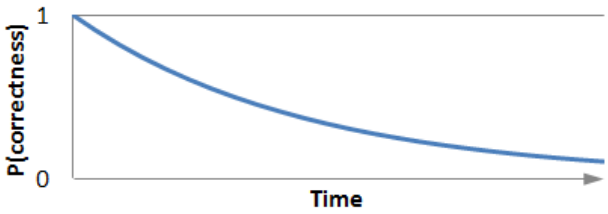

Figure 2: Probability of correctness as a function of time for information under exponential decay.

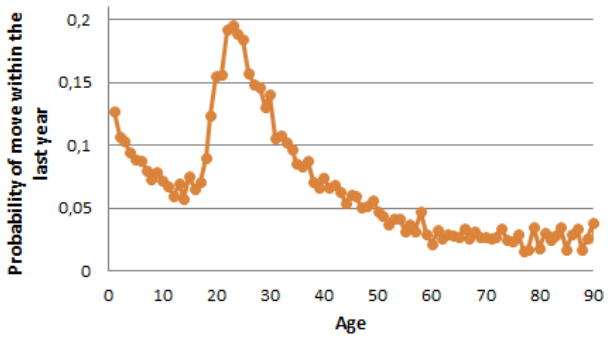

Figure 3: Illustration of varying decay rates. The probability of a person in the taxpayer dataset [16] to move varies with the person's age.

is mathematically easy to describe, was already used in the introductory example. In [5] it was shown that the update behaviour of webpages follows an exponential behaviour. We therefore in the following focus on linear and exponential decay.

Both classes of functions use a *decay rate* λ as parameter. The decay rate describes the velocity with which entities get outdated.

The probability of correctness for an entity under linear decay is

$$z_{lin}(t) = max(1 - \lambda_{lin} t, 0).$$

Linear decay can also be characterized by the maximal lifetime $t_{\max}$ of an entity, which is calculated as $t_{\max} = \frac{1}{\lambda_{lin}}$.

Under exponential decay, the probability of correctness for an entity is

$$z_{exp}(t) = e^{-\lambda_{exp} t}.$$

The shape of $z_{exp}(t)$ is shown in Fig. 2. Often exponential decay is described in terms of the so-called *half-time* $t_{\frac{1}{2}}$. The half-time is the time after which the probability of an entity to still be valid is 0.5. Given λ_{exp}, the half-time $t_{\frac{1}{2}}$ is calculated as $\frac{ln(2)}{\lambda_{exp}}$. Furthermore, the mean lifetime of an entity can be computed as $\frac{1}{\lambda_{exp}}$.

Example 1 (Decay Coefficients) *As we do not have data about doctors, we use here the UCI dataset about Californian taxpayers [16] to illustrate varying decay rates. The tax payer dataset contains 200k records of Californian residents, and contains 42 attributes describing socioeconomic features of the taxpayers. In particular, one of the attributes is called* "lived in this house 1 year ago"*, and describes whether the person changed residence within the last year. We aggregated this attribute by person age, obtaining moving probabilities per age as shown in Fig. 3. It is interesting to see that newborns are more likely to move than older children, probably because their parents are likely to move to places*

Age	P(Move within last year)	P(No move within last year)	λ_{lin}	λ_{exp}	$t_{\frac{1}{2}}$ (years)
25	36.7%	63.3%	0.37	0.457	1.52
30	27.1%	72.9%	0.27	0.316	2.19
40	15.0%	85.0%	0.15	0.163	4.25
50	9.3%	90.7%	0.09	0.098	7.07

Table 1: Decay rates λ for linear decay, for exponential decay, and half-time for exponential decay for persons of age 25, 30, 40 and 50 in the UCI dataset.

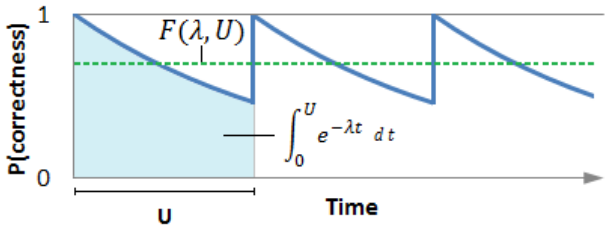

Figure 4: Calculating the average freshness.

accommodating the bigger family. Less surprising is that the moving probability peaks at age 25, before it continuously decreases.

While we do not know the true decay function for residences, if we assume that it follows linear or exponential decay, we can compute the corresponding decay coefficients from the moving probability within one year. The results for persons of age 25, 30, 40 and 50 are shown in Table 1.

Since without interventions, the probability of correctness would steadily decrease towards zero, updates may be used to obtain fresh information. The abstract *update operation* that we consider here retrieves the current correct value for a given entity, thus yielding again a correctness probability of 100% for the updated entity. Subsequently, the updated information is subject to decay and the correctness probability decreases again over time.

Repeated updates of an entity could in principle be executed using arbitrary patterns, however, if the decay rate is constant, a uniform distribution of updates leads to a better average correctness. We therefore next investigate the impact of a uniform update frequency U onto the average freshness of the entity.

Relation of Update Frequency and Average Freshness. Given a decay function z, a decay rate λ and an update frequency U, the average correctness $F(\lambda,U)$ of any entity with that decay rate under this update frequency corresponds to the area under the curve z from zero to U divided by the update frequency U (see Fig. 4 for an illustration):

$$F(\lambda,U) = \frac{\int_0^U z(t)dt}{U}.$$

For linear decay with $U \leq \lambda$, this becomes

$$F_{lin}(\lambda,U) = 1 - \frac{\lambda U}{2}. \quad (1)$$

For exponential decay, this becomes

$$F_{exp}(\lambda,U) = \frac{e^{\lambda U} - 1}{\lambda U e^{\lambda U}}. \quad (2)$$

Age	λ	Update frequency U (in years)					
		0.25	0.5	1	2	5	10
25	0.457	94%	89%	80%	66%	39%	22%
30	0.316	96%	93%	86%	74%	50%	30%
40	0.163	98%	96%	92%	85%	68%	49%
50	0.098	99%	98%	95%	91%	79%	64%

Table 2: Average correctness of information for persons of age 25, 30, 40 and 50 in the UCI dataset for various update frequencies, assuming exponential decay.

Example 2 (Updating Addresses) *In Table 2, we show the average freshness of address information of persons of various age depending on the update frequency, assuming exponential decay. As we can see, an update frequency of once a year for 25-year olds yields nearly the same average freshness as an update frequency of once every five years for 50-year olds, and an update frequency of once every two years for the former nearly the same average freshness as a frequency of once every ten years for the latter.*

It is clear that higher update frequencies increase the average freshness. However, from a certain update frequency on, it may be questionable whether the marginal increase of the average freshness obtained by further updates is worth its costs. Therefore, we investigate in the next section the question of what the optimal update frequency is and how it can be computed.

5. COMPUTING THE OPTIMAL UPDATE FREQUENCY

To determine the optimal update frequency, we have to fix a goal. The goal to optimize in our case is the net income that can be derived from an entity. The *net income* from an entity depends on two factors: The expenses spent for updating the entity, and the benefit derived from the average freshness of the entity. To determine both values, we need to know the values of the following two constants:

1. Update cost (C): The update cost describes the cost of an update operation. Depending on the application, this cost may contain e.g. human work time, computational resources or network utilization.

2. Benefit per time from up-to-date entities (B): The benefit describes the economic value per time unit that is derived from a correct entity. It may e.g. describe the yearly benefit derived from being able to send advertisement to a person, or the daily value of good answers to queries of services relying on crawled data.

Net Income Calculation. Given values for B, C and λ, we can compute the net income $NI(U)$ wrt. an update frequency U as:

$$NI(U) = B \cdot F(\lambda,U) - \frac{C}{U}. \quad (3)$$

Without loss of generality, we assume in the following that the update cost C is always 1, therefore, the benefit B expresses multiples of the update cost. Then, the net income under linear decay with $U \leq \lambda$, can be calculated by plugging Eq. 1 into Eq. 3 as:

$$NI_{lin}(U) = B - \frac{B \cdot \lambda \cdot U}{2} - \frac{1}{U}. \quad (4)$$

Note that the best update frequency U is always either $\leq \lambda$, or infinity. The latter is a special case which occurs when entities change too fast, such that the benefit of any update frequency is too small to outweigh the costs. This case is treated below ("futile entities"), here we can therefore safely assume that $U \leq \lambda$.

Under exponential decay we similarly calculate $NI(U)$ by plugging Eq. 2 into Eq. 3 as:

$$NI_{exp}(U) = B\frac{e^{\lambda U}-1}{\lambda U e^{\lambda U}} - \frac{1}{U}. \quad (5)$$

Optimal Update Frequency. To find the optimal value for $NI_{lin}(U)$, by common calculus we take the derivative of 4 and set it to zero. For linear decay, the derivative is:

$$NI'_{lin}(U) = -\frac{B\lambda}{2} + \frac{1}{U^2}.$$

The maximum of $NI_{lin}(U)$ is therefore at $U = \sqrt{\frac{2}{B\lambda}}$.

For exponential decay, the derivative of Eq. 5 is:

$$NI'_{exp}(U) = \frac{e^{-\lambda U}(-Be^{\lambda U} + B\lambda U + \lambda e^{\lambda U})}{\lambda U^2}$$

We find that $NI'_{exp}(U) = 0$ at the following position:

$$U = \frac{-W(\frac{\lambda - B}{Be}) - 1}{\lambda}, \quad (6)$$

where W is the Lambert W or product log function.[3] While there is no symbolic way to find U, the value can be found with iterative methods (we used bisection).

Example 3 (Optimal Update Frequencies) *Consider again the person groups of various age as shown in Table 1. In Fig. 5, we show the yearly net income for various person groups depending on the update frequency. Yearly benefit B and update cost C are both 1. As we can see, the net income is negative for very high update frequencies, reaches a positive maximum at roughly 3 to 5 years, and then drops gradually. We also see that the maximal net income for the 50-year olds (~0.6) is much higher than the maximal net income for the 25-year olds (~0.2). Furthermore, we see that the position of the maximum for the former is at around 5 years, while for the latter it is at around 3 years. Below we report the numeric values of the maxima:*

Group	*Optimal update frequency U (in years)*	*Maximal yearly net income NI*
25-year olds	*3.38*	*0.21*
30-year olds	*3.61*	*0.32*
40-year olds	*4.42*	*0.49*
50-year olds	*5.36*	*0.59*

If we take the average of the yearly update frequencies reported above, we would update all entities uniformly every 4.06 years, yielding an average gain of 0.3973. Using the optimal update frequencies for each group, the average gain becomes 0.4026, which is an increase of 1.3% over the uniform update frequency.

We can make two observations. First, the maximal possible gain for entities whose freshness is decaying slower can be considerably higher than that for entities decaying faster. Second, in order to achieve the maximal possible gain over all entities, the update frequency should be adjusted separately for each group of items of different decay rate.

[3] http://en.wikipedia.org/wiki/Lambert_W_function

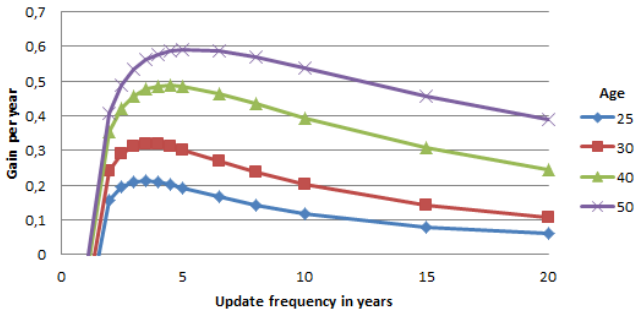

Figure 5: Net income NI per year for various person groups wrt. different update frequencies. Yearly benefit B and update cost C are both 1.

Futile Entities. Entities may get outdated so fast that it is better to not update them at all. For linear decay, the maximal benefit that can be achieved by one update is the area under the decay curve $z(t) = max(1 - \lambda_{lin} t, 0)$, which reaches zero at $t = \frac{1}{\lambda}$ and thus covers an area of $\frac{1}{2\lambda}$. If the benefit $\frac{B}{2\lambda}$ is outweighed by the update cost C, that is, if $B \leq 2\lambda C$, then updating the entity does not make sense as the benefit from the update is outweighed by the update cost. Similarly, for exponential decay, the maximal benefit is the area under $z(t) = e^{-\lambda t}$, which is $\frac{1}{\lambda}$, and thus, whenever the values of B and C are such that $\frac{B}{\lambda} \leq C$, the benefit of an update is outweighed by the update cost. Note that futile entities only appear in applications where outdated entities have zero value, as it is the case for addresses, but possibly not for crawls of webpages.

Example 4 (Futile Entities) *In our example above, where B and C are fixed to 1, entities under linear decay with a decay rate ≥ 0.5 or under exponential decay with a decay rate ≥ 1 satisfy the above condition and thus it is more efficient to not update them at all. Such entities would have a maximum lifespan less or equal to 2 years, or a half-time less or equal than $ln(2) \approx 0.69$ years, respectively. This could e.g. be trainee doctors or doctors in military hospitals that frequently relocate.*

Use Case: Webpages

In this section we discuss another use case, which is about the frequency with which a crawler recrawls webpages.

In [5], Cho and Molina did measurements on the change frequency of 270 popular websites. Subsequently, they created a model where they grouped the sites into five categories, for which they assumed that 23% were updated on average daily, 15% weekly, 16% monthly, 16% quarterly and 30% yearly. They then assumed to have resources to update the crawled version of every web page once a month, and discussed how to best distribute these 270 updates over the whole set.

Expanding our analysis from above to the same set consisting of five groups of entities, we find that spending 270 updates per month on the set of 270 sites is the best choice only if the ratio between benefit per up-to-date entity per day and update cost ($\frac{B}{C}$) is 0.355. If the cost-benefit ratio is different, providing resources for 270 updates per month leads to a benefit that is significantly below the optimum, as shown below:

Benefit-cost ratio $\frac{B}{C}$	Optimal number of monthly updates	Loss wrt. optimum when using 270 updates
0.1	60	18.5%
0.3	242	0.1%
0.355	*270*	-
0.5	333	0.5%
1	1,033	6.0%
10	4,710	25.8%
100	15,882	31.8%

In the third column, we show the loss resulting from using 270 updates instead of the optimal number of updates (Column 2) for varying benefit-cost ratios (Column 1). For the first two rows, this 270 updates would imply that entities are updated too often than optimal, for the last four rows it would imply that entities are updated less often than optimal. Note that the loss for higher $\frac{B}{C}$ ratios grows less fast, because the net income depends on the average freshness, which is bounded by 100%. Since 270 updates already achieve a reasonable average freshness (>60%), significantly higher update resources cannot outperform the net income similarly.

As we can see, depending on the true value of $\frac{B}{C}$, a fixed provision of updates which is different from the optimum may lead to significant losses in the net income, even though we assume these updates to be distributed in the optimal way according to short-term optimization [5]. For instance, if the true ratio is 100, using resources for monthly updates (ratio 0.355) leads to a reduction of the overall income by 31.8%. Note also that the ratio for search engines may be much higher, as in the next section we compute an estimated $\frac{B}{C}$ ratio of 16k for Google.

6. DISCUSSION

Computing the optimal update frequency under exponential decay (Eq. 6) requires iteration, which e.g. cannot be done in standard Excel. We therefore provide a Java implementation that computes the location of the optimum. It can be downloaded at

`http://www.inf.unibz.it/~srazniewski/updatefrequency.jar`.

Executed with values for the two parameters r and $\frac{B}{C}$, it returns the optimal update frequency under exponential decay and the obtained net income.

Finding the Correct Decay Function. Finding the correct function class and the right parameters requires either domain knowledge, or data that can be statistically analysed. The work in [10] shows that in different domains, different decay functions may be applicable. In [5], it was verified that Poisson processes for data change lead to exponential decay curves. Regarding the finding of the right function parameters, note that in our example of the tax payers, we only used the age attribute. It is likely that other attributes will help to identify variances in decay rates too, e.g. it is likely that certain professions require higher mobility than others, or that persons with bigger families move less often than singles. Regarding the update frequency of webpages, an extensive discussion on how to estimate it can be found in [6].

Sensitivity to Wrong Estimates of Decay Rates. Even if the class of decay functions is known, the individual decay rates can be varying. In Fig. 5 we can see that the optimal update frequencies for different decay rates are relatively close to each other, which is likely due to the fact that the half times of the various age groups are still relatively similar. For the web page scenario discussed in [5], the half times show a much higher variance (1-365 days), so there a wrong estimate for the decay rate might have much more severe impact. Given the shape of the net income curves in Fig. 5, it is generally advisable to be conservative in doubt. It is better to underestimate the decay rate, and thus, to perform updates less frequent than optimal, than to overestimate the decay rate, and perform updates too frequent.

Applicability. In general, the presented analysis can be used in any domain in which information is changing, and in which the *cost of checking for changes is close to the cost of pulling the changes.* An example where this is generally not the case is caching: It is usually much cheaper to detect whether a change happened, e.g. using flags or hash codes, than pulling a change. For web crawlers, it is plausible that the benefit from outdated information is not necessarily zero, but instead decreases gradually with the number of new versions. It is not clear how that can be taken into account in this model. For addresses this problem does not occur, as an address is indeed either correct and useful, or outdated and useless.

Search Engine Business. We can instantiate our framework with known values of the Google search engine. Note that this is just a hypothetical example, the techniques Google uses for determining recrawl frequencies are not public and may be very different. The equation in Eq. 6 has four parameters (B, U,λ and C). Thus, if we fix three of them we can calculate the fourth

We have seen that the cost C for one crawl is in the order of 0.0002 Cents. It is reported that the website quietnightbeds *(http://quietnightbeds.co.uk/)*, an online store, gets crawled by Google 75 times a day[4] (this gives U). If we assume that this webpage changes on average daily (λ), we can compute the value 16,000 for the ratio $\frac{B}{C}$. Thus, given the known value for U, we can deduce that the crawling service (Google) obtains a benefit B of around 3.2 ct per day from having the current version of this webpage. Analogously, if the webpage would change twice a day, the benefit would be $\simeq$1.6 ct per day.

7. CONCLUSION

In this paper we have introduced the problem of long-term optimization of the number of resources used for refreshing decaying information. We have discussed two use cases (postal advertisement and web crawling), and presented a general way to calculate the optimal update frequency, along with concrete instantiations for linear and exponential decay. We have shown that when adjusting their resources based on the optimal values found using long-term optimization, businesses can potentially increase their net income considerably.

Future work will focus on finding the best update frequency when parameters are not exactly known, and on the impact of life-cycle changes of decay parameters.

Acknowledgment. We thank Anastasia Mochalova for discussions that started this research, and Divesh Srivastava for pointing out connections to web crawling. This work was supported by the research project MAGIC, funded by the province of Bozen-Bolzano.

[4] *http://www.sitepoint.com/increase-search-traffic-getting-site-recrawled-often/*

8. REFERENCES

[1] M. Baker, K. Keeton, and S. Martin. Why traditional storage systems don't help us save stuff forever. In *Proc. 1st IEEE Workshop on Hot Topics in System Dependability*, pages 2005–120. Citeseer, 2005.

[2] M. Bouzeghoub. A framework for analysis of data freshness. In *Proceedings of the 2004 international workshop on Information quality in information systems*, pages 59–67. ACM, 2004.

[3] L. Bright and L. Raschid. Using latency-recency profiles for data delivery on the web. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 550–561. VLDB Endowment, 2002.

[4] D. Carney, S. Lee, and S. Zdonik. Scalable application-aware data freshening. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 481–492. IEEE, 2003.

[5] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Trans. Database Syst.*, 28(4):390–426, Dec. 2003.

[6] J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology (TOIT)*, 3(3):256–290, 2003.

[7] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. *ACM Trans. Database Syst.*, 37(4):25, 2012.

[8] F. Geerts, G. Mecca, P. Papotti, and D. Santoro. The LLUNATIC data-cleaning framework. *PVLDB*, 6(9):625–636, 2013.

[9] L. Golab, T. Johnson, and V. Shkapenyuk. Scalable scheduling of updates in streaming data warehouses. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):1092–1105, 2012.

[10] B. Heinrich, M. Klier, and M. Kaiser. A procedure to develop metrics for currency and its application in crm. *Journal of Data and Information Quality (JDIQ)*, 1(1):5, 2009.

[11] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[12] A. Labrinidis and N. Roussopoulos. Balancing performance and data freshness in web database servers. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 393–404. VLDB Endowment, 2003.

[13] P. Li, X. L. Dong, A. Maurino, and D. Srivastava. Linking temporal records. *PVLDB*, 4(11):956–967, 2011.

[14] J. McKeeth. Method and system for updating a search engine, July 13 2004.

[15] T. Schwarz, M. Baker, S. Bassi, B. Baumgart, W. Flagg, C. van Ingen, K. Joste, M. Manasse, and M. Shah. Disk failure investigations at the internet archive. In *Work-in-Progess session, NASA/IEEE Conference on Mass Storage Systems and Technologies (MSST2006)*, 2006.

[16] UCI Machine Learning Repository. California taxpayer dataset. `http://archive.ics.uci.edu/ml/datasets/Census-Income+(KDD)`.

[17] J. L. Wolf, M. S. Squillante, P. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *Proceedings of the 11th international conference on World Wide Web*, pages 136–147. ACM, 2002.