WEB INFORMATION MANAGEMENT SIMPLE (FOR REAL)

Gabriel Rovesti

1 Summary

Disclaimer

This file is entirely based on the whole slides I screenshotted by recordings of video lessons of 2021-2022 and all the 2023-2024 course material (I put both the incomplete and complete versions – I'd say complete 95%, but those are represented by the "Complete" suffix on the specific PDF lesson name). They are represented by 25 modules (here, for usefulness sake, I put references to slides' PDF directly in the title).

The only difference in the present course is the presence of ChatGPT/chatbot-based material of discussion, present in this file. This course is actually really fun, like a relax from the other ones and actually interesting at that. The professor is amazing and is a great example of how to teach and bring in educational content to masses; I'd argue it was the only fun course to follow in this semester.

Consider, as other files of mine, this is entirely based also on confrontation with notes of the same kind, both in English and Italian (I'd say at least 5/6 files to be as complete as possible – so, good stuff and lots of work). So, I hope this can be useful. Feel free to reach me to give some feedback over its content. Even to thank me, it doesn't kill me that much.

2 Introduction to Web (Lectures 1-2)

Some course miscellanea:

- Course reference site: http://corsi.math.unipd.it/wim/
- Official course email: wim@math.unipd.it
- A core "swiss-knife" of reference for the examination: http://corsi.math.unipd.it/wim/node/5
- The project info here (a usability analysis of a website): http://corsi.math.unipd.it/wim/node/6

(All this part is just for context and historical reasons – so, go ahead)

A crucial question to consider when evaluating a product, whether it's a website or any technological innovation, is to ask "why" it came into existence and why it succeeded, without being confined to a specific timeframe. It involves critical analysis. As time progresses, things evolve, and so do we. Understanding this evolution is essential, and it's a process of change.

Merely knowing how a technology works is insufficient because technology itself is subject to change. What holds greater significance is understanding the "why" behind it and not so much the "how". This is why we often find value in revisiting the past, looking back to gain insights for the future.

The first example we come across is in 1945 with the *MEMEX,* an analogic calculator with an archiving system created by Vannevar Bush, making possible the *hyperlink* between information, via a primitive "link" and the possibly of view data inside other machines (based on the work *As we may think*, at its core "linking knowledge continually"). We then quote in 1960 the *oN-Line System (NLS)*, with the first computer made by mouse, keyboard and fast writing device, making the first of its kind audience presentation of a technological product, showing it to the audience.

Ted Nelson creates the definition of both "hypertext" and "hypermedia", launching in 1967 the project *Xanadu*, first Web draft where hypertext, with "bivisibility" and "bifollowability" (going back and forth logical links) of strings connecting documents with a versioning system. It was meant to browse interactively via point-and-click interface, with the underlying moral of "being free as a social tool", allowing even micropayments inside for compensating licenses and copyrights: for as much revolutionary it was, it was never really implemented.

2.1 The rise of WWW and Web Browsers

Tim Berners Lee, in 1980, crafts some ideas on how to link the miscellanea of information present in CERN; in 1989 writes a draft of the "information mesh" between media hierarchically linked between each other: the *World Wide Web,* complete with a browser of the same name (before Enquire, then WorldWideWeb) allowing even editing and server side of pages (navigating but also actively contributing).

It was proposed an implementation on the NeXT computer (the best at the time) of Steve Jobs, but in 1991 had its project rejected because it was "too simple" and because it was based on "pressing numbers" (a public server all in text was created the same year to show its power). Here the GUI and hardware were fatal in a time where the social system were not grown enough to understand it.

On the same year, the first search engine in the world, but not in the web, was presented and was called *Archie*, working with FTP. Also, there is *WAIS* (*Wide Area Information Servers*), able to search text within documents spread across different databases on the Internet, which after initially succeeding in 1990-1993, it disappeared in 1995.

Another interesting protocol developed by University of Minnesota is *Gopher*, where pages were accessible having a fixed structure, either a menu or a document (text/image).

In the early 90's, it was successful and other browsers appeared (Viola, Midas, Erwise), creating more pages and web servers (around 26). In 1992, *Veronica* was created as the search engine of Gopher. After success, it declined because it was needed to pay a small fee to create a Gopher server.

People started utilizing WWW instead because it was free, doubling its users in 1993 (servers from 23 to 50 and web traffic is 1% of the world Internet traffic). lot of browsers were created in this period (some names: Viola, Cello, IBM WebExplorer, SlipKnot, Lynx, Mosaic), but who survived are:

- *Lynx* (only-textual browser who exploited a specialized niche and it's still in use today)
- *Mosaic*, the browser rampant in this period, which despite being slower than all the other browsers, became the browser of reference, because it viewed images inline (inside the page and not in a separated link)
    - On slide, many examples of websites circa 1994-1998, which experiment with images.
    - This teaches us to think out of the box and sometimes a good look matters more than the underlying quality, with the right compromises.

In this point in time, the first commercial website was born in 1993 and we reach 200 web servers. WWW covers press, international conferences and we get 2500 servers by 1994. CERN decided to stop funding the project (because it was essentially too big to control); it was then funded by MIT, where W3C was funded. Netscape released in 1994 the *Netscape browser (Navigator).*

The page loading was incremental (*incremental display*), showing something that appears and give some information to the user but also engaging him.

3 Web Usability (Lectures 3-4)

A website can be seen as a home or as a shop, where we watch it from the outside and decide if we want to go in or not. The *homepage* can be seen as the windows to the website, where there is the most important abstract level of information, the important stuff: the user has some needs that we want to satisfy. *It's not a problem of information synthesis but rather in information communication.*

3.1 Information Axes and Timers

A good homepage should be like writing a good journal article on a certain subject. We follow the 6 W, which define the information axes:

- Where
  - Where did I arrive?
- Who
  - Who's behind the website?
- Why
  - What are the benefits? Why should I stay?
- What
  - What choices do I have?
- When
  - What are the last news?
- How
  - How do I arrive to where I want?

One of the fundamental problems in the Web is *time* and *user timers*, so the time users dedicate to a website. Infact:

- users have *limited time* and *expectations* about a product (a must-see the clip of Fantozzi, get some good culture and have a laugh).
- On average, *the time* the user will be willing to stay in our website *homepage* is *31 seconds*.
  - In this time, we must make tricks and calculations to convince users to stay and show him the information components.

How much text can we put inside the homepage?

- A good, cultured adult can read on average 200/300 words per minute of reading (going down to 180 in computer screen reading)
  - If you go above the threshold, you risk losing the user completely
- Up to 93 words per minute is allowed in 31 seconds
  - The number should be much less, given user in that time analyzes layout and a lot of things
  - Consider 93 words are only text, you should also consider *page layout*

User behaviour is *dynamic* and follows a temporal line: we want to make the user *happy now*, but also to *come back* later. Here we think on the axis "*what-when-how*" paradigm (and not "*who-where-why"*, not asked anymore).

- The user coming back is more demanding, having more expectations and less time to spend for us than the times before
    - First visit: 31 seconds
    - Second visit: 25 seconds
    - Third visit: 22 seconds
    - Fourth visit: 19 seconds (approximately 57 words)
    - After fifth visit, timers become more stable

- So, do not put too much text, avoid text overloading the other axes and don't put too little text
    - Examples on this are the QuadGraphics/Dial before you Dig websites: both got redesigns that got rid of much text and also layouts definitely unusable.

After the homepage threshold is passed, no need for the information axes (user knows us) and wants to reach his goal *as fast as possible*: given we built user loyalty

- At this point, we attracted the users to stay inside the site, they are less likely to go away, and the returning user will stay more time (going up from 31 to 53 seconds).
- This time allows us to add information more specific in internal pages to the context than the home page (but not exaggerating, having about 159 words for 53 seconds at our disposal)
    - An example not to follow is a legacy iPod nano website, with so many words one can't even move, and so much visual layout wasted – still a huge problem on many company websites

A thing to consider in design first and foremost: just because it's nice, it doesn't mean it's useful. It's needed to find a balance between compact design and all the information. I'd say it's just a matter of *modularizing*, so it's important to craft "modules" of information to make it digestible to the users.

3.2 Timer types

Are we over with timers? So far, we've talked about *local* timers (attached to *internal* pages), but there are also <u>global timers</u>. This is the time it takes for the user to be satisfied on what he wanted on our site (to satisfy individually the user) and generally it's the maximum time the user has to satisfy his goal

1) The first is the <u>preliminary time</u>, in which the *user has got an idea* of us and *decides to stay*; this is the so-called <u>choice time</u>, which is *1 minute and 49 seconds*
    a. We do have more time for the internal page, but if the user doesn't find what he needs, he may get angry and then leave
    b. This is the most important one. A user getting out before finishing the task has a probability of 88% of having totally lost the user (so we really must pay attention)

2) The second time is the time the user expects to have found what he wanted, and he successfully navigated into the website. This limit is called <u>success time</u>, having about *3 minutes and 49 seconds*
    a. What matters most is the good <u>balancing</u> among *homepage*, *internal pages* and *trail (path)* that the user follows to arrive where he wants

Some quick rules we have:

- Homepage (31 seconds)
- Internal pages (53 seconds)
- Choice time (1m. 49 seconds), so it means the user, after having seen the homepage *and a bit more than one internal page* of our website, makes up his mind and chooses whether to stay or to leave
- Moreover, given the success time (3 m. 49 seconds), he expects to reach the goal after *three pages and a half* of our website

4 Website Structure (Lecture 4)

The web structure becomes critical, we can see that as a tree:

- after *one click* (*max 2*), we must convince the user
- after *other two clicks* (*max 3*), we must give what he wanted

Inside internal pages, we need to consider people do not follow a defined path always starting from the homepage as it happened years ago:

- browsing can start from any possible point inside of a website and we need to make the same considerations about information axes also for internal pages.
- user can appear inside any web page of a website, for example looking for something with a Google search
- technically, this is called deep linking: search engines need to be optimized themselves in order to guarantee a scan as fast as possible. Every page can possibly be the first the user sees.

Let's see the axes in detail: some become mandatory, other ones optional, others can be omitted. In general, we might want to categorize them accordingly for the deep linking case.

Mandatory axes:

- *Who* (typical shortcut: logo in the upper-left corner)
- *What* (typical shortcut: direct link to the homepage)

Completely optional axis:

- *When*

Optional informative axes (but suggested):

- *Why* (short description, even a few slogan words)
- *How* (typical shortcut: the search functionality, preferred position in the up-right part)
    - optionally, related pages

4.1 Don't get lost

At last, but not least: *Where* axis. The user is "thrown" in the middle of the information forest (arrives in a page), so I can show a "minimap" to at least start movement, deciding the neighbors and the near links (so, he can start to look around and browse at his ease).

- To avoid for the user to always go to the home page "wasting" a click (more effort), if the user directly lands on a page, we have more info on what he wants
- To avoid this, it's convenient to give "Where" as additional information in the internal page
    - Use that information to guide him, given its habits and preferences, instead of pushing him back to homepage

Another important thing is having windows or special elements to welcome "teleported" people to the website, to inform them and give them more information.

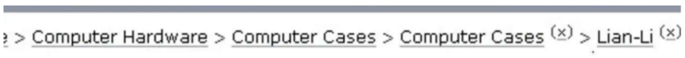
- Something simple like "Welcome Google user" and some links to help him find other information on the inside of a website, like in the *cnet news* slides' example: simple thought yet quite effective in practice

An example "definitely not to follow" is in All Musicals website:

- unwanted popups and ads in the middle of the screen covering the content (something that happens even today in many websites if you don't have UBlock Origin – please use it)

These techniques that make people understand "where they are" are called breadcrumbs, essentially "pieces of bread describing how you arrived and how to backtrack" (like Hansel and Gretel of sort).

There are three main types of breadcrumbs:


> Computer Hardware > Computer Cases > Computer Cases (x) > Lian-Li (x)

- the *location*
  - place of page in site hierarchy, like figure here shows
  - problem is: doesn't solve Where problem after user was "teleported" inside the page

- the *attribute*
  - showing the category and page attributes, just like happens now with hashtags
  - a page can be inside more categories
  - problem is: this can be complex to handle and can get very big sizes in some cases

- the *path*
  - showing the user path taken to arrive to the page, retrieving all the links levels
  - typically they are dynamical, because they depend on a specific path, unlike the location ones that are static – so they use *cookies* to keep track of this information

Inside breadcrumbs, we can use separators to differentiate information inside:

- the classic ones are > and /
  - they come typically with stylistic variations (e.g. angle brackets, tabbed breadcrumbs, etc.)
- also to note, the double >>

5 Web Content (From Lecture 4 to 11)

Usability problems can be divided into two main classes:

- Persistent, are those that didn't change much along time (typically, the worst problems and some remained particularly severe up until now)

- Non-persistent, instead, did change (typically, for the better)

5.1 Persistent Problems

The first problem users face is the navigation, because we want to avoid people getting "lost in navigation" and we want to make them aware of *where they are within the site* (always being conscious of the "where" axis), even after *getting lost*. If done correctly, the "where" axis solves the problem.

- This can happen because the page has a layout unclear for the user: it doesn't display the position (e.g. breadcrumb) or has a completely different layout, frustrating him
- If they are not, they get lost, frustrated and timers expire. We saw the problem already in *breadcrumbs*; navigation is not only where we are now, but also *where to go next*
- We can't rely on the user remembering where he has been, otherwise, this would *make the navigation heavier* for him

A simple trick to avoid overloading the user is changing colors to visited links

- This is simple yet very effective, hence present since dawn of WWW (since Netscape, ffs)
- Amazing part, many websites still *don't change colors to already visited links* (75% of websites changes them luckily)
- The paradox is that we like users navigating more, but the more they navigate, the more they must memorize, in terms of pages visited and information
    - they will get tired, with timers will still running
- Hence, remember: graphic effects are less annoying if they reduce effort and give functionality

Users must be able to move quickly: which are the navigation movements that are most used?

- the first movement is *clicking a link*
- the second movement is pressing the back button
    - Users *prefer to navigate back even many times*, even when there is a direct link
        - This is particularly true specifically for multiple clicks and multiple deepness levels
        - The corresponding thing happens with the TV remote: channel by channel instead of using numeric buttons
    - The threshold here is *7 clicks*, rather than 1 direct click.

The key usage is minimizing the computational effort, much preferable to minimizing time:

- users don't need to remember the followed path
- they rely on something consistent outside the website design (always present in the browser)
- no need to look for the right link to follow
- only *backtracking* needed
    - short term minimization, choosing what maximizes results working with minimum effort
    - this is done via "trial and error"

The corresponding usability problem is not allowing the proper use of the back button:

- usually, it happens with dynamic pages badly handled, not saving the navigation state in a way compatible with the back button (like our well-known Uniweb, *am I right*?).
- this forces users to restart the session, clearing local user history hence feels like "starting again"

There are other problems related to navigation: another interference possible is opening a new browser window (it may be new tabs or windows entirely, but the worst one is the new window).

It may be tempting at first, because it allows to clearly separate new content, but this gives a few problems:

- *disallowing the use of the back button*, with all consequences seen above
- this loses navigation history, losing context of what the user was doing
- having different windows open is non-standard and may be frustrating for the average user; he doesn't know *how to get back*

New windows can typically be or two kinds:

- full-screen tab
  - overlaps with existing browsing in a non-standard way, user is confused and can't go back
- not full-screen tab
  - average user doesn't close windows, so typically clicks on the underlying one
  - too many windows and their focus will seem not working given there are too many

In general, opening a new browser requires some discussion:

- this might be useful to separate content, but the average user does not like it, because it seems like the back button is not working anymore, confusing the user
- this overlaps with existing browsing in a non-standard way
  - if the window takes up all the screen, the average user gets confused and irritated and does not know how to go back
  - if the window is not maximized/overimposed but just overlaps, the average user does not close it, but typically *clicks on the underlying window*
  - that page session is not closed, and the screen can fill up with a lot of unwanted "pending" windows
  - even worst, if the user goes back to those windows, or to different windows but not properly handled by the website, a new click will open the same window, which will appear not to be working

So, we need to be very careful to open new windows or tabs (although in some cases in may make sense)

Another terrible choice is the infamous pop-up window (yeah, remember that atrocity?), which are *opened without the user consent*.

Violating web conventions is another big usability problem.

- Nielsen, one of the first to consider usability in web design, referred to the so-called *Jacobs Law*, which states "users spend most of their time on *other* Web sites"
- this results in having users expecting the same behavior in our site as the rest of the Web
  - so, we don't want user to continuously trying to adapt to an original design, because it will lead towards frustration and timers' expiration
  - instead, we should accommodate users things that are used to.

An amazing example of this is the Zinc Bistro website:

- it is not clear where the menu is for going from page to page
- the menu is located in a particular position (in the egg) and only 0.3% of users understood that they had to mash there (also, it is not known why some eggs are activated, others are not).
- certainly whoever designed this had thought it was an original design, forgetting about the user needs and experience

Respect also the What axis properly:

- <u>avoid using empty language</u> with little content and lot of slogans
  - users putting time and energy to read content/information is not useful here, especially with propaganda content
  - good counterexample, the old design of Montblanc Pens website
    - here the info contained in this axis is "why should I buy this pen", is not saying anything to the user; even if it is a specific page
    - too many slogans, very little usability
    - subsequently, a redesign made a clear description of the pen appear easily, hence helping the user and giving something useful

The user who arrives at a certain page expects content, not pompous language and difficult, which causes navigability problems. Instead, the What axis is like an average computer scientist: very little text and understandable (there's evidently crazy people like me who like writing, but never mind).

Usability problems can potentially lead sales in a different way: *content is important but form also is*.

- many problems come from <u>usage of difficult and monolithic text</u>
  - it often happens in sites that ignore user timers/feelings (e.g. public/governments sites – private/monopolist ones)
  - they usually don't remember a simple thing: web text is different from normal text
    - so, it needs to be simplified, because reading on a screen is more difficult and we should counterbalance the additional effort

Here are some rules for web text:

- Base rule: *100%* normal text → *50%* web text (half)
- If generalist audience (generic text): *100%* normal text → *25%* web text (a quarter)
- Moreover, it is helpful to *start with the conclusion*, subsequently expand

Some counterexamples are Social Security Online and Directgov, which definitely don't consider the usability factor of putting wall of texts (hence, myself included in this file, it's suggested to structure things properly, which I try to do with itemized lists, different formatting, carefully spacing things).

So, to write text suitable for Web, remember to start with the *conclusion* in the "end-start-content" form.

5.2 NON-PERSISTENT PROBLEMS

Let's discuss now the non-persistent problems; this usually regards content and how it is presented.

Consider for instance the splash pages, which a welcome page useful as introduction to a website

- it's important to *avoid them at all costs*, because users don't like them and make them lose precious time – they are basically homepages barebones, without 95% of the content
- to go the real homepage, users will need to click once more
- they are even worse if animated
- a good counterexample is the Wynn Lass Vegas website, which basically it's all fancy but it's completely useless, little links and zero usability

Another important thing is to avoid asking for personal information, given not all information may be accessible to the user at once and this can cause anger and frustration – hence bye user (this is something that is so present in website and apps today still and it's infuriating to say the least to me).

About this specific problem:

- this can occur in case of premature registration (go on here only if you're registered – a lot of apps and websites still do this, and I think this is infuriating), in Homepage or other pages
- there is the disadvantage of the further *computational effort* involved to remember a new couple "login and password" (not liked compared to classic email) and a lot of lost time

Other thing to consider is the trust given to the site, which in case of premature registration must be established (become *trustworthy*)

- user will give their information only if he knows us, otherwise it's
- this brings a potential decrease of users on average 1 out of 10 (here, some generic website examples were mentioned – the classical "you can't do anything if you don't login" kind of thing)
- only ask for personal data *at the end of the process* (like you see in checkout in many sites)
    - first gain trust, then ask something back

The scrolling is a bad problem in sites (especially if it happens consistently), users may get tired.

- They are willing to scroll *1.3 screens* – 1/3 scrolls
- At most, in total, they will see a total of *2.3 screens* - 2/3 scrolls), and everything beyond that is just not viewed anymore

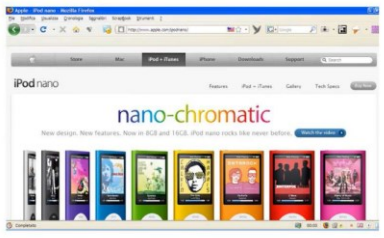
So, *give already all the information visible to the user*. Some percentages:

- First visit to home page: only 23% of users scrolls (the majority does not go down)
- Internal pages (users are more benevolent and they want to see more): 42% scrolls
- Repeated visits to the homepage (users are more demanding and they won't scroll): 14% scrolls

Scrolling definitely *depends on screen size* and also on *devices used*:

- The classic max reference size (because not every user maximizes to full screen, even on big screens) is *1024 x 768* pixels
- *Net-books* have reference size 1024x600: what happens is that some websites may get truncated badly switching from 2-3 screens to 1.

Figure of iPod nano website here to discuss over sizes. In home, without scrolling, only images and a few text are visible (like you see here). Subsequent versions made the content better, but you see only the image again.



To avoid problems of losing information, *safety size* is of 800x600.

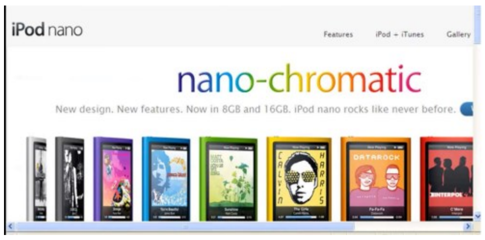
When designing a website, apart from reference sizes, we must consider another big problem: the so-called frozen layout problem.

- The problem is having too much information expanded on a geometrical axis (both vertical/horizontal one)
- Given page does not adapt on screen, information will clash on the other axis the frozen layout problem occurs.

This can lead to many serious visualization problems:

- One case, with a big screen, information stays smaller (too small, comparatively) part of the windows, with very bad visual results
- The other case is the opposite: when the window is smaller than the set/min size (either by user choice or by necessity for a screen that is smaller, think about smartphones), then things go wrong for lack of space
- Now we have a lot of resolutions, and we have to consider them all

In this case, to access horizontal information, the users need horizontal scrolling; this is rather uncommon on the web, and it's *one of the most hated by users*, because this does not belong to our *classic information modalities* – see figure here.


iPod nano
Features
iPod + iTunes
Gallery
nano-chromatic
New design. New features. Now in 8GB and 16GB. iPod nano rocks like never before.

Infact, in every single system (may it be books, screen pages, etc.), we're used to have the horizontal axis always set (per say, one leading dimension is always set).

This does not hold for everybody though; for example, Asian people have a very different writing from use westerners; besides that, there is an additional serious problem: it augments the *dimensionality* of the information space, going from a 1-dimensional to a 2-dimensional space, requiring *a lot more computational effort* to be had.

- A good example is again Engelbert (mouse inventor) and its external device to control with the mouse, which allowed the text input of 31 combinations using only 5 fingers (have a look here)
- While good in conception, it had a non-linear complexity, given the number of parallel commands needed to do actions

Other examples of such situations are other Apple sites (iPod nano/iPod touch):

- they have one common bad traits, like too much space used improperly (too many images useless completely for browsing, but just there "because it's good aesthetically" or the iPod nano website, broken when seen on little devices)
- generally, they had an empty layout and information appears after 14 scrolls

In any case: scrolling is better on mobile and if layout encourages it, then people are actually inclined to do the effort and go down.

Written by Gabriel R.

5.3 ERRORS IN DESIGN AND INTERFACE

Technically, the problem is in bloated design, inserting elements over-the-edge trying to impress people with special effects, fancy but not useful and extremely annoying, both aesthetically and practically

- It causes computational effort increase and slowing users timers down
- This happened at the time of the browser war, with commands that break standard
    - e.g. *marquee/blink*, used respectively to make text flow and scroll – to quote the "blink" tag inventor: "One of the worst things I've done for the internet")
    - blinking text requires big computational effort (we're not used to it), and some browsers don't even allow it anymore (luckily so)

- In this category, also to note the multimedia abuse
    - auto-starting audio, 3D interfaces

Bad examples are:

- the famous local show in London and its horrendous color pattern design
- the MIT Center for Visual Studies, artistic but unusable, so spaced and so bad to look at
- the lovely example with the *Dancing Jesus* web page from a Simpsons episode – this term will be used a lot by prof. Marchiori, to remark the concept of "so fancy but so unusable" (image here).

Another problem is three-dimensional movement, fancy looking but a repeated failure in all media it was attempted. Physically, this is perceived as "having too much work" to process. On the web, the reasoning is similar: *computational costs* too high (both in *perception* ad *usability*). Examples here are:

- the Aspen Movie Map (Google Street View back in 1980 by MIT), with moving via buttons needed
- the Ford Acura website to see the cars' interiors
    - here directional inputs/keyboard shortcuts are the only possible input – not standard for normal users
- the "dragging to move" inside Google Street View itself
    - non-standard movement of dragging, even in more dimensions of complexity – very clunky interface and very few users know how to move and to use it properly
    - users had to understand a bit how to use it and initially it was not a success
- the BumbTop interface, where the desktop is redesigned to be a simulation of a real desk
    - you can drag things, pile them up, circle them and activate in-context options: you don't learn new concepts, everything is already understandable and visible to the user
    - anyway, it takes time to learn, because it completely revolutionizes the well-known desktop interface and simply taking reality and make it virtual does not work
    - Google bought it but apart from the initial "wow" effect, it was too hard to interact with

So, even without Web boundaries, we must consider inertia (the will to avoid change)

- Instead of 3D views, it's massively better to offer 2D snapshots of 3D objects: they have relatively low complexity and allow the user to explore the 3D aspect of an object)

Another way to impress users with special effects is plugins, but the problem is *they are not standard* (they *need to be installed*).

- The normal user doesn't trust installing new things, only gathering some trust after *one year* from distribution
    - *trust factor* – user says "I don't know, I do nothing, I avoid problems"
- On average, every request of this kind makes 90% of non-loyal users go away

The most famous plugin was definitely the "mythological" Flash Player:

- it gave content creators a new way of expressing themselves on the Internet
- it also gave birth to a lot of *bloated web designs* and *higher computational cost*
    - many loading-related problems, constant updates, non-standard interaction
- it died in 2020, giving a lot of usability problems in general
- there were some good websites, like the Tiffany&Co website, which tended to not be used because of its "original" design requiring the user to adapt to that and not interacting well with search engines

Many non-proprietary technologies have risen (HTML5, JavaScript libraries, etc.), in place of proprietary ones (Adobe with Flash/Gopher vs WWW) because they integrate automatically and natively.

- Using standards doesn't imply standardized technologies force things to go well always
- Not all HTML5 solutions avoid bloated design
    - E.g. Speedcrunch website: confusing and puzzling interface in one of its versions, with buttons that don't do what they seem visually (betrayed user expectations)

5.4 Visual media, metaphors and menus

If 3D interfaces are problematic, there is a multimedia type that has a low computation cost: video. It's quite attracting as a media (consider TikTok), given its computational low cost, but we have to consider:

- Bandwidth usage, which can be heavy for some users but also for servers
- Typically, video exceed user timers (note: depends both on target and platform)
    - the preferred *average* time is *1 minute*
    - the *maximum* suggested time is *2 minutes*
- These times contribute to the timers, so videos should be inserted taking a lot of care of users and their navigation goals – never force users, but guide them naturally towards the goal
    - Good examples are Victoria's Secret website, where even with long videos, users are attracted to those

There are also problems related to the visual metaphors:

- users see something and have corresponding expectations
- if expectations are disattended, users will be puzzled and confounded

Some problems of this kind are:

- scroll button replaced with an image
- image with written "Click" but actually not clickable
- some bold text which seems a link but actually it isn't
- content which seems like image but it's not
- non-clickable/partially clickable buttons

Some website examples are:

- the Norah Jones website and the sun scrollbar or links that seems clickable but it's not
- the Agricultural Fertigation website, with images that seem clickable, but they're not
- other websites like Bicsurf/Organize Everything/Bank One/Goetz, where products seem clickable, but actually they are not or even using different fonts on text which is not a link
- the Burger King website, where users have to find work clicking on "click the Fries/click the Whopper" – false assumption of thinking the user might know the context
- Google Images, with its little camera icon near to the search icon which indicates Google Lens, which goal may not be clear to a normal user: this allows to upload images and query a search

In liquid design, buttons expand and shorten; the mistake in this context is to <u>make clickable only the text and not the button</u>. The betrayed visual metaphor is noticed not in small devices, only when expanding or on large devices.

- Things get worse when the mouse is placed over the button, it goes into prominence; but you can´t actually click everywhere (double visual metaphor betrayed)
- The converse also applies dually (so, links which in reality are buttons)
- E.g. An old version of Rotted Tomatoes, where asterisks and tomatoes are used to represent negative and positive reviews – another betrayed metaphor

Another problem is *excessive convergence between desktop and the web*, like <u>menus</u>, something the user long knows since the desktop nature:

- it seems like a win-win: no new training for users, low computational costs, saves space and fast navigation
- as a drawback, they are more suitable for a desktop environment, where they show commands, but on web they show information, with a tree-like structure
    - potentially, they can possibly contain too much information
- we can have the risk of *menu explosion*
    - too much information; in worse cases, it may even go out of screen
    - nesting many levels of menu one after the other is another problems; this makes navigation heavier and may get user lose his focus

The situation gets even worse considering the *reference sizes* and mix things that don't go well with each other (not everything combined is good). For example, consider mouse and menu: by themselves, they are good tools, on the web they are pretty dangerous and disruptive.

- 83% of users fail to hit the right box in web menu at first click
- 54% get out of the menus, which means that the menu closes, and they have to start browsing again, increasing the level of frustration

5.5 THE MOVEMENT PROBLEM

Another problem regarding user habits is movement: how will users move on the screen?

- on average, user wants to go from A to B and simply *follows a straight line*
    - this happens because computationally it's the cheapest one
- if there are too many levels, we go above other layouts and other menus, activating them or possibly going out of the current one
    - see the Black Mountain Bicycles website and its many menu levels with too many voices
    - in a subsequent redesign, menu was spread horizontally, avoiding menu exit
- this forces users to force the pathfinding algorithm, forcefully following the menu layout
    - 92% of users go out of the menu path, leading to even more frustration when wrongfully moving the cursor and maybe losing all the menu path – lost time and anger

It is then recommended to create *multilevel* (cascade) menus as *two* as max level – so, like desktop, menus should have a structure, but actually a solid one at that.

- The shortest path rule should be considered, so design *fault-tolerant* menus for such paths (so, menus that never close automatically, avoiding wasting user time).
    - For example, introducing a timeout inside menu voices can be useful
        - Instead, consider fast floating images/a lot of text inside chatbot popups/funding popups can be a huge mistake
        - In this case, never annoy, but be original and place these in an interesting position

A useful advice is also *not having that many scrolls* in a web page to go down and even voices in menu hidden and not opened by default, forcing the user to make a computational effort

- For example, the site heart.com, which implemented fault-tolerant menus making the site bigger
- It was then restyled to be single-level

A good way can be compressing all the voices, avoiding overload of information. It's better to have a vertical menu not overcompressing all the information in a forced way, allowing a scrollable design.

5.6 THE TEXT PROBLEM

Consider these good rules for good text:

1) It must be readable
    a. Avoid diminishing font size to put more information/to fit the layout
    b. It has to have a minimum size of at least *10 points on every screen*
2) There will always be someone not comfortable to the text size, so it's recommended to create different people resizing options, so each user can choose what he prefers
    a. Don't block resizing inside websites or apps
    b. Avoid putting buttons like "+/-", because they are not easily seen by users
    c. Do like browsers: allow an integrated zooming tool
3) The user should be able to easily recognize text as such and beware of different fonts and attributes to create graphical effects
    a. A good effect can be adding border to text to make that stand out from background
    b. Usually use *one* font (maximum two): if in doubt Verdana (now: some sans-serif)
4) Beware of contrast of text over other things

Moreover, don't put uppercase text:

- this is generally less readable and more computational effort is required
    - we might want to get user attention to an important thing "in a fancy way", but we waste user energy), because it takes 10% more time to read (non-standard)
    - remember also you're on the Web, where uppercase is considered shouting (bad practice)

Another worse tentation is using graphics instead of text which has many problems:

- bad scalability (usually, no resizing employed)
- page size gets bigger
    - longer loading time
- no copy and paste allowed
    - maybe the users want to choose some text and do searches with those; even if it's a relatively small percentage, allow all users "be happy" with their experience
- bad interactions with search engines
    - usually, engines don't read images, they are just starting to do it now, so the page won't probably be indexed in search engine results

Another "big curse" is the Lorem Ipsum curse, very long text usually put as placeholder in layout to understand how to position things later on.

- It stuck as an habit in web/layout design
- It is just very long latin text automatically generated has no meaning nor context
- It puts first place layout, second phase text
    - Remember the users first want text inside a page: so, wrong from the start

This way, text is seen as something detached from design and typically his content is ignored (users may consider it as a priority, because it's the information required to understand the page)

- Text is not a block element and to guarantee usability, it should be structured properly
- Users may see these "unplaced" elements and not gather a good idea of our site.

Users usually preliminarily see a web page in a so-called *scanning mode*:

- They get a glimpse of the page "as flash", quick and fast attempt of understanding the context and after we start reading), quick and efficient, not precise (*scan*) - not necessarily ordered
- This analyzes the basic components of a page in a fast way, then creating a mental map of the overall information, which happens in a continuous way
- The scan happens continuously with our eyes, so a good design will minimize this effort, keeping into account the linear progression of this scanning.

When our eye meets a wall of text/big text box, doesn't perform a good fast scan and fatigue grows: we need a *good structure*, so fast scanning can be successful.

- For example, in the InFocus website and the subsequent slide full of text, instinctively, we create mental notes constructing how information is built, of text itself, structure, titles, menus, etc.
- This is where the "Curse of Lorem Ipsum" strikes: we use text as placeholder, but we need the final text to understand how to structure the content inside a page, helping user mental map
- Don't treat text as blocks: graphics and text are not always two separate things

Not only is important the structure, but also the text *content*.

- A good practice is to *divide text into blocks*, but also to break it into *smaller* blocks, which can be done also using descripting *titles* to help create a categorization/subclassification
- This happens also via *keywords*
    - They should be kept short, relevant, useful to make scanning fast and easy
    - Users can memorize up to 6/7 keywords, remember that
- It happens also with *graphical elements* to distinguish things
    - E.g. hashtags to get topics in Twitter; they should work as summary of content
    - Even different formatting, say bold, can help

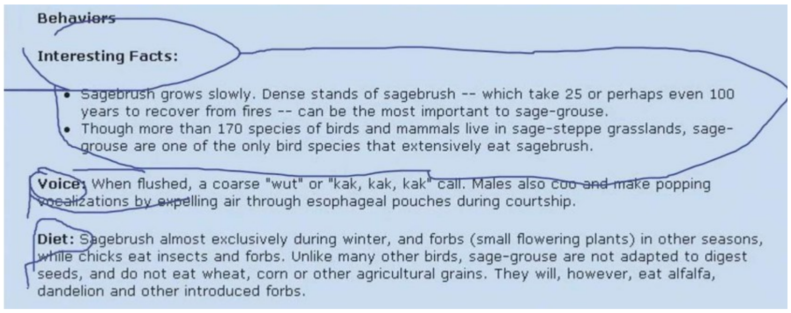
A particular case for keywords is the link case, noted even more than keywords:

- the common mistake is to *give them the same title the page it redirects to*
    - it does not help always user browsing and scanning in page, may get confusing
- an even worse case is *using the web address of the page*
    - users will never memorize it
- do not take full sentences and full keywords
- avoid too similar names between them to help scanning
- another infamous example is the "click here"
    - it invites users to "do effort" in order to see what the content is about and puzzles them as a result (not always effective)
    - we waste text for an obvious action the user is going to do anyway

A good practice to give structure might be using *itemized lists*, via simple HTML and CSS, this way giving visual cues to where to look at, making content more digestible (what I also tried to do here).

A minimap will be built much easier organizing content much better. This improves user satisfaction up by 47%.

Typically, itemized lists should have at least 4 elements (1 to 3 just use text, over that adds complexity). On the right, an example of structured text and users mapping.



Behaviors

Interesting Facts:

- Sagebrush grows slowly. Dense stands of sagebrush -- which take 25 or perhaps even 100 years to recover from fires -- can be the most important to sage-grouse.
- Though more than 170 species of birds and mammals live in sage-steppe grasslands, sage-grouse are one of the only bird species that extensively eat sagebrush.

Voice: When flushed, a coarse "wut" or "kak, kak, kak" call. Males also coo and make popping vocalizations by expelling air through esophageal pouches during courtship.

Diet: Sagebrush almost exclusively during winter, and forbs (small flowering plants) in other seasons, while chicks eat insects and forbs. Unlike many other birds, sage-grouse are not adapted to digest seeds, and do not eat wheat, corn or other agricultural grains. They will, however, eat alfalfa, dandelion and other introduced forbs.

Be careful not to exaggerate:

- the efficiency of a bulleted list decreases *linearly* with the number of lists arranged vertically and exponentially with the number of lists arranged *horizontally*

Sometimes we can get the guillotine effect, where text will be *cut* and then lost; particularly, this happens also on internal scroll site, with devastating effects compared to the browser one.

- E.g. menu voices with text that gets cut like the Amsao.it/Atlantis websites
    - typical design error it's thought text comes later in design

Occasionally, a blonde effect occurs, signifying a misinterpretation of a situation leading to comical or impractical action (user scanning abilities exploited/final result is the opposite), often stemming from stereotypes related to blonde hair (a bias that's unfortunately prevalent) – reference of this metaphor here.

- For instance, this might involve designing a feature like a website minimap, which seems helpful in theory but can actually unsettle users: it's not standard, forces users to do new things and learn
- During scanning, users might get zones wrong – so keep this in mind while designing

5.7 E-COMMERCE, ADVERTISEMENTS, PRICES

Among the various generalist websites, a very important category deserves to be discussed apart: *commercial sites*, specifically e-commerce websites, which have some peculiarities and can be more precise on their guidelines.

- The most important thing there is the product, but it's not the only answer
- The average user considers another equally important component: the price

Users want to *know the price of a product in an easy way* (remembering the timers and rules on the associative mental map). Remember, the price has to stay *near the product* (just like a physical shop: some sites do not do that because they think it ruins the layout, while it's simply useful for all users).

- There is the problem of "hyper-association": the price is there, but it's not on the product, it's some links away (price hidden until last click on the product)
    - This ramps up the computational effort and loses the scanning map created by the user
    - E.g. the Atlantis website to get new trips and accommodations, you get to experience the full site in order to get a single one
- This causes a loss of primary information, where the user wants to know more but is forced to a click "hoping" that things will go well

These are called gambling clicks: users do a kind of risky gamble to "try to get the information if they're lucky".

- These don't attract users and cause mental stress (about -40% in site satisfaction)
- They tend to not be clicked (only about 30% of users use gambling clicks).

One of the main problems is that sometimes there is *not a definite price at all*, for instance in case of consulting sites or representative sites, where the price is per store description.

- No justifications are needed: simply give a price; when the exact price is not available
    - we can give an *approximate* price (best)
    - or we can give a *range* of prices (second best)

Remember also to not test users' trust, but remember the guidelines seen about the "who" axis.

- The other worst thing is abusing other techniques, typical of classic media inside the web: the two are completely different
- In the classical media (posters/TV/papers/radio, etc.), you have a short time of exposure, and the user has to be impressed and hit by something he sees

Classic advertising makes in practice these techniques using the product price itself.

- A common way is the fishing price technique: using a bait price, different from the real price
- Another one is the net price, which does not correspond to the final overall price

In both cases, the price that is shown is different from the real price. Even if Internet is virtual, remember we see the website as a shop. Let's give some considerations about short/long-term memory:

- advertisement is needed to attract someone into a shop, leveraging the short-term memory
- the *primary recall* (product and good price) tends to stick to *primary memory*, while the *detail* (exact price) tends to fade staying in *short-term memory*
- so, inside web, users will remember the price, so don't change it to encourage purchase

This technique does not work in a website and when applied it irritates users.

- Approximately 90% of users will leave the site right away because of the “fishing price”
- The remaining 10% has a trust decrease for the site (frustration, etc.): as a consequence, timers diminish by 50%
- The other trick, the “net price”, in which you don’t indicate VAT (Value Added Taxes – IVA in Italy), the users going away are 85%

A few bad examples:

- Overstock website with unclear prices
- Target website with premature registration in the old version/in the new version, fishing price in the “from this price” section (late section inside website, not good for users)

Typically, the final price is obtained by completing the transaction (the so-called *checkout*), which looks OK, but actually it’s an optimal solution (like Walmart, putting items in the cart and actually gives information of users’ action). In any case, simply *inform users about all prices* (like shipping costs)

To trigger the “ice-cream effect” (like ice-cream, giving something nice to the user), we can use the “magical” <u>free</u> word to give something to the user (e.g. something simple like making a free downloadable .pdf, for example sending him an e-mail).

- Giving something for free to users gains users trust and also encourages users to give away personal data (like the e-mail)
- This also gives users happiness and joy

Let’s pass now to the other side of e-commerce sites: <u>the product</u>.

- The big error is to assume the user already knows a product and comes to the site for the price.
- In fact, the user always expects a <u>complete description</u> of the product in the site (even if users don’t understand the exact specifications).
- The perception is positive, you seem more professional; incomplete descriptions bring users to other competitor sites and more professional ones.
    - Sites with bad product descriptions bring users to look elsewhere in about 98% of the cases

So, what price to ask for then? If the price is lower, will the user come back?

- On average, lower price up to -20% with respect to the competitors brings back *5% of the users* (appearance is more important than content in its perception: we gained less trust).

Beyond the description of the product features, it’s also important the <u>visual description</u>.

- If interested, the users want to see a product at maximum level, even *full screen*.
- In these cases the timers practically *turn off (freeze)*: when the user *chooses* to have more detail, he comfortably waits for a more detailed image
    - Note the critical “choose”: beware not to instead force a detailed view from the beginning, with timer impact.

Some website examples:

- Good example:
  - Amazon, with free zoom and complete control over images themselves
  - A website about pots, approximative description and clickable images about products which one can zoom in
- Bad examples:
  - Kitchen website with very small images and no description
  - The SATA website with unclear descriptions and unpleasant layout to use and to look at

The best way is to offer *perspective views with high detail*, mimicking the guidelines seen on the *3D as 2D* discussed some time ago: give *very simple descriptions and use 2D* to avoid computational effort going up.

- Good examples:
  - The Abercrombie website with thumbnail and zoomable previews of images, products price in evidence and scrollable images
  - The Overstock website allowing multiple takes of the images shown
  - The Crutchfield website redesign allowing a complete show on multiple images of products

- Bad examples:
  - the J.Crew website with popup windows, confusing menus and not so clear navigation between images inside the site
  - the "Cordura" website assuming users know what the material is). Always leave the choice to the user to what detail: so, there is always the need for *detail dependant on the context*
  - the Zara website, with images moving across all the screen, lots of scrolls and very few descriptions, with user liking decreasing each time

6 USER BEHAVIOUR, ATTENTION AND IMAGES (FROM LECTURE 11 TO 13)

We saw already many characteristics of user behaviour, let's delve deeper in the analysis.

- In 1990/1991, the Poynter (well-known and respected journalism school and research organization) research started a behavioral analysis of users with respect to newspapers, tracking the eye movement, typical reading patterns (a complete overview here).
- Precise behavioral classes have emerged (we're very similar). The same fact there are fine-level rules that are valid for almost all of us, is far from obvious (it's also good news).

Let's start to see how users behave on newspapers:

- the high attraction spots are *photos*
- the other attraction is *color*: the more the page is colored, the more it is perceived as *rich in information*
  - because of this, newspapers have their first page printed in color

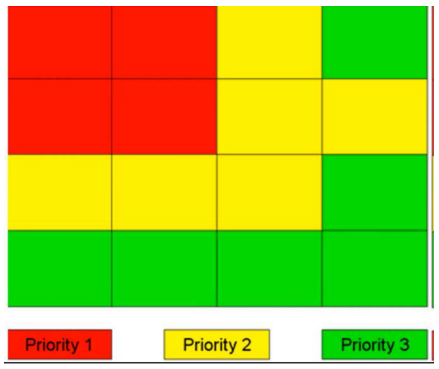
Definitely, in the battle between text and images, *images are seen much more than text* (*80% vs 20%)* – our life is a video infact (remember the assumption over Lorem Ipsum).

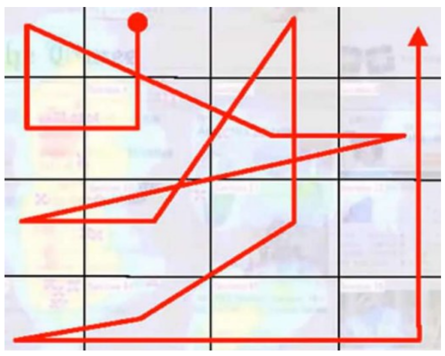
- Pieces of text that are together with an image are seen much more than without a companion image
- Take for instance the *entry point* in a newspaper page: *it's actually the biggest image*
- Also, in page division, *two open pages are perceived as one big page* (and the biggest image is seen)
  - user likes this more, because keeps everything under control

In case of Web, people just thought that the Web would simply follow the same rules, but as said it's a much different world than "traditional" media.

Consider the typical movement in web homepages: historically, to track what users were doing, was used the eye-tracking technique, finding different patterns thanks to which pages are scanned (understanding with color how much the single zones were used, so-called "hot zones"),

- This one uses thermography of a page with +/- hot areas based on how many times visitors' eyes fall on it, understanding the attention map
- There are different zones (from 1 – Higher to 3 Lower)
  - Red represents Priority 1
  - Yellow represents Priority 2
  - Green represents Priority 3
- The entry point is the top-left corner

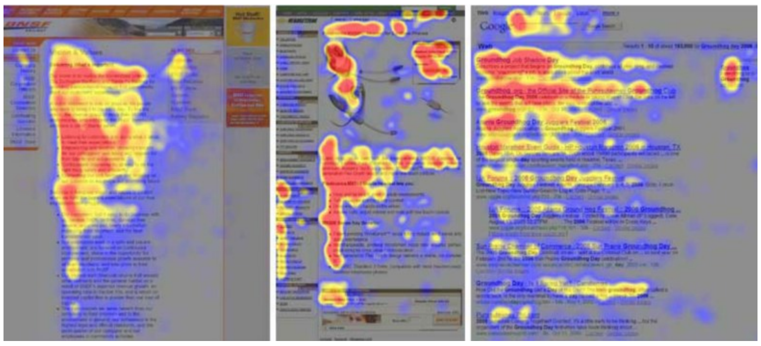


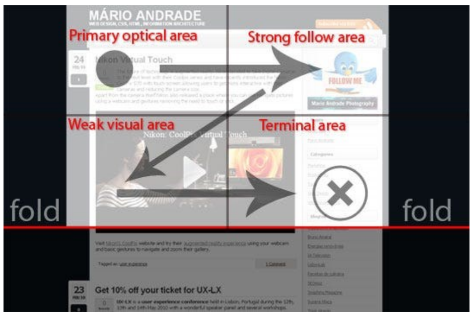


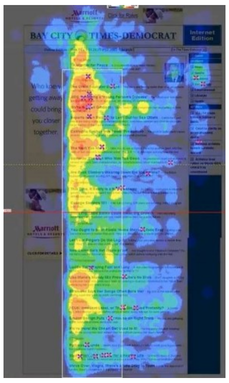
6.1 Attention Zones, Spots and Areas

Eye tracking is an approximation: the attention map shape follows a F-shape/ice-cream cone pattern (also called Gutenberg diagram). In that, users tend to start at the top left of the webpage (usually where the logo or headline is located) and then move horizontally across the page.

They often pause briefly at subheadings or key visuals. Next, their gaze moves down the left side of the page, creating a vertical line of attention. This pattern resembles the letter "F." This area is common in the point where information is mainly concentrated. This is represented by the following set of figures.







There is another curse derived from attention zones, which is mainly linked to *scrolling*. The majority of users scrolls *page by page* (because intuitively it's faster and gives quicker access to new information).

There is also a "blind spot" in pages, which changes along with screen size. This is usually where the scrolling happens, and this will never be seen.

Actually the battle between text and images is won by the text. As a matter of fact, the real attraction point is just text; a logo without text inside or nearby can confuse users (habit of having it placed top-left over a portion of the screen).

Some advice about improving information flow when using text:

- The point of attraction (top left) needs text, the logo should always be accompanied by some text perhaps responding to the What axis
- Text on a single column is better than on multiple columns as in the newspapers (the effect is very similar to horizontal itemized lists)
- Keywords placed close together dilute the importance given in the scan
- Better to use different formatting to mark keywords, like bold, underlined, increasing font size or separate the keyword/s in a row by themselves

Also remember some rules to use bold:

- use bold by itself in a single line (like titles)
- make the font bigger
- use hyperlinks

Shorter paragraphs attract much more than longer ones (typically, at least double); the same happens for titles, so *shorter titles are much more successful* (much less computation effort required) than longer ones.

- Having a text broken with more paragraphs relaxes timers and encourages reading (+100% over timer relaxation – valid both for text and emails)

Titles are important attractors, but we can actually give short explanation/summary, called the <u>blurb</u> (between title and text). The attention of the usage is taken just for a limited amount of time and comes only for a few things over limited time. This is also the case in the page you can see here.

- Blurbs *relax timers* and make users digest more information and *do not* significantly change users in going on with navigation
- They cause a bigger *return rate* (because there was much more significant content) at about +20%

The blurb can be perceived in different ways is not just seen as a single block of text by users but can be divided itself into various zones and it's perceived asymmetrically: what you want to do is *put important words on the left of the blurb*.

- The algorithmics implied are asymmetrical and the leftmost part is well assimilated, the rightmost part is not. It is way better *to have the right words on the left side*.

Another question - is it better to have pages:

- *skinny* (few links/few scrolling) or
- *fat* (a lot of content/a lot of scrolling)?

The *skinny* pages get much *more* focus over the *fat* ones in the *color* zones.

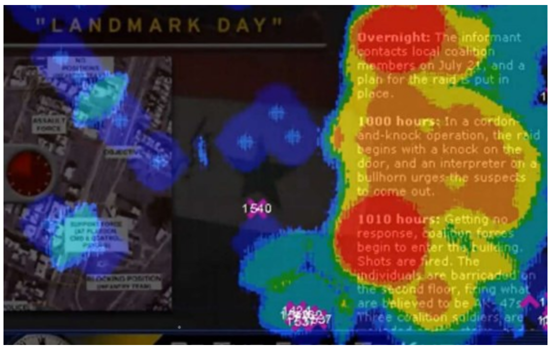
We need to find the right compromise between separation but also compactness of information; excessive doesn't imply better usability but actually causes *diluted design* (diluted = lots of things)

- Separation brings faster scanning, useful for *navigation* pages with *links* and little scroll
- Pages with a lot of content and a lot of scrolling have separation as a bad thing, creating diluted design (this design works well for browsing pages and a little scroll)

Let's switch over *images*, which are more important when compared to text (like e-commerce).

There is a *minimum size* of *at least 210x230* pixels (below this, images are too small, which is ok for icons but not for images), which is the minimum suggested size in normal conditions (because, differently from text, images can change).

As you can see from the left, it doesn't matter, the most focused part is always text.



Overnight: The informant contacts local coalition members on July 21, and a plan for the raid is put in place.

1000 hours: In a cordon-and-knock operation, the raid begins with a knock on the door, and an interpreter on a bullhorn urges the suspects to come out.

1010 hours: Getting no response, coalition forces begin to enter the building. Shots are fired. The individuals are barricaded on the second floor, firing what are believed to be AK-47s. Three coalition soldiers are

Images have a peculiar *magnet effect*, even if they fall behind text for users' priority (because they attract clicks over a minimum 20% of users). Remember also:

- whatever the *image*, *each one should always be clickable* with an *event* related to them
  - when this does not happen, users tend to get frustrated and click even on empty images
- they should always have a context and a meaning, putting a click even if it's already there
- they are much useful in the scanning page, even as summaries

6.2 Fitts Law, Magic Pixel and Menus

The user always wants to be in control and be careful to instances that take some control away; in any case, always leave the user *an easy way out*. There is a missing piece still: *time*. There is a beautiful result that comes to the rescue, which is the Fitts Law.

$$T = a + b\log_2\left(1 + \frac{D}{W}\right)$$

- ◆a = start /stop
- ◆b = co-speed («slowness»)
- ◆D = distance
- ◆W = width

This is a predictive model of human movement working in a single dimension (as a straight line) and studies how fast the user moves inside a webpage.

Let's describe its parameters:

- a: start/stop → the time it takes the user to start/end an action
- b: covelocity → inverse of the speed the user has to move with the mouse
  - this depends on user, tools at his disposal and programmer's cautions
- D: distance → distance the user has to travel
- W: amplitude → size of the target to click
- T: time → average time to conclude the movement

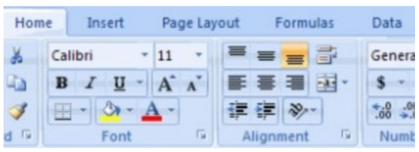
The law predicts that the time required to rapidly move to a target area is a function of the ratio between the distance to the target and the width of the target (modeling the act of pointing – distance matters much more than width in the final target).

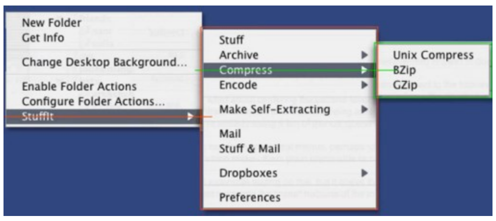
- Note that T increases as D increases but decreases as of W. Distance does not matter much in the final time T but weighs much more the amplitude (W) of the final target.

A ratio makes it take less time the larger the object, and the smaller the distance. The logarithm has a "cleaver" effect, making sure to cut into time. So, we are not sure this movement is always satisfying. Infact, we usually have:

- *point-and-click*
  - this is simpler and preferred by users
- *drag-and-drop*
  - this requires bigger co-speed (depends on tools/user and it's the opposite of speed, so slower speed)
  - leave it only as an option to the user because it's a muscle fatigue
  - this should actually be avoided if it can be substituted with point-and-click

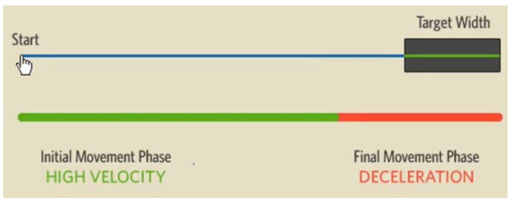
Fitts says *bigger objects are easier to click* (minimizing object distances) or *reduce distances, so they have the same size* and *button size being proportional to how much a button is used (target size rule)* – e.g. the 2007 Office redesign which does this with images for its buttons.





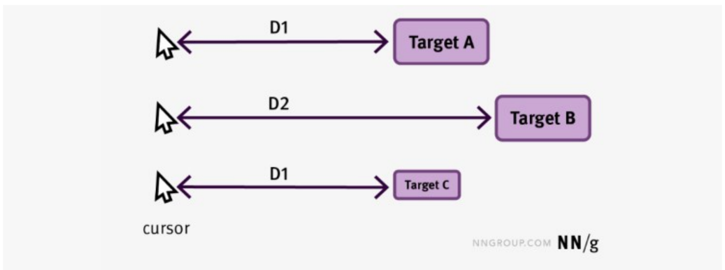
For example, in a menu, we keep commands close, but *proximity is not the only factor*: given Fitts works in one single dimension, if area is the same but has different shape, *the best button depends on the angle created by the starting point and the line.*

Even if distance is not that much, also width and size matter: *area* and *number of points* (space between objects) to reach things is important, giving we want to cut distance to achieve the desired goal faster.



To summarize Fitts research original goals:

1. *The bigger the distance to the target, the longer it will take for the pointer to move to it*. In other words, closer targets are faster to acquire.
2. *The larger the target, the shorter the movement time to it*. In other words, bigger targets are better.



Fitts' s law says that the time to reach Target A is shorter than the time to reach any of the other targets. Although Targets A and B have the same size, the distance from cursor to A (D1) is shorter than the distance to B (D2), so movement to A will be faster. Target C is placed at the same distance (D1) from the cursor as Target A, but it's smaller, so it will take longer to move the cursor to it than to A.

For curious people (like me), the original research can be found here (and a subsequent 1992 one here).

Can we draw useful indications to improve interfaces?

- In smartphones, physical buttons tend to disappear because they cause energy and excessive time consumption

There are special places according to Fitts Law:

- *the borders*
  - they may seem like "buttons with infinite size", so making something large and simple to immediately interact with it)
  - it may seem like crazy, but it's easier going to the opposite side of the screen rather than clicking an object a few pixels away
  - given humans are asymmetrical, users are more comfortable using these ones

In this case, we may find two schools of thought:

- Windows-style, in which options are inside the program
- Mac-style, in which you have the menu over the program window (always overlaying on top of everything), avoiding problems over the y-axis.
  - This is great following Fitts Law, and these menus are 5 times faster (+500%)

Also consider the *taskbar*, which has been developed afterwards following Fitts Law. Another example are the top borders of windows redesigned to accommodate bigger icons for the most used actions.

There are the so-called "magic spots", in which *corners* are used as interface buttons, because they don't require braking time and have two infinite areas (with two borders at the same time).

For example, the scrollbar designs of OSX systems apply this principle and, as a matter of fact, are much less stressful than Windows ones. They don't need braking time and the objective area is very big, unless the user hasn't got a reduced windows on his screen.

Written by Gabriel R.

The same applies to the taskbar: the largest "Start" button is in the corner. There are ways and means to implement the taskbar:

- previously, in Windows, it looked like a button, with its own shading (a feature that is no longer present)
- it also had a border which made it seem like it was not clickable
- the shading had 2 pixels (before the start of the button) that could not be clicked, which eliminated the "magic" of the Task Bar (which is why user acceptance plummeted)

Careful that anyway, staying specific to the web, usage of windows can diminish the importance of magic spots for navigation.

- For example in a trackpad, you have everything flat but here you lose the magic zones, especially when going towards borders
  - because the width of target is reduced near edges and this speeds up the navigation, making it easier

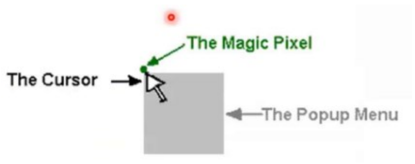
Another thing to note: are all corners equal? No, some are easier while others are more difficult, given we're *asymmetrical* as human beings. Here there is a ranking of corners (from best to worst) from a right-handed perspective:

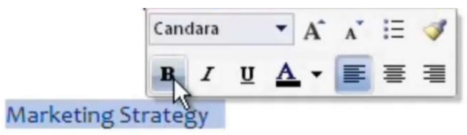
- Bottom-right
- Top-left
- Top-right
- Bottom-left

Note that the worst point (bottom left), is the point that is most used by interfaces, for a reason of habit. By now even though the corner is the worst point (consider Start button position inside Windows), it's the common case of usage in UIs.

There is another magical place called "magic pixel" (or "magic spot"), which refers to a hypothetical point or pixel on a user interface element where an action is triggered without requiring a physical click.

This example can be briefly described visually by right figure.





For example, consider the context menu, which many times, like Word, is in the best spot. This is present in the example here on the left, which is a *popup* menu (Fitts to the extreme, buttons a pixel away from mouse cursor).

This case was so revolutionary that it led to a change in the mouse hardware itself (from 1 to 2 buttons). Later attempts were made to add in the mouse also some third buttons and the scroll wheel; but they turned out not to work well.

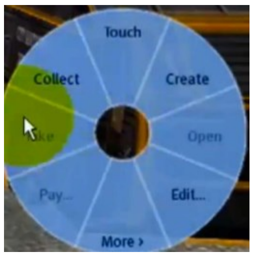
In the Windows world at some point Vista was born. It was not as well liked, because it focused on security and any potentially dangerous operation required authorization from the user. This resulted in:

- appearance of pop-ups
- thinking about what to do
- pressing on multiple buttons to complete actions: too much computational effort, terrible usability

So, the pop-up menus are a relatively recent invention derived by Fitts Law. Here we can compare how easier in the long run are OSX products over Windows ones (which derived the many mockup ads made by Apple over Windows, particularly on Windows Vista; a collection of them here).

Because of Fitts, many types of menus were experimented and created:

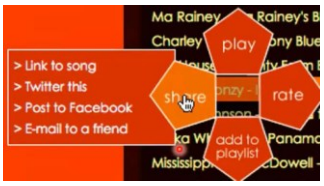
- *pie menus*, popularized by things like Second Life (sorta-metaverse of 20 years ago, feel old talking about this) with the need of having visible menus
  - This is more liked by users rather than a rectangular menu (the classical one), which is also easier
  - This allows to make multiple actions with less mouse movements
  - They work badly with too many elements (slices too little)



- *fan menus* (hybrid between pie menu and magic spots – so, between borders and corners), which are activated on/near the borders or on the corners (45°/180°).

Linear menus work better when there are many elements or when their descriptions require much text.

We can combine both, though, in a good way. A good example is the *songza* website here on the side, which users loved no matter the quite original design.

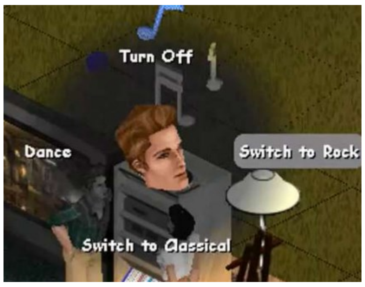


The pie menus were long present thanks to videogames, and these can be found in many different forms and shapes, according to the specific context.

Again, games the first to introduce another type of menu:

- the so-called *radial menus*, parents of the pie ones, which don't require a special key or a mouse
  - they give only buttons labeled visually with actions that don't require reading but are created in a way which makes the pointer interaction easier, training via muscle memory

- great example: the Sacrifice videogame, which will be executed in real time by our heroic professor (here on figure, The Sims)



7 ADVERTISEMENTS (LECTURES 13-14)

Advertisement is a crucial topic is part of the classic business model, in which you offer free service, reach a good number of users losing money, then making income with ads. We then focus on advertisement within the site, talking how to externally publicize our site/product/brand.

Take a wild guess: *users hate ads* and roughly *only 0.4% of users will click* advertisements. We need to have:

- *good positioning* (here we list the best places)
  - the left column
  - the top of the page
  - the right column

Other useful things to note:

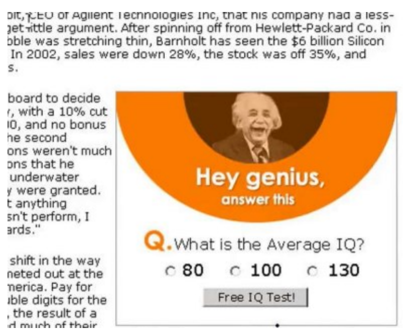
- Put ads *nearby interesting content*, so ads are seen more
- Size influences visibility: don't make banner so big they cover other things; find a compromise in their dimensions

We also need to have:

- *more attractive* ads, which are *beautiful* and *attractive*
- the user perspective is the only one that matters

The top then of disgraces in ads are (the percentages mean how many users hate in percentage, from decreasing to increasing):

10) automatically-playing ads (79%) and moving within a box (79%)
9) ads that blink (87%)
8) ads occupy most of the page (90%)
7) ads that move along the screen (92%)
6) ads that don't say what they are about – gambling clicks (92%)
5) ads that cover something you were trying to read (93%)
4) ads that don't have a clear way to be taken away/get rid of it (e.g. tiny X to close that appear after a while, etc.) (93%)
3) ads that try to make you click over (94%)
2) ads that load slowly (94%)
1) pop-up (95%)



Other tricks are commonly used in advertising, such as:

- sexualizing the content to make it more appealing and provocative, just shady ways to attract user attention (which they work, sad spoiler)
- we are naturally more attracted towards beautiful people (judging the book only by the cover in practice, it's a mad world)
- use shiny colors and special effects like in "classical advertising"

7.1 Ads Content, Presentation and Behavioral Advertising

People usually try to avoid annoyances and of course ads are just like that:

- we use something called *the zapping effect*, in which we use *defensive algorithmics* to avoid ads (which are annoyances between us and our goals)
    - as adults, we know ads are bad and we avoid it; young people are more attracted because they don't have those filters yet and zapping effect protects adults this way
- ads usually to avoid this try to mix colorful layout with interesting form factors and content, other times trying to mix how content is presented
- examples: Susan Boyle at Britain's Got Talent where bias is faulty or the Big Brother TV Show using normal people instead of known people to put on display.

To avoid the "zapping effect", we try to leverage the opposite: find *paradoxical* content (like using colors not too sharp and people not too beautiful, but are good and authentic); this way, we try *confounding* the user, perplexing and engaging with him, *mixing* content and advertisement.

- A negative example is the *border effect*, where you put the border to separate the ad from content: this further drives people away, it's a clear graphical separation

- A positive example is *blending*, where there are no borders and ads clearly become part of a website
    - An example of good integration are *web games*, which easily capture user attention and they are effective for advertisement (if properly used, like the Colgate flash game)
    - A not so good example is Google, which put ads inside the search results, clearly separating content from the advertisements (because they were fined to do so avoiding monopoly)
- This way, the algorithmics don't work because we don't feel those as ads, but as content

<u>Text</u> is a fundamental instrument for *blending*, so content and ads are integrated naturally towards the main user attention source.

- Other effects if you want to use images effectively is to create advertising images in which people (in the advertising image) look at the object of interest
    - Also to show full-size people, since first the face is analyzed and then the genitalia (this happens naturally), and it is therefore useful to put the advertisement there

Besides how ads are presented, there's also <u>content</u> inside advertisement, which implies being careful to context:

- there is the infamous *distraction effect* where ads are too much detached from the context
- timers diminish up to 40% and the return rate can diminish up to 80% when this happens

This is a big problem, but also a *big opportunity*. People get angry because the informative content of the ad is not consistent with their goal.

What happens instead if we provide informative content helpful for users goals? This is the so-called <u>behavioral advertising</u>, ads that try to offer relevant/helpful content to the user, which had a massive boom and made Web/smartphones like ads a major profitable space.

This works if data on users is collected each time, and we try to create some content which interests the user and adapts to their behaviour.

- This is more effective of at least x10 and can arrive to x100 times and beyond.
- This can also increase both the timers and return rate, in case.
- All sites have a great number of trackers employed in them and this is the main reason

Behavioral advertising also has a beneficial effect on the site: not only does it not suffer the negative effects of "distracting advertisement," it even reverses them in some cases, leading to an increase in timers and desire to return.

In any case, ads should be integrated such that:

- the size doesn't cover content
- they are placed inside strategic points so the user might click on them
- avoid using bright colors to separate ads from content
- just by looking at them, they give you immediately their idea and their context
- it doesn't have to be slow on loading and shouldn't force the user to click on it
- if ads are present, they should be easily closable, so the user won't be annoyed even more

8 SEARCH (FROM LECTURE 14 TO 17)

In sites it's important to find the right information; site size matters because there are two critical thresholds:

- beyond 100 pages, it is necessary to offer some *search tool*
- beyond 1000 pages, it is essential to offer a *good* search tool

It's really important:

- if search is available, users tend to use it around *99/100%*
    - so, usually users will arrive on site via deep linking
    - this highlights how a user can be catapulted to any web page on the site, and to continue browsing he can either return to the home page or can use the search box
- If there is no search and the site is above the first threshold (100 pages)
    - there will be more unsatisfaction (-20% return rate, -70% on timers, because the flow of navigation of a website is forced on the user).

People are used to low-effort:

- with so much information, people are used to search engines as a way of ultra-quick navigation and people pretend the same from a website.

When a user arrives with deep linking, on average 60% of users uses the search functionality (if available) and 40% normal navigation (via links).

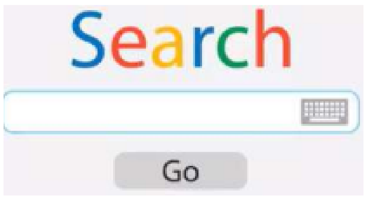
- For example, we can use the *site* command to localize pages within a single website (which works like *keyword site:sito.com*, like *shoes site:zappos.com*), all at zero cost.

Large-scale search doesn't work so well on local scale:

- there are advantages in development (zero-cost), but the algorithmics employed do not find well fine-grained information within a website and don't give good responses
- moreover, if the user already arrived via a search engine and didn't find what he needed, it means the engine itself was not precise with the site information.

As a matter of fact, a search engine *truncates* information and doesn't index anything. So, it's important to offer search according *to what the users want*.

There are various ways to offer search functionality, but users prefer the most important one (the Google-like):



- a *textual box* where to write what they want
- a *search button*

On the search button:

- People expect a button with label "Search" (it's useful to localize the label according to the language used in the context).
- As a matter of habit derived from mobile devices: use the search icon, alone (or safer) when combined with "search".

Examples on this:

- The bank site with a search function non-standard completely, hidden within the text and the color; one could think the dropdown menu could be related to search, while of course it is not
- The Dime website, where actually there isn't a search functionality, while in reality people thought the dropdown menu was the search box and of course it is not
- The Nordstrom first website, with a non-standard button/functionality as a deep link compressed within the website; the second version actually put it top-left in position
- One of the Disney websites version, where the search is actually a link, and this confuses users
- The Disneyland website, where the search box with a "Search" label and "Go" button, completely non-standard; similarly Disney Movie Finder, with label "Find it fast" and "Go"

Another important thing to remember: *less is more*.

- An example is the BASF website, where the search is too much complicated to be comprised to a simple search box, so there is the Advanced Search and filter to further refine the research.
- Another example is the legacy Google search, where the "Advanced Search" button disappeared in favor of a simpler and more effective search box, given it is easier for users of all kinds.

Always remember to not exaggerate: the Tafazzi principle, thank me later – not self-inflicting damage, trying to combine multiple options for searching, just overloading the user.

So, remember "less is more": better having something simple and working if possible.

- Examples of this are the Carlo Cattaneo or Iusreporter websites when there is an "in-site" search option as search and also an external one over Google search
  - this is Tafazzi-like, users may be pushed away from website, making users lose time and confused. Low-cost solutions only worsen the solution

8.1 Types of Search

Beside classic search, we can offer the so-called constrained search, which works best when used *in addition* to classic search (like you see in Zappos site: a search box and in case some filters to be used).

- This is efficient and much appreciated by users
- As a con, there's the need to pay attention on how it is implemented, because there is no "reference model" in search engines, given it is not part of classical search engines

There are usually two ways of research:

- static, when the user has set already all the parameters and has to press a specific button to start the search
  - the problem is the button label is non-standard; if we use "Search", the user may try to write text and/or looks for a textual box
  - if user is used to dynamic search, if first parameter is set he can think the search does not work, being confused

- dynamic, as soon as a user fills a field, the search starts
    - many sites offer it, but that requires some seconds (because it triggers the search every time)
    - many users wait for the result, until they realized a button has to be pressed (more frustration and time lost) – so, users may get tired of waiting for the query result

The constrained search seems to be the better one, but its usage depends on the *number of constraints*:

- when the number is 1, dynamic search is superior
    - because you limit the number of suggestions to one field and with this approach computational cost is definitely reduced
- when they are many, a new search starts every time a parameter is input
    - this increases waiting for each constraint and overloads the server
- If the user wanted to do *incremental search*, this is perfect

- This is not always the case:
    - Sometimes the user already knows a few parameters and is forced to do *separate searches* rather than one search, which lead to more frustration, lost time and diminished timers.
- *With many* parameters, the *less risky* choice *is* definitely *static search*.
    - This is the approach a version of the Zappos site tried to take: combine a static search with multiple filters to help research, putting the search box clearly visible.

There is also the hybrid solution:

- static search, with automatic launch when *all* the parameters have been filled
- this is not recommended because it's not good for the user, it distracts him from the goal giving delay in results, increasing user efforts (has to put all parameters) and we risk losing him

Let's see now how to deal with *input/output* of a search:

- starting from the output, this should be a list of items presenting results compactfully trying to follow user habits, behaving almost like search engines.

Since we deal with constrained/parametric search, it's helpful to allow users to *sort* on any object parameter and this is the bidirectional *sort*.

- Example: put a price ordering high-to-low and low-to-high, not only one and forcing the user to one option but give everything to him in case.

When *there are zero results* for a search:

1) give zero results
    a. do not use strategy (1) - it confuses users, thinking search is not working
2) say there were zero results
    a. this is preferable, but it can be improved even further

At this point, one could say: who cares about zero results?

- The sum of small details make up the bigger picture of website
- With simple textual search, the zero results scenario is not so uncommon
- It's important to handle it gracefully according to what the site offers and to the specific situation

Let’s see the third strategy (from the order just given above):

3) the *404 effect*, where we consider *dangling links* (either links that do not exist inside a website or just some gibberish, so something like site.com/ahahah) and you have to tell the user the context does not exist.
    a. The general advice is: don’t use technical jargon without thinking of the implications of dealing with normal users.
    b. Also, give users links to go back home or to search for other pages, in a simple language, understandable and even emotionally designed for him

Here we will list a series of examples from slides:

- Misleading and technical ones: The Guardian/La Repubblica/UniPD/Gucci
    - Using Error 404 and URL terms, definitely not understandable by people not in the context
- Too easy example: Euronics, misleading user with too easy terms
    - “We momentarily lost the page”: it doesn’t mean anything in reality
- The blank page with error code on Dainese italian version: atrocious but funny
- The MIT/BBC/CNN websites offering a web search to find other meaningful pages helping the user
- The Zappos site is reasonable, because it reasons preemptively: when putting random/wrong words, it tries to correct or tries to launch a search similar to the input parameters to help the user
- Funny ones like the Disney ones, with the Monster Inc./Inside Out 404 error
- Have fun with some present at http://www.b3ta.com/404

The *presentation of search results* can be:
- *linear*
    - displayed sequentially, one after the other, guiding the user towards information naturally
- *in grid*
    - it focuses vision on the center and allows for shown elements to be more compact
    - content tends to lose relevance, and this confuses the user, making him lose focus and time
    - too much choice/too much effort in micronavigation, so details which can worsen up the experience

The size of the search box depends on various factors, like *context* and so the kind of search, but we also need to consider the evolution that search has had with users (before = keywords/now = long sentences). The searches evolved from only single keywords up to 60/70 characters long and grow more and more; the size should consider all of these factors, which gives benefits even with a small increase.

The suggested size for the search box is *30* characters: with this, we can accommodate 90% of the queries, given this is the average length for most queries (for reference – Google/64, Yahoo/36, Bing/50):
- If the box is too small, users start to have discomfort increasing proportionally with how much text is not visible in the search box because typed characters disappear
    - +5% initial penalty, +1% for each extra character
- To avoid this, search engines have search boxes dynamically adjusting to accommodate dimension of the textual query input, not forcing scrolling
- To avoid penalty, users tend to use smaller queries, giving less precise queries to that and this gets users worse results

Let’s see some examples on this:

- a good example are the multiple redesigns of the Heart.org website, which redesigned its search bar to a much longer one since the first versions, making it more suitable for complex searches
- a not so good example is the Disney, with the .it version without search box, the USA one with portions of images not clickable, both with portions of layout too big to fit and confusing
- good example is the Nordstrom search bar, above 30 characters, giving suggestions to search
- the BASF website has a good search bar, allowing a big search box opening up when clicking the search lens icon
- the math.unipd.it website, not allowing a search box, also the text is only in Italian non translatable (let’s not talk about the baffling choice of reaching the site only if putting the “www” manually to the URL without any redirect automatically)
- the Amazon website, which offers a suboptimal search, with 50 characters length on average

There are some solutions:

- allow space for the search box
- make the box *dynamic*, so it grows when the user types in it
    - so, when not used, small box: when clicked, bigger box to search
    - consider it covers other parts of page, no need to have reserved space

8.2 Searches 2.0

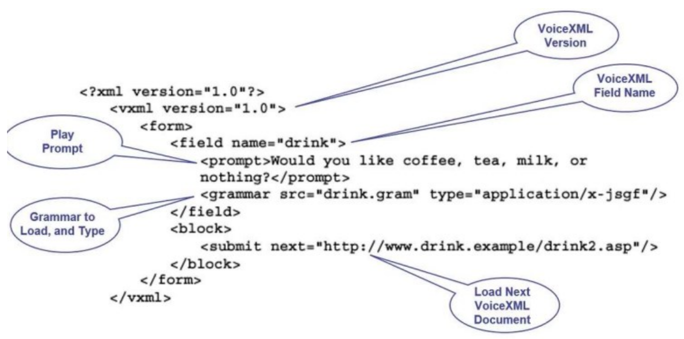
There’s not only the classic search though, given we also have different modalities to look for information:

- *voice search*, using audio as a way to look for information
- *conversational interfaces*, like ChatGPT, with generative prompts and answers given in real-time, specifically on contexts
    - we have *custom GPTs,* which are changing and developing at the same time, even though they are far behind the rest of training (new GPTs are created only by paying)
    - in this case, some customized versions which can be easily created to localize
    - this can be Tafazzi-like because we use other websites instead of our own to index content and chatbots to not always give good results
        - they can create mixed/fake content without any control (no clue what they are doing, only “intelligent copiers” mixing stuff at random
    - in text, chatbots are stronger, because they generate content in a regular and scripted way, which can actually be leveraged for advantages when used for limited purposes
    - they are young in conception for the masses, so potential for improvement is present

There are other alternatives:

- VoiceXML, which is a family of languages used for developing in website development for audio and voice (*natural language*) response interactions (and in general, for superior interaction).

  This is not used only for audio though. The side figure is an example of how it appears.
  It’s like writing a webpage in a slightly more powerful way.

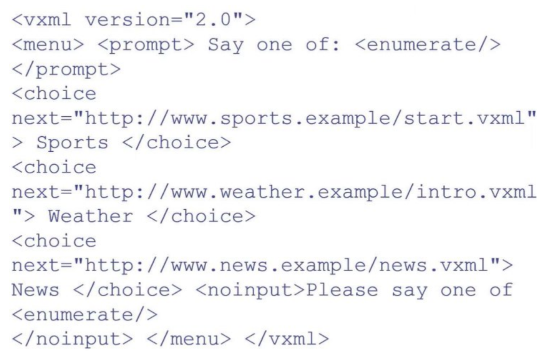


```
<?xml version="1.0"?>
   <vxml version="1.0">
      <form>
         <field name="drink">
            <prompt>Would you like coffee, tea, milk, or
            nothing?</prompt>
            <grammar src="drink.gram" type="application/x-jsgf"/>
         </field>
         <block>
            <submit next="http://www.drink.example/drink2.asp"/>
         </block>
      </form>
   </vxml>
```

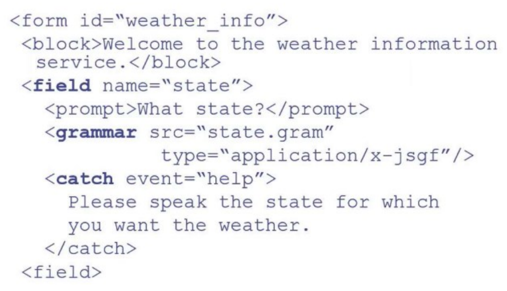
There are other types for VoiceXML:

- *SSML (Speech Synthesis Markup Language)*, which passes enriched text to voice, so it gives the ability to give the accents and all the enrichments tone to the computer voice, by means of special characters
- *PLS (Pronunciation Lexicon Specification)*, which specifies word pronunciation
- *CCXML (Call Control XML)*, which handles a voice interaction flow (like a phone call, voice web service, digital assistant and so on) managing human-machine interaction
- *SCXML (State Chart XML)*, general language for defining execution environments based on state-machines (using statecharts, simple and more powerful than UML), more general than CCXML
- *SRGS (Speech Recognition Grammar Specification)*, which defines the user grammar, so providing context to the audio channel
    - The big problem of every vocal application is the <u>context</u>
        - voice assistants only recognize text but do not understand the meaning
        - we can constrain the grammar, avoiding misinterpretations
- *SISR (Semantic Interpretation for Speech Recognition)*, which gives a meaning (corresponding actions) to what the user said
    - This one, like shown in right figure below, interprets commands with a *.gram* file and vocal recognition happens via an external engine, not VoiceXML/in the device itself

The following are examples of constrained commands, with grammar or fields for example the context of a menu of commands and meteo grammar, interacting accurately with the audio layer:



```
<vxml version="2.0">
<menu> <prompt> Say one of: <enumerate/>
</prompt>
<choice
next="http://www.sports.example/start.vxml"
> Sports </choice>
<choice
next="http://www.weather.example/intro.vxml
"> Weather </choice>
<choice
next="http://www.news.example/news.vxml">
News </choice> <noinput>Please say one of
<enumerate/>
</noinput> </menu> </vxml>
```



```
<form id="weather_info">
 <block>Welcome to the weather information
  service.</block>
 <field name="state">
   <prompt>What state?</prompt>
   <grammar src="state.gram"
            type="application/x-jsgf"/>
   <catch event="help">
     Please speak the state for which
     you want the weather.
   </catch>
 <field>
```

VoiceXML is fully integrated with JavaScript to deduce/interpret actions.

Overtime, voice assistants were improved trying to seem more context-based and human-like (many times even cartoonish at that). Examples are: Loquendo (ah, the memories), Lulu, Hellena, Umanify, Sonia.

- These assistants often try to do too many things without excelling in any specific area. They may appeal visually to certain users, but in reality, they struggle to perform well or be user-friendly.
- Usage of assistants like the ones quoted leads to an average -42% on user satisfaction level
- Remember "less is more": users have corresponding *expectations* to voice assistants and want a good interaction (given it is human-like, they expect something more precise)

The lesson is: don't use digital puppets but instead create a professional product, like voice assistants, which are neutral ("robotic"), don't get users angry, leaving to the user the imagination of the possible assistant appearance, hence not leaving particular illusions or expectations.

- Another case, actually useful this time around, was the "Anna" case, used in Ikea
    - It was a textual assistant which used a "cartoon" interface, with a language engine (based on VoiceXML) quite powerful and complete
    - It was optional, giving the users choice and was very much loved by the users
    - It was redesigned multiple times to counteract sexual interactions by users

Other historical examples of interactions are:

- 1966 Elisa, which was an experimental psychotherapist chatbot found here
- From 1975 the textual adventure games genre, which had to set their own gameplay to receive a specific syntax from the user (usually, actions/commands/something specific)
- Some presented game examples are Grim Fandango (play it please, it's a reminder also for myself) and The Inner World (indie game which won several awards); also, we get a glimpse of the Steam username of our precious teacher here (left as bonus, for you to investigate)
  - We can see examples of how videogames influenced our everyday UIs, from the multiple camera angles and textual boxes of interaction (here) up to so simple interface it does not require a manual, just clicks on the environment (here) – yes, I am passionate about this

Something that initially appears unnatural can be improved by *introducing a certain level of noise* or chaos, which ultimately makes it appear more natural.

- The assistants are unnatural to look at because they are missing some chaos components (moving robotically), and this leads to uneasiness by the user
- Make characters move makes more pleasant assistants by an order of magnitude, even adding for example nonvoluntary movements, called *fractal noises*
- Noise in fact is the key: you create a curve that starts with a large amplitude and little vitality, and you create all chunks by continuing to increase the vitality and decreasing the amplitude.
  - Eventually they overlap and you create an effect of realism, for example waves, a terrain or the movement of grass (some examples in slides about 3D modeling)
- To generate noise, given its natural fractal recursive nature, just reduce amplitude and higher the frequency

Some examples of these concepts in practice:

- Google Glasses, which tried to leverage on the virtual/augmented reality aspect:
  - they had an input that was too invasive and stressful and as output the same principle of popups in real life, interrupting users actions with something negative and annoying
  - they failed because of these reasons and too pricy
- Wearable devices, which as of now don't completely work, because they do not integrate that well in our reality
- Smartwatches, which have more potential compared to Google Glasses, accompany a fairly simple human interaction with "distant-enough" integration sufficient to be considered helpful

9 VISIBILITY (FROM LECTURE 18 TO 20)

For the success of a site, it has to be visible *outside*, usually via search engines. This needs to be made, given the websites, if they are outside the Top X websites, will never get reached. To get ranked highly from a search engine, it's important to leverage the appearance inside SERP (Search Engine Results Page).

The top ten of websites absorbs *more than 95%* of all clicks (because the users will never do the computational effort of coming to us). Consider some facts:

- the first position alone takes 51% of all clicks
- the second position takes 16% of all clicks
- the third and fourth 6%
- the fifth 5%
- the sixth 4%
- the seventh 2%
- the eighth and ninth 1%

In position number 10, there is the so-called "Malabrocca effect" (Black Jersey Effect), related to Luigi Malabrocca, an accomplished cyclist, winning the Italian National Cyclo-cross Championships and other races. He left a lasting legacy as a cult hero of the Giro d'Italia, despite often finishing last in the race.

This effect states that at the last place in the top ten, the click-rate *doubles* compared to the previous positions: 2%. We can see from the attention zones; usually, only the first two links are considered as hot zones. Mixing images to text doesn't alter the properties of the "textual" top ten.

9.1 SERP/SEO AND SPAMDEX

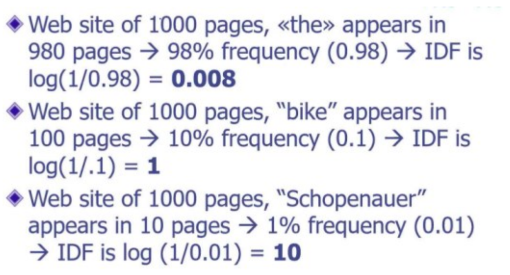
One way to climb the rankings on the web and getting the highest SERP pages is to use the SPAMDEX (SPAM InDEX), also called SEO/SEP (Search Engine Optimization/Persuasion). An amazing paper explaining these terms (suggested reading: trust me) here.

Currently, information is given by the *textual* component of a page, plus an *hypertextual* component (which will be the PageRank part).

- An interesting example of this was GroovyMovies, which is an empty site containing nothing but a single static image which seemed like a page with a search bar looking for movies but was nothing
- At one moment in time, it was number one site when looking for "movies" on Google because it exploited SEO fairly well

We focus here on the *textual part*, called *TFIDF* (or *TF-IDF*), which stands for *T*erm *F*requency-*I*nverse *D*ocument *F*requency, which gives a measure of *how important a word is for the page*:

- *TF = Term Frequency*, which is how much the word appears in the page
    - If we were to use TF only, many frequently-used words would have a very high TF, hence unbalancing the stats
    - Think for example the word "the"; it would be the most important word in a page
    - To solve this, the following component comes to help



- Web site of 1000 pages, «the» appears in 980 pages → 98% frequency (0.98) → IDF is log(1/0.98) = **0.008**
- Web site of 1000 pages, "bike" appears in 100 pages → 10% frequency (0.1) → IDF is log(1/.1) = **1**
- Web site of 1000 pages, "Schopenauer" appears in 10 pages → 1% frequency (0.01) → IDF is log (1/0.01) = **10**

- *IDF = Inverse Document Frequency*, which is the *inverse* of the frequency of the word *within* the set of documents, scaled logarithmically

In the images above, when a term has a low score, it's like a linking word (something like "the"), in other cases, considering only a selected number of words appearing with a fairly low score (but weighted more importantly), trust effect kicks in and a word is valued much more.

Hence, this measurement is calculated as $TFIDF = TF * IDF$.

- To raise the TF, a keyword is used
- If this keyword is inserted on too many pages the IDF comes into play which cuts the score

When a search engine receives a *query* made by a word w, it takes all the pages where w appears and then computes their TFIDF. If the query has more than one word, we can just sum the different TFIDFs computed for each word.

- So, if we want to raise the textual score of a page for a word w, we have to be careful because raising too much its TFIDF automatically causes lowering other words' TFIDF.
- The strategy is to focus on a set of words, called *champions*, and raise their TFIDF, lowering the others
- We will have to carefully *choose a set of keywords* and appropriately *manage* them in the website, for example also understanding how links are using those specific terms on a global scale

There are various ways to increase positioning and TFIDF of such words, like:

- <u>body spam</u>, inserting words into the body of an HTML page
  - simple and effective
  - as a drawback, we are touching the actual content of the page and this can become heavy for the user, because it disadvantages reading for the user
  - if asked in exam, then talk about also hiding techniques and cloaking (thank me later)

- <u>title spam</u>, inserting the keyword inside the *title* tag of the page
  - the content of the page is touched way less because users will never see it (doesn't get showed in a page)

- <u>meta tag spam</u>, using the *meta* command, command built to describe a page
  - so, consider *<meta name="keywords" content="…">*
  - no user side visible content is touched
  - as a drawback, this is abused, so there is very low score by current search engines, given everybody does this and metatags are not so important anymore

- <u>anchor text spam</u>, a part of body spam but considered apart usually
  - we insert words in the anchor text *<a>… </a>*
  - so, basically it's "inside of a link"
  - they give special scores, breaking the purely textual model and the keywords are also added to the *target* page of the link, with *less limitations* with respect to TFIDF
    - it means that we count score not only in our page but also target ones (bonus)
  - as a con, they can be penalized by search engines (penalty filters), since anchors are supposed to describe the page they redirect to and can touch the content of pages

- URL spam, technique to insert keywords directly into the web address of the page
    - very powerful technique but requires editing the URL
    - it is used mostly in combination with other spam techniques
    - consider search engines also analyze the addresses, giving *bonuses* similar to the anchor text spam, given it does not touch page content
    - it may be considered spam if done multiple times, decreasing SEO rankings and TFIDF

So far, we saw positioning in term spamming, but also matters *the content*.

Consider the following "starter kit" of where to insert keywords:

- repetition, revolving around repeating the same keyword
    - if we exaggerate, we are penalized on the TFIDF (balancing) with SEO results (it's easy being considered as spam that way)
    - this is easy to spot from search engines

- dumping, inserting terms rarely used which can rank higher, given they are rare keywords and can be ranked relatively high, even when they are not related or useful to the page content
    - lots of terms rarely used with low TFIDF, but given they are not generally present, their score will be very high
    - as a con, we are inserting terms not related to page content, which can make difficult keeping the users, even if it tries to facilitate access to the site
        - this happens because the page may be different to what they were looking for, losing trust from users

- weaving, taking pieces of other websites and then modifying them by inserting our keywords (usually in a random way), giving more score overall
    - as a con, we are affecting the page content, so try not to put unrelated content to avoid users going away
    - if used improperly, this can bring in unrelated keywords, giving irrelevant content and potentially a form of keyword stuffing, overusing the same rare keywords

- stitching, copying and pasting fragments of other web pages, uniting them into a single page (for example taking words out of order and giving relevant content to the user)
    - this works as an automatic way to create "interesting" content to populate a site, giving *global bonuses* over the quantity of *information* a site offers
    - for example, from a user response on a forum we can develop a new article
    - using different topics gives a global bonus, and the search engines struggle to understand if that is copied content, hence getting more value for the site
    - this is usually more used than weaving, considering it is harder to spot

- broadening, insert keywords not only selected, but also opportune synonyms or related keywords/phrases
    - this is useful to better cover user queries but also because many search engines also use measures of *similarity* among keywords in order to give added bonuses
    - e.g. If we search for Disney on the Web and within our Web page is contained the word Winnie the Pooh, the search engine will give our site a bonus (and we have not written Disney in the content

Beyond these techniques, there is the keyword choice, which is another fundamental problem. There are many methods, like Google Ads Keyword Planner/keywordtool.io/Google Trends:

- this leverages discovery not only based on paid advertisements over results
- there are alternatives and they work using autocompletion, calculating suggestions precisely
- basically, they look for advised words and the percentages for those words from the users

The *problem of text spamming* is that is *generally changes the content of the pages*, with pages that get "powered up" and so users are attracted from it, getting higher results inside search engines, but content gets corrupted or not so relevant and so the users get angry.

Now we consider usage of a series of techniques called hiding, which hide the "trash" inserted with spamming.

- content hiding
    - using the style of page to make content not appear in pages
    - for example, put text spam with same color of background
    - put very small images (something like 1x1px) to trick "invisible" user clicks

- use redirection/302 technique, which consists in putting keywords empowering a page (spam page) making the user redirect to another one ("clean" page); search engines will stop on the first one
    - consider *<meta http-equiv="refresh" content="0; url=pippo.html">*
    - this is not so effective, given search engines will recognize the *meta* redirecting
    - if one uses Javascript, *meta* tag will be added dynamically and the search engines, to avoid consuming resources, will not execute the code (or simply obfuscate the code a bit)
    - example: *<script language="javascript"><!--location.replace("pippo.html")--></script>*

The most powerful technique is cloaking, technique used where a website shows two versions of the same site: one for the bot and one for the user. Some features:

- We recognize user and bots given the mandatory recognition banner "Are you a bot" from the search engine – you can see that inside search results if you use VPNs for instance
- This way, if a bot visits the page, we present a special page to that; if that is the user, we offer a completely different page
- There is truly little chance of getting caught, given only humans can spot the difference
- The technique is powerful, but the penalty is extremely high, given Google will ban our site for a long time if that happens

9.2 PageRank Algorithm: Versions and Problems

Let's now consider the hypertextual component, complementary to the textual one, which contributes with a good deal of points, derived from the *network topology*.

We consider the PageRank algorithm, which works this way: the more links there are, the higher the rank of a page. This is based on the network topology, so considers how many links (pages) are connected to one.

Specifically, for each page π_v we use this formula:

$$\pi_v = \sum_{(w,v)\in E} \frac{\pi_w}{d_w}$$

where d_w is the number of outgoing links, (w, v) a link from w to v.

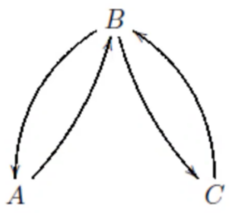
Like water reservoirs, there is a balance: if a page w has a link towards another page, the page gets a fraction $\frac{1}{d_w}$ of pagerank from this one, having as constraint the total sum amounting to 1:

$$\sum \pi_v = 1$$

The calculations can be done iteratively, matrix-like, etc. We don't care about how it is done, search engines do that for us.

- Reformulating, we talk about *Markov chains*, which is also called "drunkard's walk": you put a drunk inside of a page and then the algorithm will go randomly clicking a link
- We count *how many times this passes clicking our link*, so the pagerank is the probability of entering a page instead of other ones

The equivalent is the "surfing monkey", which navigates the page at random completely and the probability of the monkey entering a page determines its score. For example, given the following graph:



- either we calculate *iteratively* or reason by symmetry
- $A = C$ has double flow than A and C so (if the total liquid is 1)
- the pagerank of A and C is $\frac{1}{4}$

There are more general problems however, which crawl data at random pages to get data on them, considering for instance:

- *spider traps*, where the spider bot can get stuck inside a website continuously redirecting pages in an infinite loop (consider for instance an online calendar)

- *islands*, in which there are some networks pieces isolated from the others, in which there are small indipendent pieces of the web which are unreachable
    - this happens because pagerank value can be either zero or infinite and it's hard to decide which page is the most important one
    - this stratagem was used by Microsoft to have higher pagerank

Considering the problem above, a new solution (PageRank 2.0) was crafted using the same formula from before, but using <u>teleportation</u>, considering the normal pagerank plus a teleportation factor, which is constant (so, it assigns score in a more democratic way following a specific probability)

$$\pi_v = (1 - \epsilon) * \sum_{(w,v) \in E} \frac{\pi_w}{d_w} + \frac{\epsilon}{N}$$

In practice, randomness increased, so with a certain probability (starting from 0=original pagerank up to 1=score equal for everybody), the "monkey" can get teleported into another page, with going in loop and preventing errors in getting PageRank results (more "democracy" is given thanks to ϵ and $1 - \epsilon$ represent teleport – the monkey "rolls some dice" and it *may* happen it can get unstuck)

So, if teleportation is not present we consider the original pagerank, otherwise we take the modified one.

This way, thanks to the teleporting factor, we will solve spider traps and the island: this factor is towards 100% and the probability score for each pages becomes equal for all of them, instead if it is towards zero, the higher the pagerank gets to what was before.

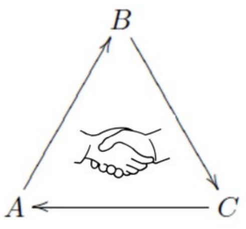
Another way is summing all of the page rank from 0 to 1 thanks to an integral (basically, an average of all possible choices – same computational cost as first version), which is called *Totalrank*:

$$T = \int_0^1 r(\alpha)d\alpha$$

Consider the full cooperative case between links: in this case, $A = B = C = \frac{1}{3}$.

When somebody breaks the cooperation, for example C, the pagerank of B and $C = 1/2$, while $A = 0$.

You can see that even with one link, situation is unbalanced, and this is dangerous.



This is the "*dead ends"* problem, in which there is a structure of websites pointing to a site which makes no links coming out of it, keeping that site score increasing without limits. Search engines penalize those.

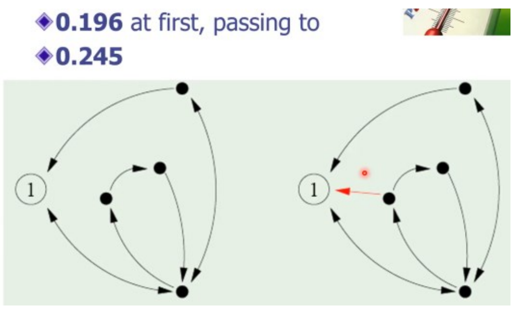
- To avoid problems, Google artificially puts outside links to avoid this problem and so a website is better to keep inside-site links – this way changing web structure and the way to see links
- The modern websites solution is to calculate the importance in every site and webpage and not on the real web – this complicates SEO and the calculations which need to be done to promote a site

9.3 Inlinks, Outlinks, Alliances and Types

There are two fundamental factors which influence the pagerank using teleport positively, getting higher:

- inlinks (links bringing to our site)
- outlinks (links bringing outside, of course)

Let's start talking about <u>inlinks</u>, and consider this inlinks example for instance:


◆**0.196** at first, passing to
◆**0.245**

Over inlinks, to increment the pagerank of a page, we should have it aimed towards the highest number of external pages.

Let's consider some techniques on this:

- *infiltration*, so one "infiltrates" in various sites and tries to insert links to our site
    - for instance inside directories, blogs, wikis, comment sections, etc.

- *honey pot*, creating "yummy" content (useful/good one), naturally receive incoming links
    - this is the right way to do it to increase pagerank (ethically)
    - this is doable by smart paste & copy from content of other sites
        - this is the main con of this technique

- *link exchange*, basically "joining forces" with other websites to exchange links
    - has to be done precisely avoiding low-quality links or unnatural link patterns

- *resurrection*, so buy defunct websites with a pagerank high enough
    - Pagerank doesn't decay overtime and it remains linked with the domain, and also the domain age counts as bonus (may websites do this as a business), transporting the traffic of inlinks to other websites
    - This may lead to a lack of relevance of the bought domain and also can disorientate users

Consider the case of an empty site called "Million-Dollar Page" here, created by a student, selling a pixel for a dollar each.

The Million Dollar Homepage
1,000,000 pixels • $1 per pixel • Own a piece of internet history!
SOLD OUT!

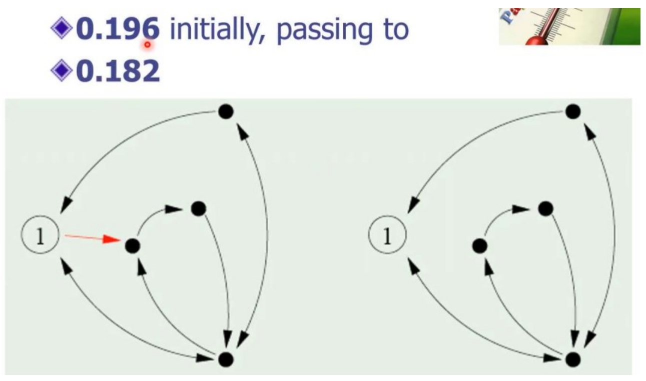
The student became a millionaire, considering the power this page had was the amount of links redirecting to this page and all the links inside acquired a webpage so high at a discount price.

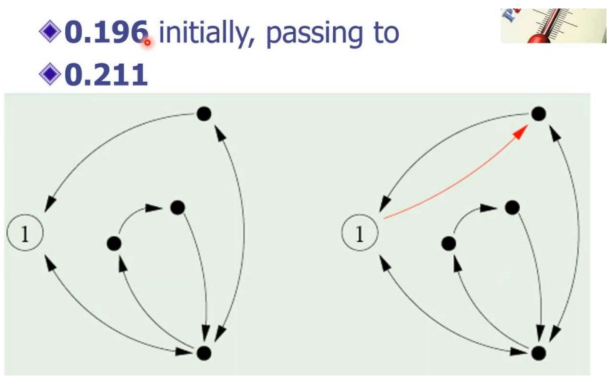
A similar idea was implemented from a football team, which wanted to keep footballer Lukasz Podolski and thought of rising up to a million dollar with the same idea (it didn't have the same success).

Now, let us talk about <u>outlinks</u>, which reason by symmetry: we give points to others, not to us (pagerank gets lower/flow goes out/cannot augment flow with an outgoing channel).

- This way *solidity* is guaranteed, considering overtime properties were modified to have this feature
- Adding outlinks to a page doesn't cause an increase to the hypertextual score
    - so the spamdex can't just be *local* (and so, it makes efforts harder)

Consider, as before, two examples, this time about outlinks:





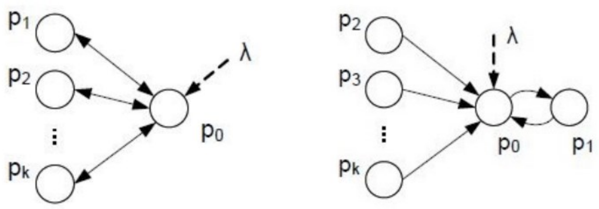
The rightmost example shows the outlink reasoning was wrong from the start, because:

- the pagerank *actually can increase* with outgoing links *with unforeseen consequences*
- this happens because of *the teleportation effect/factor* which does not make score decrease

There are different techniques to use outlinks/inlinks to increase the pagerank. Consider, for instance, the <u>spam farms</u>, which are structures created to increment the score and ranking.

- They consist of one or more pages under our control redirecting the flow of traffic towards other pages, called *target*, pointed by bidirectional links by other ones, called *empowering*.

They usually appear as follows, with the first example being the structure of a generic farm, while the second sacrifices reachability, because the search engines may not know those pages exist:

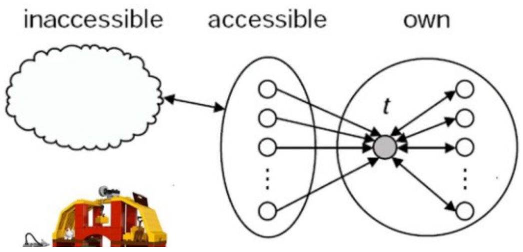


It has a good property:

- it uses *the least possible number of links*, while still keeping *reachability*
  - this means that if search engines spider can reach the target page via an empowering page, then all the spam farm gets reachable too
- it is also important in the case of only ingoing links (unidirectional, where you add a loop so that Google cannot insert outgoing links) – this makes for the inaccessible structure of following figure

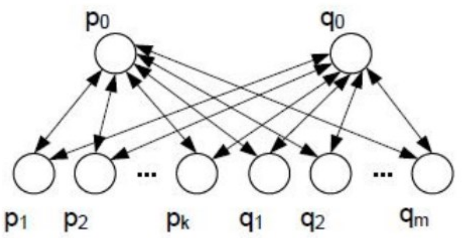
So an optimal structure can be constructed with some pages which are bidirectional linked to the target one. It maintains a very important property called reachability, where every page must be reachable by the search engine.

The optimal structure, when generalized, appears as follows:



Where there is no reachability, we need some "patching", for instance forming an alliance, joining our spam farm with someone else's. There can be different kinds of alliances and schemas to consider:

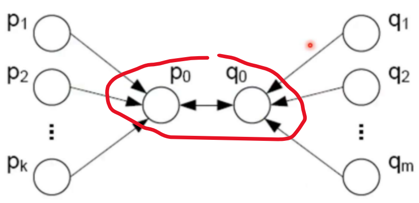
- deep alliances (pagerank = average of sites)
  - Here the target pages redistribute empowering pages flow of another site creating a *stable* pagerank (the *average* of two pageranks)
  - This provides target links to our pages but also to other target pages of the same alliance



Given 2 target pages owned by different entities the alliance consists in having 2 target which are double-linked with its pages and the allied pages. In this way the ranking score is the average of the 2 target pages.

This implies more robustness and stability.

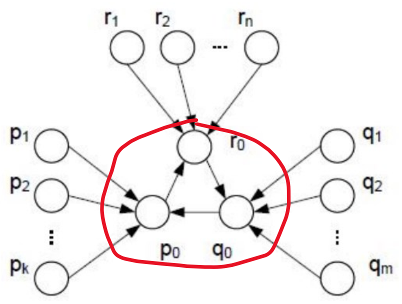
- superficial alliances (pagerank = *max* between sites)
  - They are minimalistic, and we use the minimal amount of links to join forces with others (usually, a single point of contact between two pages)
  - The target pages between two sites are connected via links and the pagerank obtained is more than the max between the two (bonus proportional to k and m).
  - This allows for a bonus on flow, and this is more powerful



Given 2 target pages owned by different entities the alliance consists in sharing all the unidirectional links of the 2 pages double-linking the 2 targets, creating a vortex of flow. In this way it is more maintainable because there is only 1 double link between the 2 targets. So the number of links are minimized.

The score for each target is an increment proportional to the other's allied pages and this implies that the score of each target $>$ the maximum of the original target score.

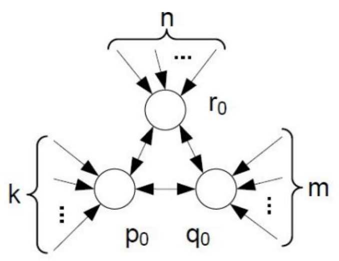
- ring alliance (flow of the pagerank)
    - They comprehend more than two websites and the target pages index in a circular path the next target page creating a sort of ring
    - If one of the internal links crashes, there are many link and balancing problems in the pagerank
    - This is better though because flow circulates broader and better



Given 3 or more target pages owned by different entities the alliance consists in sharing all the unidirectional links creating a unidirectional vortex of flow between the targets.

This ensures scalability and the score for each target is an increment proportional to the other's allied pages and this implies that the score of each target > the maximum of the original target score.

- complete core alliance (complete flow of pagerank)
    - This empowers ring spam farms considering a bidirectional structure and each target page as ingoing and outgoing links towards near pages
    - This is considered a good solution, but generally search engines try to avoid unbalanced situations as much as possible



Given 3 or more target pages owned by different entities the alliance consists in sharing all the unidirectional links creating a bidirectional vortex of flow between the targets.

This ensures scalability and the score for each target is an increment proportional to the other's allied pages and this implies that the score of each target $>$ the maximum of the original target score.

In this case, it doesn't look so difficult: just find the ring/complete core structures among different sites. In reality, ring and complete core are now the only two optimal structures, because there are other cores that give the same result.

- So, to create other cores, we just need to have a *strongly connected* graph among the target pages, this way creating a vortex, so from every page one can arrive to any other).
- Spam farms are usually penalized by search engines, given they try to elude pagerank measures and it's important to reason how many strongly connected graphs there can be.
    - They depend on the way alliance was formed and number of participants.

So, consider that if there are only a few ones, then the countermeasures will work well and in every sequence, the number of links grows enormously given the number of participants (N).

- An encyclopedia of sequences was studied, and nobody was able to quantify such sequences: number only grows a lot and there is no formula calculating it; the computational cost is very high
- The specific sequences was called A003030 (more here in case), which is really powerful and in 10 allied sites, the search engine is not able to identify alliances

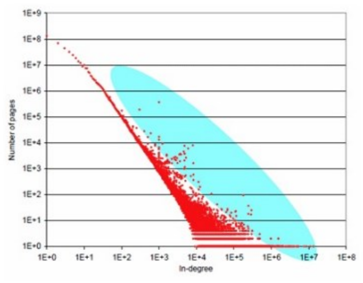
Alternative solutions are hence considered, to avoid search engines losing:

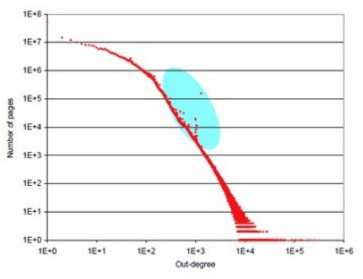
- *Get rid of the teleporting component*, calculating the *relative spam mass*, calculating the old pagerank and comparing it to the new one:
    - If it is too big, maybe something is wrong (too many "secondary" pages are offering contributions, hence the page was empowered), estimating the success rate of finding spam *between 95% to 100%*.
    - This serves as a threshold to avoid being too greedy in attracting flow.

- The *web shape structure* is papillon-like: the idea is to analyze the "shape" of a website and if it is too much different from the average, then something is wrong
    - This is averaged over a "slab" of incoming links from Google, from which it is possible to draw a norm and from there note the sites that deviate from it
    - This has low computational cost and it's efficient

Following here, the distribution of ingoing links (left) and outgoing links (right) compared to size as an exponential scale.

The outliers represented by the elliptic light-blue shapes are sites suspected to have been empowered:





Useful read and reference of image here.

These concepts are not present because they were not treated:

- Personalized PageRank
- Janus Graph
- Email with Janus Graph
- Social PageRank

10 NAMING (LECTURE 21)

In th web world, naming things the right way is fundamental and we will talk about this both on the *social* side and the *technical* side.

10.1 SOCIAL SIDE: CHOOSE THE RIGHT NAME

The *social* side means choosing a good web address, to recognize our site immediately. There are some good rules to follow, considering on average 10-20% name can impact, up to 50% as variance on the audience. So, this is a quite important thing to do.

1) Shorter names work much better than longer ones

2) Name has to be unique avoiding confusion with other names, sufficiently separatable from them
   a. For instance: never choose a plural when the singular is already taken

3) Take the .com domain, which on average has an impact greater than 4.5% (it's more trusted)
   a. Better than the national ones (.it, .fr, etc.)

4) It should be easy to memorize and to write

5) Better to choose existing words rather than creating new ones
   a. In any case, if new words or acronyms are used, the important factor is the ratio between the "standard" words and the new ones: the higher, the better
   b. Impact here range from +1.5% to -5%

6) Beware the sound of a word, being nice and harmonic
   a. Be careful about each specific tongue, in case ask native language people
   b. Names that start with a vowel work well (about +3.7%)
   c. Semi-vowels (r, j, y, w) work well (about +2.9%)
   d. Consonants like (f, v, s, z) work even better (about +3.3%)
   e. Consonants like (p, k, t) work better than the remaining ones (about +1.9%)
   f. Sounds associated with bad word words in English (like the phoneme "uh", like in "yuck") damage the site up to -44%)
      i. In other contexts like adult material, this gives an advantage of +7%

7) Don't use dashes (so, "-"), impacting about -3%

8) Use numbers, impacting about +8.2%
   a. Use short numbers, not more than two digits – one digit are great)

There is also the “zero rule”:

- Be careful in checking if the *domain name is free to use or not*, checking if our original names were already taken.
    a. Usually, infact, queries by the user are recorded to understand name variables and this is used by sites selling domains to exploit this

- Be careful about the *choice process* of the website name
    a. It often happens that after having checked the list of names found free the next day they are taken by others. Domain distributor sites in fact advertise most used terms
    b. To check if a name was already reserved, it’s advised to use the site of this public organization called Internic found here

10.2 Technical Side: URN/URL/URI

Now, let’s go on to the technical side, so we are talking about:

- URI (Uniform Resource Identifier)
    - The real milestone, those are the “names” that identify resources on the web and were defined in RFC2396 of 1998 – many times are confused with URLs
    - They are a superset of web addresses and are the most general Web identifiers
- Examples
    - www.mysite.com is not a URI (browsers translate this for us to https://www.mysite.com, which is infact a URI)
    - https://corsi.math.unipd.it/wim
    - news:it.culture
    - telnet://example.net:453
    - mailto:massimo@gmail.com
    - *tel: +358-555-1234567*
    - *fax: +358-555-1234567*
    - *mode: +3585551234567; type=v32b?7e1;type=v110*

- URL (Uniform Resource Locator)
    - They define how to find a resource via a representation of their primary *access mechanism*
    - All addresses starting with “http/https” are infact URLs

- URN (Uniform Resource Name)
    - They specify a resource via “a name” inside a particular “namespace”
    - They have to stay unique and persistent even when the resource doesn’t exist anymore, or it is not available anymore

The URIs follow this structure:

- *schema: part-depending-on-the-schema*
- The schema defines the *semantics* (the meaning) of the URI
    - For instance, in https://corsi.math.unipd.it/wim the schema is “https”, and this is a URI whose semantics is set by the formal specification describing the “https” schema

The URIs can be either:

- *hierarchical*
  - Typical general form: *schema://authority path ? query*
    - *authority*: the element that states that URI is under control of a certain authority
      - https://corsi.math.unipd.it/wim
      - https → schema
      - corsi.math.unipd.it → authority

    - *path*: composed of zero or more *segments*, each of the form */segment*
      - *ftp://library.site.com/books/Ken_Follett/ThePillarsOfTheEarth.txt*
      - *ftp* → schema
      - *library.site.com* → authority
      - */books* → segment
      - */Ken_Follett* → segment
      - */ThePillarsOfTheEarth.txt* → segment

    - *hash (#)*: the hashtag is a reserved character in URIs
      - needed to separate the URI of an object with an identifier to a fragment of the object, so URIs can refer not only to a resource, but also to a subpart

    - *? query*: information interpreted by the resource (using "input parameters")
      - Example: http://www.mytravel.com/timetable?start=Padua&destination=Rome
      - *http* → schema
      - *www.mytravel.com* → authority
      - */timetable* → segment
      - *start=Padua&destination=Rome* → query

- *opaque*
  - Typical general form: *schema: opaque_part*
  - They differ from a hierarchical URI because they don't describe a path to a resource
  - Its schema does not begin with a slash character
- Examples:
  - *mailto*: director@cnn.com
    - *mailto* → schema
    - director@cnn.com → opaque_part

A URI can be:

- *absolute*, which means the complete address (already complete as it is)
  - Straightforward example: https://www.example.com/about-us/page.html

- *relative*, which means this is incomplete and to be completed is has to be turned into an absolute URI via information deriving from the *context*
  - Straightforward example: */contact-us/form.html*
  - Beware to use relative URIs that are completed into something we don't want
    - Example of one of the most common errors: in our https://www.mysite.com we insert a like to www.othersite.com

        - This turns completed into “https:www.mysite.com/www.othersite.com”, which is wrong and leads to nothing

    - Another example
        - Within http:www.mysite.com, we insert in a web page the email link to shop@mysite.com
        - This turns completed into “http:www.mysite.com/shop@mysite.com”, completely broken again

Continuing with URNs:

- Their typical form is *urn:NID…*
    - NID = Namespace Identifier, the “schema” of the URN in a sense
- Examples:
    - ISBN codes like $0-395-36341-1$
        - This identifies with digits: group of editors/editor identifier/title identifier/checkdigit
        - As URNs the good way to write it $is\ URN\!:\!ISBN\!:0-395-36341-1$ (so, even if website dies, there locator will always exist)
    - The New Zealand, first country to ask and obtain a URN on its own ($urn\!:\!nzl$: …)

URI/URL/URN all use ASCII as base, but giving their usage and their expansion, we passed to the IRI (Internationalized Resource Identifiers) as a standard, which extends to names and even extensions.

- This can possibly lead to attacks however, for example $p\Phi$ $(.ru)$, which can be subject to *homograph attack*
- In fact, it is possible that there are addresses that are graphically the same, but the symbols come from other alphabets
    - Example: GOOGLE.COM becomes G00G7E.C0M for instance), enormously powered up with IRIs (es. Cyrillic, Greek, etc.)
    - The attack can be very powerful, given the majority of users don’t even care about warnings and just click without problems

Back to the URIs – at their core, they represent an *opaque string*; infact, it is not permissible to infer any properties of the corresponding resource. Consider some examples:

- What is the data format of http://www.sito.it/a/b.html?
    - it does not necessarily indicate the version of a website and the language; it may be Italian/English/French
- Also another one, Queen Elizabeth Street
    - does this mean everybody living there is a noble?
    - The semantic information is not part of the URL/URI: the web address is a black-box string, and every property depends on the *schema*, not the string (because this one tells nothing about the underlying website).

To avoid opaqueness, HTTP provides methods (*content negotiation*) to transmit the correct data format. Every other algorithm (like using the final extension of the URL like .txt, .gif, etc.) is not 100% reliable.

There can also be URI problems, usually happening via the TLDs (Top Level Domain – like .com, .net, .org), proposing to solve the vulgarity/absurdity solution of the Internet general crafting a “.something” to represent just by looking at the name something about the underlying resource.

From here, some proposals were given to add new TLDs given from ICM Registry (California-based company which was selling names) – this, on the contrary, brought an overload of TLDs, too many:

- *.xxx*, to represent "pornographic" site (or .sex)
- *.kids* to represent "secure" sites for kids
- *.adult* to represent adult material

Monetarily, the *technological* cost of adding new TLDs is trivially easy and on this problem there were two types of proposals:

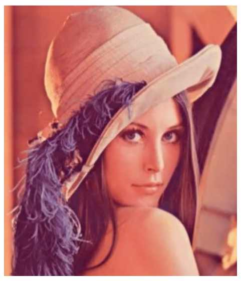
- "*light*", adding domains with these new TLDs by choice
- "*strong*", forcing all the content of a specific king into the corresponding TLD, using the law

The *social* cost, however, does not bring any advantage – consider again the same proposals:

- "light", nobody would have adopted the TLD model given companies will (maybe) buy another "*.sex*" domain, keeping the existing one.
  - This would have been subject to tracking/filtering and so the "*.com*" would have been used by convenience and easiness

- "strong", in which the WHO axis would get a lot of extra information by the TLD, but the main problem is: how to define pornographic content?
  - It all depends on the context and its perception like the photos on slide, some may be considered dangerous and other ones not
  - This does not eliminate categorization problems, considering using URLS for usages not meant to be their goal are just inconsiderate if not stupid

Consider this image, which is the most famous test image in computer graphics (great mixture of details, "flat" regions, shading, textures) and was included in every graphic program (e.g. Paint within Windows). This is based on Lena Soderberg, a Swedish woman now helping as a volunteer for disabled people.



This image became porn overnight and was banned eveyywhere; this was first scanned by programmers of University of Southern California, to first use it as a test image for digital compression and transmission tests on Arpanet.

Problem is: it was scanned from Playboy magazine from 1972. In 1996 this was discovered and a petition was started to ban forever that image (for instance, all IEEE Articles on Image Processing); formally, it was never "forbidden" but "strongly suggested" to use it less and less, so it was then taken out from all graphic programs (*social pressure* to change the perception of a resource along time).

The problem is more actual than you think:

- a lot of art can be considered porn at any time or recall the censorship over operas exposed by museums here in Italy.

So, who decides on every single conversation (emails/chats/sites/messaging, etc.)?

On this we can quote different “social ambiguity” examples:

- One comes from Douglas Crockford, co-inventor of JavaScript, JSON, etc.
    - He wrote JSMin, a reference library to minimize JavaScript to minimize code and loading of pages, which was declared open source (“shall be used for good, not evil”) – this specific sentence, included in the software license, led to many legal battles
    - The sentence is essentially an attempt to restrict the use of JSMin for certain purposes.
    - The intention behind it, as explained by Crockford, was to discourage the use of the tool for malicious or harmful activities but the open-source community discredited it a lot because of this.

- A similar example is the SQLite, in which the “May you do good and not evil” sentence was misinterpreted once again, because the “Public Domain” declaration of the software addressed some issues but left uncertainties on other ones.

The moral is the following: using URLs for scopes that go beyond, forcing information that doesn’t belong there is just stupid. In 2011, .xxx domain has been approved: this was good for ICM Registry (+200 million dollars each year), but no other useful effect in sight.

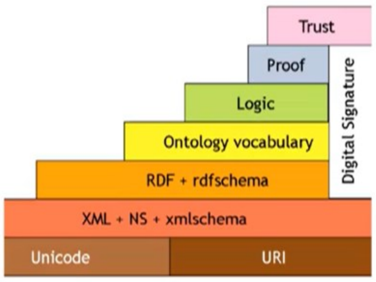
11 Information and Semantic Web (Lectures 22-23)

The primary strength of the Web lies in its decentralized nature, with its potential hinging on the *aggregation* of diverse informational resources due to the fragmented and often inaccessible nature of information. While a vast amount of data exists, it is notably sparse. Drawing a parallel with the Tower of Babel, where humanity initially spoke a unified language, the current state of the web reflects a diversity of information sources akin to the linguistic dispersion that followed the construction of the tower.

One of the first efforts to try to capture information present on the Web was via bots. One of these systems was created by MIT allowing for an easy purchase. A famous query used by this assistant was "send a rose to my girlfriend". The assistant completed the purchase successfully, but also found a completely different product, misunderstanding the context.

The *Semantic Web* tries to facilitate automatic aggregation of information and, even more, try to enable *automatic reasoning* on such information.

To help machines, we want to add "semantics" meaning in an appropriate way, so to enable information understanding and reuse. On the web, we can describe the "classic" version of the Semantic Web Tower like you see here.



11.1 Semantic Web: RDF, Ontologies

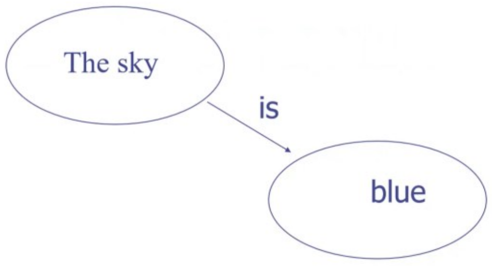
The base technology is the RDF (Resource Description Framework), used to describe resources, which was the milestone of the Web2. It can be defined as:

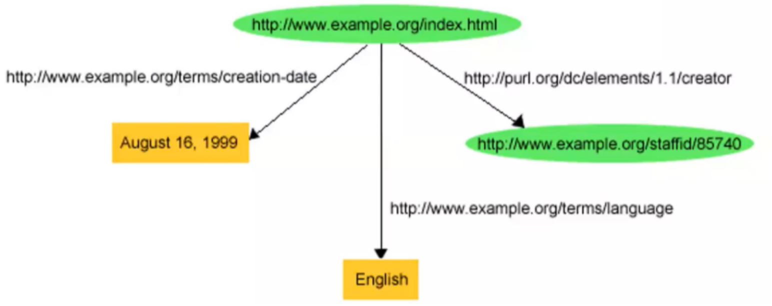
- a universal language to express information on the web and beyond (for the machines, not for us), describing *relationships* and *concepts* in a more formal way
    - metadata – so data on data, higher-level data
- a technology that allows to structure information and remove ambiguities

Other things to note:

- The idea was to build an "*enriched entity-relationship*" knowledge model that was so simple any machine could have easily understood it.
- The base grammar is made of sentences composed by "*subject-predicate-object*".
    - This is the backbone of RDF, with more power ("enriched"): referencing, quoting, bags, etc.
- Inside sentences, the data which can be inserted are URIs or Literals (strings)

Seeing RDF as a *graph* can be like the following, describing concepts and data of them:

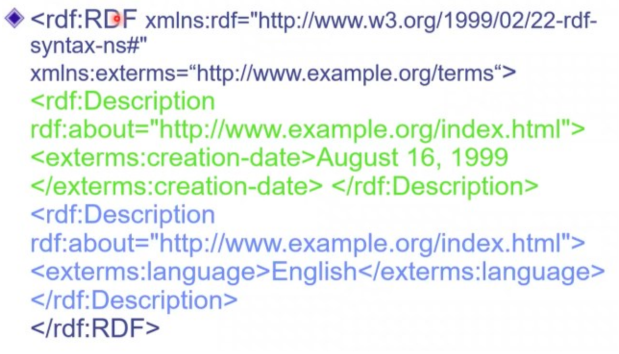



http://www.example.org/index.html
http://www.example.org/terms/creation-date
http://purl.org/dc/elements/1.1/creator
August 16, 1999
http://www.example.org/staffid/85740
http://www.example.org/terms/language
English

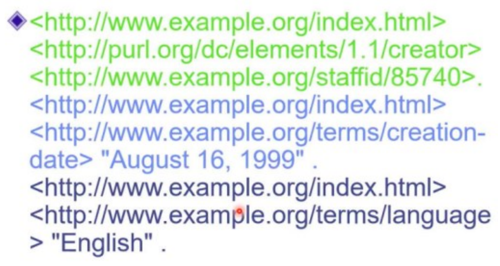
RDF, as a model, can be *written* in various ways:

- as *XML* (in a specific dialect), to elevate the complexity creating harder sentences while also enriching the meaning, allowing interoperability of informative parts
- as *N-triples*, using the triples of data "subject-predicate-object"

The following are examples of both (left/XML – right/N-triples):



```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
xmlns:exterms=“http://www.example.org/terms“>
<rdf:Description
rdf:about="http://www.example.org/index.html">
<exterms:creation-date>August 16, 1999
</exterms:creation-date> </rdf:Description>
<rdf:Description
rdf:about="http://www.example.org/index.html">
<exterms:language>English</exterms:language>
</rdf:Description>
</rdf:RDF>
```



```
<http://www.example.org/index.html>
<http://purl.org/dc/elements/1.1/creator>
<http://www.example.org/staffid/85740>.
<http://www.example.org/index.html>
<http://www.example.org/terms/creation-
date> "August 16, 1999" .
<http://www.example.org/index.html>
<http://www.example.org/terms/language
> "English" .
```

RDF as a whole, remains "separate" from the web, given it was not integrated natively in the Web. RDFa created back in XHTML2 times added "about" and "property" as attributed, integrating this way in Web.

- This model enables easy aggregation of information sources, because essentially merging graphs (a forest = many graphs) given again graphs, which don't always stay distinct but can melt together via URLs/URIs.
    - This allows to create links "automatically" between nodes
- The RDF is not enough however, considering it gives only the *basic* layer (basic model and merging for aggregation): so, to do more we need more.

The main thing we would want is to *classify* information, giving an order and a categorization.

<u>Ontologies</u> do just that; they put order between data, and they are used for information classification, creating "labels" (classes) for objects.

- Here, information is of "type X" and "type" is a *semantic type*
- Classically, types in computer science are datatypes (like integers, strings, URIs, etc.) and they give information on the *syntactic format* on the object, specifically providing the *meaning* of objects – types are more general, abstracting from the syntactical representation

Bots are not able to really understand text, but they understand the meaning of Web pages:

- Examples are P2P, MP3 songs exchange, RIIA… and the Usher Case
    - Basically, for this last example, we talk about copyright protection from RIAA using automated tracking bots to avoid piracy – one of these bots finds an instance of a song used illegally, and a legal case automatically starts against the owner of the server
    - Thing is, the song found was completely different, created for educational purposes on space by Professor Peter Usher – just a misunderstanding over a simple term
    - Now RIAA uses bots based on the "new" level, adopting semantic types

That arises because:

- *Syntactic* type (datatype)
    - "Usher" is of *string* type
- *Semantic* type
    - "Usher" is a *singer* type

Such semantic types are commonly called *classes*. So, an ontology is basically composed by a *collection of classes* and each class can contain objects, so these *belong* to a class.

- For example, let's define a wine ontology: red wines, white wines, pink wines, Chardonnay, etc.
- The class "red wines" can contain the objects "Merlot bottle from 1999", "Cabernet bottle", etc.

Interestingly, an ontology can also have an internal structure (not just a "flat" collection of classes).

- The easiest one is the so-called *hierarchical structure* (e.g. a class can be contained in another class and so on)
    - In general, the hierarchical structure is provided by a *containment* relationship among classes, which can be true or false
- Usually we see ontologies as multiple objects, considering nothing keeps us from having multiple features for the same object, according to the information we are interested in
- The point in having a structure is to allow *integrity controls* but also *deductions* (reasonings)

11.2 RDF Schema, Axioms, OWL

For instance, we can now perform information *integrity check* and also *deductions* (reasoning). To support ontologies inside the Semantic Web, we use RDF Schema (RDFS), to extend RDF for *classes*:

- This is the standard that enriches RDF with the basic support for ontologies management, defining the "informative structure", allowing objects classification with a minimum computation cost.
- It defines objects via classes, sub-classes and individuals and verbs via properties, domains and intervals (ranges), allowing taxonomy via the usage of ontologies

Let's define its main features – classes, "verbs" and also:

- *class*
- *rdfs:subClassOf*
- *individual*

Regarding its objects, we define:

- *rdf:Property*
- *rdfs:subPropertyOf*
- *rdfs:domain*
- *rdfs:range*

Example:

- "*eat*" can be defined as a property, subproperty of the "*act*" property, with domain "animals" and with range "*food*"

So, with RDF Schema we can enrich information with *categorizations*, providing semantic types and relationships.

- RDF-Schema provides basic support, called the *taxonomic* layer
- Taxonomies are the first step that can be further enriched – for instance, think again about the RDF model, reducing almost to zero the complexity of *information aggregation*

Infact, RDF allows for automatic aggregation via URLs (URIs – but those ones can be ambiguous). Tim Berners-Lee had already foreseen that at the Web dawn, giving the Axioms for Web Architecture:

- *Axiom 0: Universality 1*
    - Any resource can be given a URI
- *Axiom 0: Universality 2*
    - Any resource of significance should be given a URI

Sometimes it's not so easy to find the right name, particularly inside the Web. Given the Web decentralized nature, finding a name is not so easy:

- *URI Variant Problem*: in general, there can be many variants (URIs) for the same concept
- *URI Variant Law*: usefulness of URIs decrease exponentially with the number of *variants*

There are other instances of such a URI Variant Meaning problem, where a URI can acquire different meanings according to the context.

- Consider the famous example of the Renè Magritte painting: "Ceci n'est pas une pipe". Infact, it is a painting. As humans, we are able to rationally distinguish, probably in semantics is harder.

From this, let's specify another Axiom:

- *Axiom 1: Global scope*
    - It doesn't matter to whom or where you specify that URI, it will have the same meaning

To solve this, adding more information was thought to be a solution.

The basic support provided by RDF-Schema has then been extended with a specific layer in the "Semantic Web Tower" dedicated to ontologies, with OWL (Web Ontology Language), which is responsible for link and relationships between words, giving to each their own interest domain.

Here, we talk about (dis-)equality of objects, hence defining:

- *equivalentClass*
- *equivalentProperty*
- *sameIndividualAs* (~sameAs)
- *differentFrom*
- *allDifferent*

So, OWL allows to reduce the problem due to the URI Variant Law, establishing relationships among different ontologies.

We can map (translate) from ontology to ontology. Besides the essential concepts related to equality, OWL also adds more functionalities for features like:

- *inverseOf*
- *transitiveProperty*
- *simmetricProperty*
- *FunctionalProperty* – 1-to-1 correspondence between mother and child
- *inverseFunctionalProperty*

OWL also allows to impose more restrictions on property types:

- *allValuesFrom*
- *someValueFrom*
- *minCardinality*
- *maxCardinality*
- *cardinality*

Given we have much more functionality, we have more power and so more applications.

- It's information, so we would like to reason about it and, to do so, we can use relationships and more to define a *logic*: what we would like is to do automatic calculations, this way having an *executable logic*.
- Already, the simple first-order logic (∀ = for all, ∃ exists) is not decidable.

In computer science terms, it just means the corresponding program might not terminate.

- It is the reason we do not yet have high-level programming languages that follow a script that is closer to our language
- Think of other contexts, lie databases and SQL: in its base versions (like SQL-92), every SQL "query" (program…) terminates
    - o This happens because SQL is not Turing-complete (a system in which a program can be written that will find an answer, although with no guarantees regarding runtime or memory)

11.3 SPARQL, Dublin Core, FOAF

We need a language allowing an high-level communication (relational) and able to define low level part. There is the equivalent to SQL for semantic data, called SPARQL (SPARQL Protocol And RDF Query Language), the reference query language of SQL, allowing query by design.

- Think of the "triples" that make the knowledge graphs: *subject-verb-object*.
- The basic tool of SPARQL is provided by *graph pattern matching* using triples, for instance *?x verb object*

It does some kind of pattern matching "à la SQL" and the query structure is like the following:

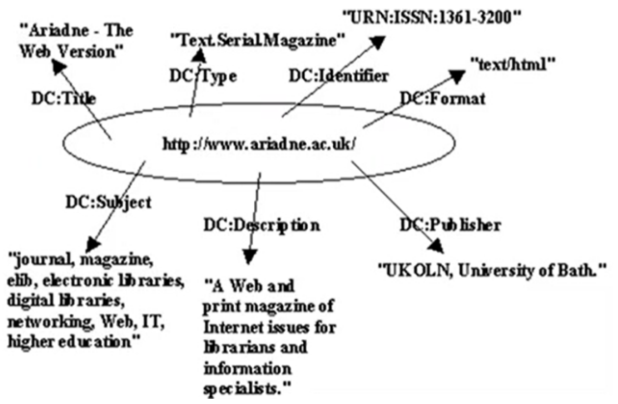
PREFIX…
SELECT…
FROM…
WHERE {
…
}
ORDER BY…

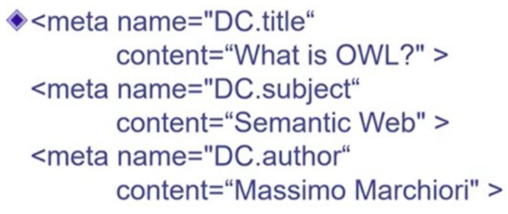
- PREFIX is just some syntactic sugar to create readable names for the URL/URI namespace
- Example: *PREFIX foaf<htpp://xmlns.com/foaf/0.1/>*, so we can use things like *foaf:name*

There are two main models for modeling data with SPARQL:

- Dublin Core (DC), which was one of the first attempts to structure the web in a semantic manner, describing URI pages and is the standard to describe the basic properties of documents (so, web pages)
    a. It allows for 15 information elements:
        i. *Title/Creator/Subject/Description/Publisher*
        ii. *Contributor/Date/Type/Format*
        iii. *Identifier/Source/Language*
        iv. *Relation/Coverage/Rights*

An example of it (left) and another example of embedding into HTML (right):


"Ariadne - The Web Version"
"Text.SerialMagazine"
"URN:ISSN:1361-3200"
"text/html"
DC:Title
DC:Type
DC:Identifier
DC:Format
http://www.ariadne.ac.uk/
DC:Subject
DC:Description
DC:Publisher
"journal, magazine, elib, electronic libraries, digital libraries, networking, Web, IT, higher education"
"A Web and print magazine of Internet issues for librarians and information specialists."
"UKOLN, University of Bath."

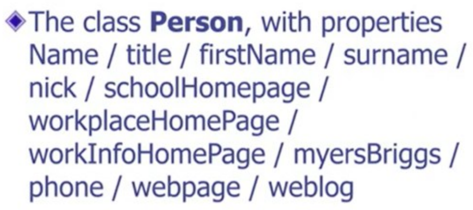

◆<meta name="DC.title“
content=“What is OWL?" >
<meta name="DC.subject“
content=“Semantic Web" >
<meta name="DC.author“
content=“Massimo Marchiori" >

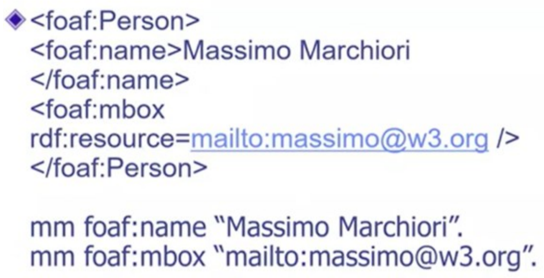
This compact set that allows to define a web page in triplets that compose a knowledge graph on which it is possible to make queries.

- For example, using meta tags it is possible via the keyword 'DC ' make these triplets available directly to the web so that intelligent bots derive useful information.

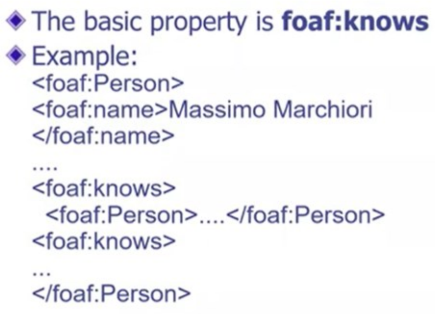
- FOAF (Friend of a Friend), the standard ontology for *Social Web*, which describes the base properties of a person.

The following are examples of how to describe a person (left), a general example (center) and friend relationship (right):


◆The class **Person**, with properties Name / title / firstName / surname / nick / schoolHomepage / workplaceHomePage / workInfoHomePage / myersBriggs / phone / webpage / weblog


◆<foaf:Person>
<foaf:name>Massimo Marchiori
</foaf:name>
<foaf:mbox
rdf:resource=mailto:massimo@w3.org />
</foaf:Person>

mm foaf:name “Massimo Marchiori”.
mm foaf:mbox “mailto:massimo@w3.org”.


◆The basic property is **foaf:knows**
◆Example:
<foaf:Person>
<foaf:name>Massimo Marchiori
</foaf:name>
....
<foaf:knows>
<foaf:Person>....</foaf:Person>
<foaf:knows>
...
</foaf:Person>

Just to note:

- *myersBriggs* classifies personalities with 16 different ones distributed over 4 different axes (Extroversion, Sensibility, Reasoning, Judgment)
- you can express the friendship concept with *foaf:knows*

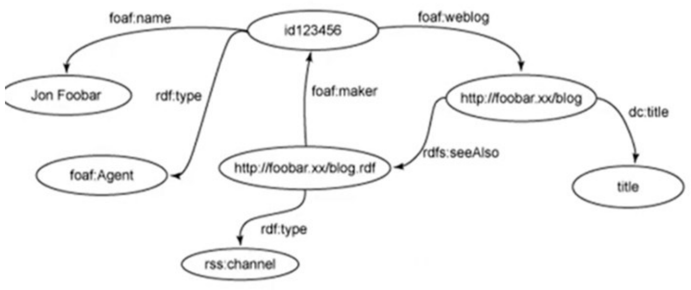
Strengths of FOAF include:

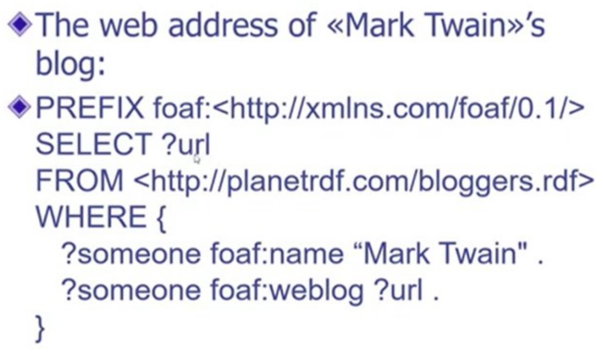
- Decentralized structure, which enables users to control and manage their own data
- Easy integration with other web technologies and vocabularies
- Enables data to be linked and shared across multiple websites and platforms
- Provides a machine-readable format for storing and retrieving social data

Weaknesses of FOAF include:

- Low adoption rate among websites and users
- Limited ability to describe more complex relationships and attributes
- Lack of standardization in the use of FOAF vocabulary and the data it represents
- Security and privacy concerns with sharing personal information in a machine-readable format

PlanetRDF has been one of the first "fully semantic" websites and, in particular, let's see a little piece of its structure (knowledge graph) about bloggers within the site (left) and a search example (right):





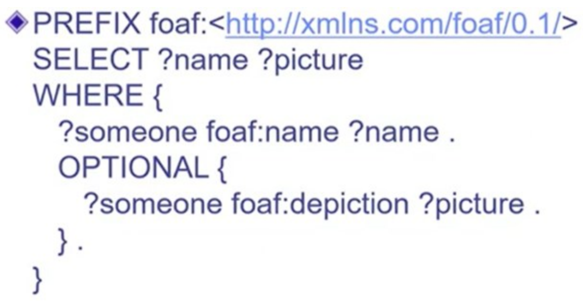
◆ The web address of «Mark Twain»'s blog:

◆ PREFIX foaf:<http://xmlns.com/foaf/0.1/>

```
SELECT ?url
FROM <http://planetrdf.com/bloggers.rdf>
WHERE {
  ?someone foaf:name "Mark Twain" .
  ?someone foaf:weblog ?url .
}
```

In the Semantic Web, knowledge graphs come from multiple sources and can be anyway containing *partial* information (for example, in a social network, not everybody has an avatar)

SPARQL allows to deal with possibly partial information by using the *OPTIONAL* command (which means data can possibly exist or not)

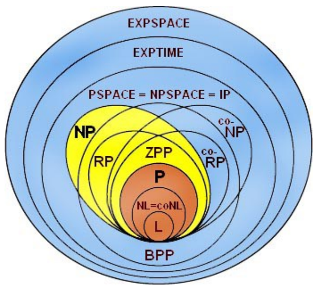


◆ PREFIX foaf:<http://xmlns.com/foaf/0.1/>

```
SELECT ?name ?picture
WHERE {
  ?someone foaf:name ?name .
  OPTIONAL {
    ?someone foaf:depiction ?picture .
  } .
}
```

In right figure example, we want to find all the people, their avatar and their picture with their name.

So, is SPARQL decidable? Well, *yes*. Studying its complexity, it seems like it has a $PSPACE$ complexity (polynomial space at risk, as shown here) – like SQL. The main disadvantage comes from particularly complex queries, which can possibly take much time to execute.
This is mainly caused by OPTIONAL, because we may fall into co-NP class.



For RDF that's' OK; about OWL, we basically maintain:
- *expressiveness*, so having an expressing logic but complex to handle
- *decidability,* having a decidable logic but limited

We try to guarantee them *both*. There are essentially three OWL layers of growing expressive power according to the desired complexity degree. In order of increasing power:
- *OWL Lite*, limited but decidable expressiveness
    a. complexity inside the so-called SHIF logic – this has $EXPTIME$ complexity
- *OWL DL,* less limited in expressiveness, always decidable
    a. complexity inside the so-called SHOIN logic – this has $NEXPTIME$ complexity
- *OWL Full*, which exploits more advanced logics, has no restrictions (so, implies high expressiveness), and it is undecidable

Consider SHIF and SHOIN refer to this table as acronym – which I added from the Italian version.

Complexity classes are statistical measures – so they are just simplifications, but learn to look beyond, because they are only a tip of the iceberg. What really matters is *the average complexity* and the *context*.
- Example 0: computational cost of search engine queries
    a. Usually, queries use exponential time algorithms, but frequently used queries are far behind. The problem is in any case solved using timers interruptions
- Example 1 (SPARQL): It is in $PSPACE$, but taking out just OPTIONAL, we fall back 100% into co-NP
- Example 2 (OWL) - Undecidable (OWL Full) / $NEXPTIME$ (OWL DL) / $EXPTIME$ (OWL Lite)

Statistically, most of the knowledge we use and that present on the web stays in sub-logics like AL (takes polinomial time – P) and ALC ($PSPACE$).

11.4 Linked Data/Linked Open Data (LOD)

The generalization of semantic web data is Linked Data (new name from Semantic Web), which is data that can be used to feed a Knowledge Graph. There is a special type of Linked Data, which is called Linked Open Data (LOD), which are available to everyone *for free*.

It is very important to facilitate their creation and management in a way that creates new knowledge and facilitate innovation, making possible the creation of ever-better services and applications.

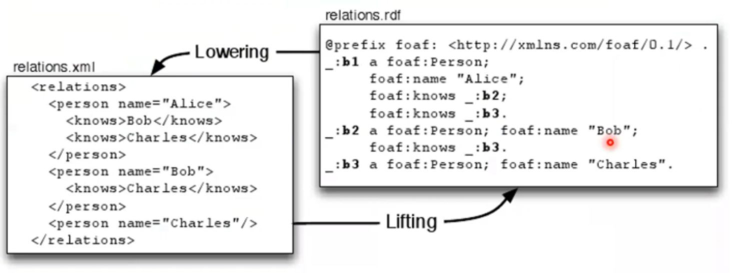
They are classified in a ranking from 1 to 5 stars (the Star Badges system) – number indicates the star.

- Data available on the web but with an *open* license (open data)
    a. Example: Images (Creative Commons)
- Data available on the web but in a *machine-readable* structured format
    a. Example: Excel
- Data available on the web but using a *non-proprietary* data format (not necessarily RDF)
    a. Example: CSV
- Data available on the web but in *semantic web* format
    a. URI identifiers are employed so that it is possible to point to a single piece of data
    b. Examples: RDF*, OWL [star here means "including all RDF versions"]
- Data available on the web in semantic web format *with data linked to other data sources* to provide context
    a. Example: Graph

There are interesting "middle zones" in LODs, in particular the three-star layer (non-proprietary data), not necessarily in RDF*/OWL format.

- Going from the non-Semantic web world (e.g. XML/HTML/SQL, etc) to the RDF world is called <u>lifting</u> (so going from 3 stars [or even lower] to reach 4 or 5 stars)
    b. This basically consists in giving data a semantic format
- The other direction (lowering the stars) is called <u>lowering</u>

The following is an example of these last two:



```
<relations>
  <person name="Alice">
    <knows>Bob</knows>
    <knows>Charles</knows>
  </person>
  <person name="Bob">
    <knows>Charles</knows>
  </person>
  <person name="Charles"/>
</relations>
```

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:b1 a foaf:Person;
     foaf:name "Alice";
     foaf:knows _:b2;
     foaf:knows _:b3.
_:b2 a foaf:Person; foaf:name "Bob";
     foaf:knows _:b3.
_:b3 a foaf:Person; foaf:name "Charles".
```

Lowering and lifting are important to do the so-called <u>mashup</u>: join the XML/XHTML/HTML world (and all of its amount of information) with the one present in the RDF/Semantic Web world and vice versa.

Useful examples of lifting tools are:

- *D2RQ* (d2rq.org), which allows to automatically pass data from SQL to RDF (high-level – JDBC database connections) and can also work as a server, giving immediate access to data as RDF
- *Triplify*, small plugin with takes tables and translates them into sentences, creating a semantical structure from relational databases – like D2RQ but without server side
- *Openlink Virtuoso*, which is a "universal server", implementing many protocols to handle information and allows information at various level among SQL, XML, RDF, web services, available both in open-source and commercial editions

About other data (for example web pages, text, pdf, word, etc.), there can be NLP (Natural Language Processing) tools, which analyze textual/hypertextual information and allows automatic lifting.

Example of lifting tools from the lowest format are:

- *Open Calais* (now called *Refinitiv*), technology who tries to understand text
    c. You take a text/a page or something and automatically recognizes it and tags it with useful information
    d. Here is taken for example italia.it, critiqued for being basically useless and wasted in terms of information
    e. Take the page and put it here, now all information appears with keywords, links, text, etc.
- Many other examples are: Spotlight, Alchemy, Extractiv, Ontos, Evri, Saplo, Lupedia, etc.

An example of how much potential of these tools give is <u>www.wikido.com</u>

- This takes a set of selected information sites and categorizes sites, filtering based on semantic information
- There is a pool of websites, lifting the information and produce a navigation system able to gather all events in the US, transforming everything in high-level information obtaining structured data – this is has been successful for years.

In offering the data, to have at least 1 LOD star, data must be on the Web:

- to do this, we could just make the data available as a file in a certain URL address: in case of 3 LOD stars, this is called an *RDF dump*
- problem: "dumping" data is not very practical – so, we want to offer a more structured access, for instance via an interface

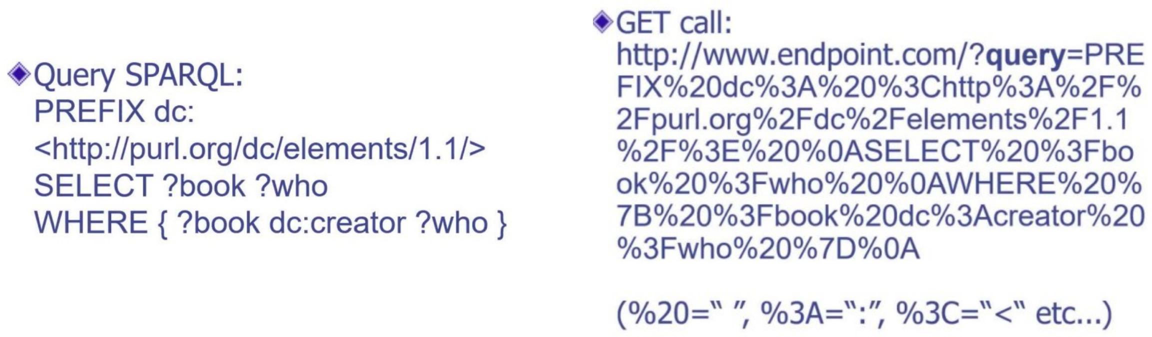
Instead of providing a raw file, we could offer a SPARQL *web service*:

- This is much more powerful and flexible, and we talk about SPARQL *endpoints*.
- The endpoints access is specified in various ways, the easiest is via GET and we can do an HTTP GET to the SPARQL endpoint, passing various parameters in the query part of the URL.

There are some GET parameters to note:

- *query*: the SPARQL query to execute (modified via the classic %-encoding ["urlencoding"] for URLs)
- *default-graph-uri* (optional): the URI of the default knowledge graph
- *named-graph-URI* (optional): other knowledge graphs that can be used in the query

Here are some query examples:



◆ Query SPARQL:

```
PREFIX dc:
<http://purl.org/dc/elements/1.1/>
SELECT ?book ?who
WHERE { ?book dc:creator ?who }
```

◆ GET call:

```
http://www.endpoint.com/?query=PRE
FIX%20dc%3A%20%3Chttp%3A%2F%
2Fpurl.org%2Fdc%2Felements%2F1.1
%2F%3E%20%0ASELECT%20%3Fbo
ok%20%3Fwho%20%0AWHERE%20%
7B%20%3Fbook%20dc%3Acreator%20
%3Fwho%20%7D%0A

(%20=" ", %3A=":", %3C="<" etc...)
```

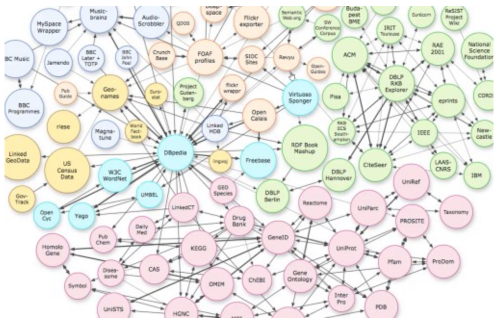
Let's see now examples of LOD instances that are available:

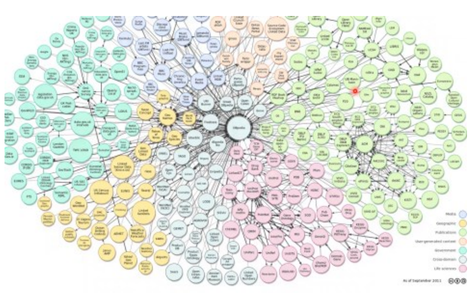
- *DBPedia*, essentially, the *semantic version of Wikipedia*
  a. Same pages of Wikipedia, but lot of properties semantically describing the pages
  b. All the structured data present can be used at any time to build knowledge graphs
  c. It has its own ontology and also reuses the reference web ontologies, so to be interoperable (more than 5 million resources)
  d. We can do the same with endpoints and offer this service to other websites easily
  e. Example link: https://dbpedia.com/resource/Italy

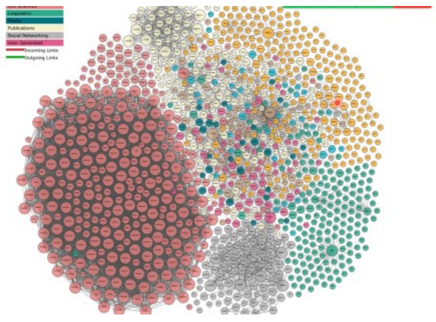
DBPedia favors interoperability by reusing reference ontologies, like the ones provided by:

- *Schema.org*, which defines the most important/used ontologies for Web data, for instance about *people, products, organizations, places, restaurants*, etc.
  a. Example link: https://schema.org/Restaurant
  b. Here we can see all important data – has menu/serving cuisine/star rating/currencies accepted/opening hours/payment methods
  c. Putting this information explicitly, we can power up a website, giving a lot of information about the context of a specific simple page

Here are three images about the Semantic Web data structure:







Examples of LOD data usage:

- *Relfinder*, which creates knowledge graphs finding common threads of information
    a. This way, it build a knowledge graph linking information forming connections
    b. In this case, it tries a shortest path towards information
- *QueryVOWL*, visual query language to create entirely queries via visual elements
- *national data*, considering every state usually makes open data available
    a. E.g. Italy with dati.gov.it
    b. In case of European countries, laws pose data as mandatory for every public data = public access without restrictions

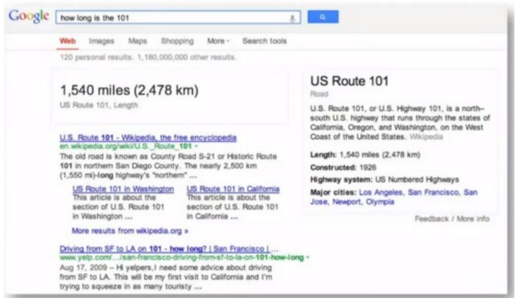
Google has full support for the semantic web, and it understands all schema.org and beyond.

- All the semantics is used internally, while externally only some properties are visible to the user
- For instance, this happens via *Rich Snippets*, which small summaries of data supporting many kinds of items (giving more points to such websites
- For example: Authors, Business and Organizations, Events, Music, People, Products, Recipes, Video
    a. These can be very useful to give content immediately for the user without browsing the entire site to find such)

Just use semantic data on your pages to enrich them: for instance for products, recipes and so on. Search engines themselves, like Google, give advice for these things as "advanced SEO", making users stay more in their engine and selling more advertisements.

All of this is integrated with the much bigger knowledge graph (used also for ranking): so, users only see a very limited amount of information.

An example of this integration to refine queries here on right figure, for example via lifting.



120 personal results. 1,180,000,000 other results.

1,540 miles (2,478 km)
US Route 101, Length

U.S. Route 101 - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/U.S._Route_101
The old road is known as County Road S-21 or Historic Route 101 in northern San Diego County. The nearly 2,500 km (1,550 mi)-long highway's "northern" ...

US Route 101 in Washington
This article is about the section of U.S. Route 101 in Washington ...

US Route 101 in California
This article is about the section of U.S. Route 101 in California ...

More results from wikipedia.org »

Driving from SF to LA on 101 - how long? | San Francisco | ...
www.yelp.com/.../san-francisco-driving-from-sf-to-la-on-101-how-long
Aug 17, 2009 – Hi yelpers,I need some advice about driving from SF to LA. This will be my first visit to California and I'm trying to squeeze in as many touristy ...

US Route 101
Road
U.S. Route 101, or U.S. Highway 101, is a north–south U.S. highway that runs through the states of California, Oregon, and Washington, on the West Coast of the United States. Wikipedia
Length: 1,540 miles (2,478 km)
Constructed: 1926
Highway system: US Numbered Highways
Major cities: Los Angeles, San Francisco, San Jose, Newport, Olympia
Feedback / More info

LOD offers data in various ways:

- *Freebase*: super container of base semantic sponsored by Google, in which data is offered in format *N-Triple RDF*, also with endpoints
    a. It contains over 3 billion of tuples
- *Wikidata*, another project sponsored by Google and how semantic media should be – contributing to semantic data in a way like normal Wikipedia, asking the users to contribute
- *Google Knowledge Graph Search API* is offered by Google to access the knowledge graph

The following concept was not treated in the chapter: Levenshtein distance

Written by Gabriel R.

12 Mobile Web & Apps (Lectures 24-25)

We talked enough about desktop apps: now let's talk about mobile web and apps. Here, apps are not only web, but related between each other. After talking about classic usability and its principles, we can now just focus in the main *differences*.

Google tried to penetrate desktop market, but actually succeeded inside the mobile market, acquiring Android. This allowed to merge mobile and web world in a way which could have been actually useful to the market itself. Infact, creating a mobile website gives more points as an overall score to a website, not penalizing it.

Google tried anyway to put some FUD (Fear Uncertainty and Doubt) to the market, marketing the penalties for those sites which have not a perfectly suitable mobile version – *webmaster spam*, if you don't follow the rules, the ranking goes down.

12.1 Mobile Components

We start from the *causes* of the differences between the classic and mobile world: a different *execution model* due to the *3 Base Components of Mobile*:

1) Being mobile
2) The screen size, which is obviously smaller
3) The interaction, which means *finger* and not the mouse

Beware of the target people – consider the Facebook case, which has three mobile versions:

- *m.facebook.com*
    a. This is what we think we see when we use a smartphone
    b. Instead, this is the non-touch version, whereas *touch.facebook.com* is for touch-based smartphones
- *touch.facebook.com*
    a. This is the version it's used instead of the previous one
- *0.facebook.com*
    a. Super-fast version, limited-bandwidth, bare-bones functionalities
    b. Images aren't even displayed and need an extra click to be seen
    c. A version trying to always use text only, with higher speed and happier users
    d. This is a good thing when for the user the cost/benefit ratio is very low
    e. This is offered free in all those places where connections are slow and quite costly
    f. This way Facebook managed to reach a *huge user base*, even in countries with limited access to mobile

12.2 Mobile Implications

There are implications in being mobile:

1) the <u>network</u>

The connection type changes: infact 3G networks are on 40% slower than desktop connections, while for 4G/LTE connections, it wildly depends (-5% to -40%).

Every site pays a *time price* when it's seen on a smartphone.

- Having already talked about the importance of time for users (e.g. timers, etc.), if a page takes 40% more, it takes more time out of any timer. Users compare that with the average loading times
    a. In the desktop case, users wait *2 seconds* as a maximum time per page
    b. Beyond this time, they have a *delay* perception, corresponding discomfort to the site
    c. Local loading time of every page is an important aspect to consider: every page should be fast enough

- In the mobile case, it's almost identical: users always expect at most 2 seconds
    a. We have to be careful, adapting not only the visual layout, but also about possible penalties of mobile network

- Always consider connection speed in the mobile cases: the best way would be (when possible) to have *lighter* pages for mobile than desktop

Beware the same temporal limits also apply to *apps*: an app taking more than 2 seconds for an *action* is considered slow. So, for the success of an app, a fundamental factor is <u>responsiveness</u>: users should not perceive a delay unless the context clearly justifies that (e.g. photo upload). This is always valid for any action that needs a network connection (even if it's not our fault).

When an application takes up too much time:

- use a *progress bar* or a so-called *spinner*; in this case, even if there is some waiting time, users are warned
    a. it looks great, but users don't like it, because it's like telling the user to wait forcefully and bars/spinners *explicitly* signal *a problem*
    b. it's like being in a queue at a shop to pay and to be told to be patient
    c. users *perceive* the time delay as *longer* when there are progress bars or spinners
        i. this is annoying as a technique both in desktop and mobile case

- use *transitioning*, keeping the user busy with animation and other stuff, even entertaining him – here is the perfect place to put the Dancing Jesus
    a. a particular case of transition is the *skeleton screen*: if you know the final layout of an action, start drawing it even if not all data is received – like Netscape, showing something to the user while content is loading
    b. an example was the first Instagram, then followed by Facebook, with the multiple taps to give likes (the server actually goes on a queue, it's only a UI effect)

- use *preemptiveness*, trying to foresee some actions
    a. for instance, an app has an “upload photo” action
    b. The classic way would be to let the user choose the photo, then ask for a description, then upload
    c. The better way is to load the photo as soon as it’s been chosen: almost instantaneous upload, no wait, users are happy

2) the <u>screen size</u>

Main issue: a page will hardly be seen without scroll. How bad scroll is for mobile? It depends, actually.

- *Horizontal* scroll is as bad as for the desktop case – horrible forever and ever
- *Vertical* scroll instead is often not so bad like the desktop case
    a. The main reason is the computational cost is quite smaller (physically/mentally)
    b. It may also be bad for users, when it’s used to offer choices (for instance menus, list of products)
    c. The user here has to keep in mind the context above to complete the choice action and this overloads the user with mental fatigue, proportional to the size of hidden information

Scrolling is done via *gestures*, without the burden of using the mouse in specific zones to manage it (even worst, sometimes using dragging too)

- It’s better to minimize vertical scrolls for choice lists
    a. for instance in some cases *even avoid images*
- Images in lists are justified only when it’s about final products (for the same reasons seen already for e-commerce sites)
    a. this can lead to the memory efforts mentioned before
    b. so the best solutions should *minimize scrolling*

The middle way would be to have *text with small images*. That’s good for general categories/menus, but not for final products: the primary pulsion is always to also *see the product*.

- To remedy for the small size, one solution can be to use *icons* rather than *text* for buttons
- The problem is the same as for the desktop though – this works if users know what the icon means (which is ok for instance to the search lens
- Ok for the search lens – in all other cases, always prefer *text* to icons

Consider the *hamburger icon*, which is the usual icon to open a specific menu. Menus are an essential components of sites, but that’s potentially so wasteful for mobile: we have to use lot of text in a small screen, but if we are strong enough, we can “push” in some other way.

- For example, Firefox changed some time ago its desktop interface, introducing the *hamburger* on its *right* side and users got *super-angry* as a result
- This is from the Mozilla Corporation, which is a “for profit” private entity – this is financed by Google, and this is because Google is the default search engine
- Of course, both Chrome and Firefox tried to get users used to this menu icon, even having something hated from users, then getting them used to
- The mobile layout was then introduced into the desktop and commercially pushed wherever possible

So, remember even here: users *always prefer text*.

Back to icons, if we want to use them, the best is to also use text (like for desktop), but keep in mind that the majority of users does not know the meaning of icons.

- Each one should respect the *explainability* principle: keeping pressed the icon the user can obtain textual information on its action.
- A companion principle is the *escapability*: a user can always "escape" from an action just by moving away the finger from the icon and this holds for any touch action

Having a smaller screen also implies a redefinition of what "invasive" means for an advertisement banner. Let's talk about the most common sizes of ads:

- *interstitial ads*, which are usually fullscreen and HTML5-enabled
    a. generally bad, but something is even worst
- *smart banners*: sizes = (screen width) x (32/50/90)
    a. there are kinds of *smart banners* which are implemented in a *fixed* position (not scrollable)
    b. an *always-visible ad* is a good idea but that annoys users, particularly if it changes distracting the user – this is totally annoying for users
    c. consider the Smart App Banner, which for example publicize the site app: this is super annoying for users (equivalent for popup for desktop)

3) <u>the fingers</u>

With fingers' interaction, *drag* is not a problematic action, because there is no muscle effort involved like the mouse usage. This aspect make gestures particularly appealing to user and related uses (like swiping).

- Consider action should not be timed (*untiming*), without timers associated on duration, just distinguishing between "tap" and "drag" (prolonged press)
- Some mobile user interfaces/apps instead assign different actions to different press durations, and this causes *errors* and *user stress*

There is a big problem with fingers: compared to mouse, fingers are rather rough and in the mobile world, this is referred to as "fat fingers" – this can be problematic for advertisement clicking, making us click more. Generally:

- the average finger is 11 millimeters wide
- children finger is 8 millimeters wide
- big finger are up to 19 millimeters wide

So, any *clickable area should be big enough to be centered well with a finger*. Consequently:

- The *minimum size* of any clickable is *7x7 millimeters*
- Around, we need a "padding" safety zone wide at least *2 millimeters*
- If we really don't have room, we could go down to *5x5*, factoring a 20% decrease in precision (hence, more user frustration)
- If on the other hand we want to make users happier, we can use *9x9* millimeters and beyond

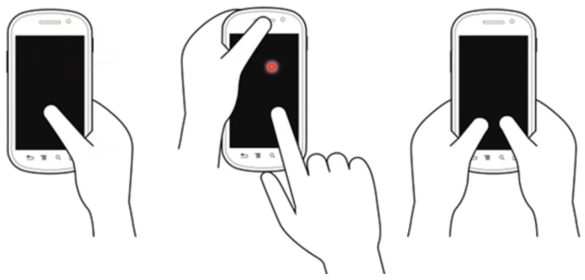
It's amazing the number of mobile websites or apps that don't respect these basic guidelines and it has severe usability problems and to diminish this we can leverage *reversibility*:

- every action taken should be reversible, so always let users be free to be wrong and escape from the action
    a. even more in situations of potential errors due to the fat fingers

Useful principle also in connection to explainability (so users can always safely try to see what a button is doing, having no permanent damage).

- Example of epic fail: the mobile keyboard
    a. This is a completely inefficient keyboard of the desktop, which was meant as QWERTY to slow down user typing in ancient times, as we all computer scientists should know at this point
    b. On mobile, this is completely inefficient, considering making users learn a new keyboard was considered bad, but usability here is definitely not that good anyways

In case of Fitts law, this "almost" changes. In Fitts for mobile, the size of the object matters, but also the imprecision due to the finger and also the *mobile grasp* (given it's something completely different from desktop and the mouse is the finger itself).

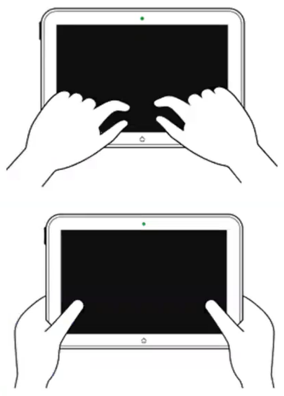


There are actually five classic cases:

- one-handed (1)
    a. a hand and thumb as pointer
- two-handed with one active hand (2)
    a. one hand holds the device, the other keeps the pointer
    b. same for left-handed
- two-handed symmetrical case (2)
    a. two hands with thumbs as pointers
    b. same for left-handed

The amount of effort needed to reach buttons or move the hand to do more movement: this is not good definitely. Each of these five cases behaves in a different way.

- For instance, using thumbs make the fat finger problem worst (clickable areas should be +2mm wider) – people like using thumbs, but *this is terribly unprecise*
- Moreover, in the cases using thumbs (one-handed), the low-cost area is limited to a certain circle: closer or farther areas are more costly, in some cases very costly (because they are not reachable)
- This is an even bigger problem for *tablets*, where typically navigation is two-handed (minimizing muscle effort) with two thumbs. The bigger screen size of the tablet makes this problem *worst*.



The best interface considers all these constraints and keeps all the *controls in the lower part of the screen* (infact, that's where the basic smartphone commands already are, because they are easier to reach – do not put things on the top).

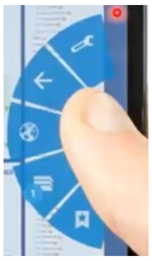
Both on smartphones and tablets, the screen shape (and reachable areas) change also when passing from normal to *landscape* position. In these cases, liquid layouts can be problematic.

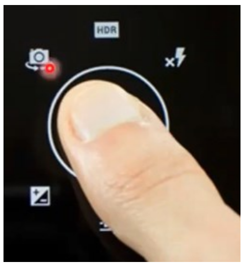
Controls here should be also on the angles, not on the central part if possible. In this case, we cover both the tablet and smartphone case.

The best mobile interface considers all these cases, optimizing according also to *finger reachability* and allowing customization for the right and left-handed grasp. If we remember Fitts – use the *borders*.

The advantage compared to desktop case is that the "window" is already maximized and so we can always user border, for instance via a *fan menu*.

Fan menus are ok for mobile (left figure), but classic pie menus (right figure) have problems (in this case, our finger is the pointer, and we cover commands and the actions below, not like desktop)





Similar historical error was also in the first versions of the keyboard: with finger we cover up the letters pressed or also swipe-based keyboard, because of not being sure if the good letter was pressed or reached. An interesting reading (Italian one) on zones reachable or not here.

12.3 Mobile Apps

We all know the success of apps: thinking better about them, they are just a consequence of computational effort minimization. It simply *minimizes access time* to the functionality of interest (even though there are website, we use apps because they are easier), usually accessible from *app stores*, minimizing the effort to find the right app, doing everything like the desktop but compressed.

For instance:

- Many users, more than a fourth, use apps more than 60 times a day
- They win over mobile web, because of their greater usability, even if more constrained
- Smartphones users pass on average 86% of their times and 14% on the web

Users are doing many things with apps:

1) Games (32%)
2) Social sites (28%, trend growing)

Attention: apps are so trendy, we might be tempted to enter this world and use them as fast recipe for success. That’s good, but be aware that just because of their success, it doesn't mean this is not an easy environment without competition.

Consider the *death sequence of apps*: 26,13,9: 26% of apps are used once, 13% twice, 9% only three times. Apps are like butterflies: so beautiful, so attractive, so short-lived. They last *from 4 months up to 1 year*.

- For instance, *games* (the most attractive) paradoxically have an average lifespan of 4 months
- Important signal of duration: if users growth last more than the critical period of *3 months*, then the app will likely last much longer, otherwise it will die quickly

So, even more critically then for websites, it’s crucial to be found by users. Primarily, apps are found via the app store and so with a search engine. Just not Google or Bing, it’s the proprietary engine of the app store and there is a problem using this specific engine.

The corresponding SEO for apps is ASO (App Search Optimization), similar to the last one with differences. Being a search engine, this also works via *keywords* that we should carefully insert. Places are like the app description, using possibly dedicated keywords and also in the most relevant place: *the name* of the app itself, where it’s useful to insert one or two crucial keywords (almost like sentences).

There are other sources for ASO:

- Google and Apple also use a lot of information coming from the *global social system*
- For instance: downloads (integrated over time), usage time, ratings and reviews, uninstalls, brand and positive/negative parts coming also from other systems (*web and email*)
    a. Doing a good number of downloads in the first weeks and the engines will give a bonus and more visibility immediately, continuously
    b. It's important also to open it many times and using it at least for a few seconds
    c. Rate and write reviews to give textual score and give more information

In some old notes, a Social Web part is present: it was not treated after this.