Axel Herold, Peter Meyer, and Frank Wiegand

1 The Technological Context for Internet Lexicography

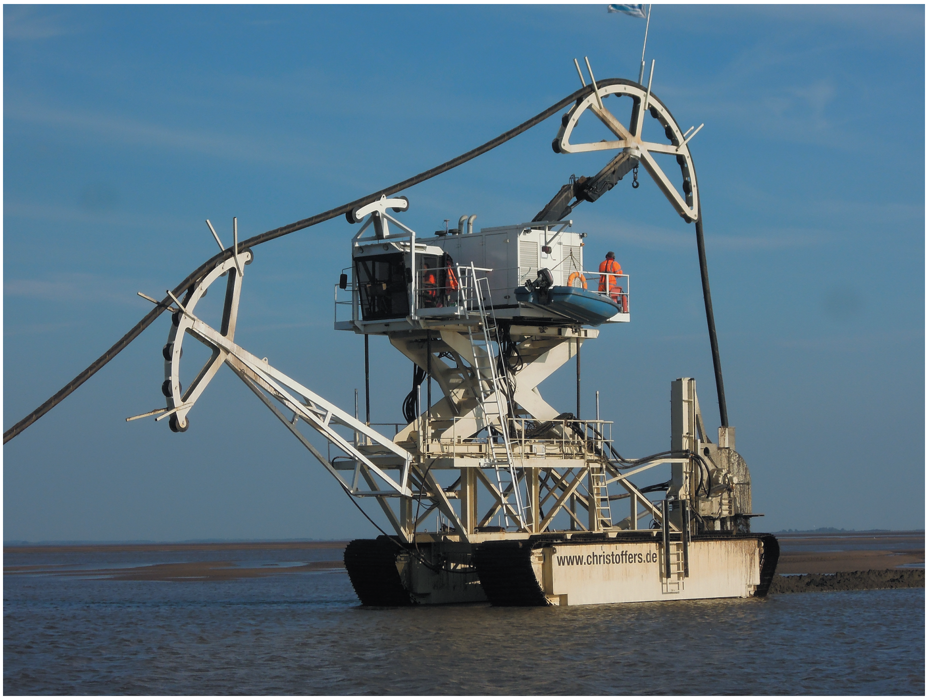


Fig. 1.1: This modern submarine cable trencher, a special machine for laying undersea cables offshore, weighs more than 100 tonnes.

There have only been Internet dictionaries for a few decades – compared to the thousands of years of dictionary writing history, this is a vanishingly small period of time. The photograph illustrates one of the many technological and infrastructural require-

Note: The German version of 2016 on which this chapter is based was written by Axel Herold, Lothar Lemnitzer, and Peter Meyer.

Axel Herold, Berlin-Brandenburgische Akademie der Wissenschaften, Jägerstraße 22–23, 10117 Berlin, Germany, e-mail: herold@bbaw.de
Peter Meyer, Leibniz-Institut für Deutsche Sprache, R5, 6–13, 68161 Mannheim, Germany, e-mail: meyer@ids-mannheim.de
Frank Wiegand, Berlin-Brandenburgische Akademie der Wissenschaften, Jägerstraße 22–23, 10117 Berlin, Germany, e-mail: wiegand@bbaw.de

https://doi.org/10.1515/9783111233758-002

ments of modern Internet lexicography: by far the largest proportion of international data transfers is handled by a network of glass fibre cables measuring many hundreds of thousands of kilometres, which often cross the oceans at great depths.

Computer technology is becoming ever smaller and cheaper, both to acquire and operate, and its processing and storage performance is increasing exponentially. This is one of the technological requirements for making dictionaries available online, but so too is the infrastructure of the Internet, which makes it possible to exchange information and data simply and reliably between billions of interconnected computers. This chapter is devoted to the fundamental technological preconditions for present-day Internet lexicography. First, we outline what actually happens "behind" the user interfaces that are visible on the screen when a user accesses a dictionary online and how these processes can be recorded in log data for the purposes of documenting them. Second, we discuss how the identity and long-term availability of content can be maintained in view of the possibility of online material being constantly updated.

1.1 Introduction

The digital revolution in the 20th century has completely transformed the ways in which dictionaries are compiled and used. Just like the resources connected to them, such as textual corpora and multimedia, dictionary texts can be represented in digital form, that is, ultimately as sequences of 0s and 1s. Digital data of this kind can be processed at ever greater speeds by computers, stored in ever greater quantities so as to be downloaded rapidly anywhere, quickly transferred to a worldwide network of computers, and presented flexibly in audiovisual form to be viewed and manipulated by humans. For both lexicographers and dictionary users, this opens up a broad spectrum of possibilities; these are the subject of the present volume, including in particular:

– the managing, searching, and exploring of dictionary data, including the large textual corpora connected to them (→ Chapter 3),
– the (semi-)automatic creation of particular dictionary content (→ Chapter 7),
– the collaborative, ubiquitous compilation of dictionaries (→ Chapter 8),
– the removal of the constraints of the print medium (→ Chapter 5).

It is an essential prerequisite when engaging with the topic of Internet lexicography to have a basic understanding of the technologies required for the technical development of Internet dictionaries, their functioning, and use. This applies in particular to the associated requirements for structuring and representing the dictionary content, as is shown in detail in → Chapter 4 on data modelling. However, even in the realm of web development, an enormous variety of technologies is employed so that this introductory chapter can only provide an overview of knowledge in selected areas of particular relevance to lexicographic work. Furthermore, in order to make the discussion more accessible to newcomers, the presentation in → Section 1.2 very deliberately oversimplifies the reality, focusing only on the aspects essential for lexicography. The result is that technical details are sometimes knowingly described in a manner that is incomplete or formally not entirely correct.

1.2 Internet technology in the context of Internet dictionaries and lexical information systems

1.2.1 Network communication on the Internet

The notional starting point for our short tour of the most important web technologies is the typical case in which a user of an Internet dictionary would like to view a word entry in the browser on their computer. Let us take a toy example. The user would like to be able to see the entry for the English noun *disproof* in the monolingual English dictionary "MyEnglishDict". To do that, they must tell their browser where "on the Internet" the website with the required information can be found. For that, the browser needs an *Internet address*, more formally a URL (uniform resource locator) that indicates where exactly this site can be found. In our example, this URL might look as follows:

```
https://www.my-english-dict.com/entry/disproof
```

A URL like this can be entered directly into the address bar of the browser. The browser then retrieves the resource (website) identified by the URL from the Internet and displays it on the screen. Normally, though, users do not enter such complex URLs manually themselves but rather click on a *hyperlink* (usually abbreviated to *link*) that is located on another web page, say, a list of results generated by a search engine like Google or Bing. Such a link leads the browser to the appropriate web page: when the user clicks on the link, this prompts the browser to load from the Internet the website with the URL that is connected to the visible text of the link. In the most basic case, the technical process that follows after a URL link has been clicked on is identical to that prompted by entering the same URL manually in the address bar. In a similar way, the main web page of the dictionary "MyEnglishDict" may offer a list of headwords that are hyperlinks to the web pages belonging to the dictionary entries concerned. The user may also use the search functionality of the Internet dictionary, for example, to search for lemmas beginning with "dispr"; the results are then presented as a further list of links on a search results page.

What does the process look like by which the browser retrieves the information from the desired website and displays it?

First, a few general points. A *web browser* is a program that runs on a device connected to the Internet (PC, smartphone, etc.) and that is in a position to download information from the Internet and display it on a screen. The *Internet* is a complex worldwide network of electronic hubs (so-called routers) mostly connected to one another by cables; essentially, every device connected to a hub in this network can send information to every other connected device via these hubs in a way that is extremely failure resistant. This functions by virtue of every device on the Internet being allocated a unique identifying combination of numbers, its *IP address*. The dictionary data (web pages, etc.) to be retrieved are stored on a particular computer managed, for example, by the provider of the dictionary or an external third party. Thus, the web browser has to have the data from the desired web page sent from that computer over the Internet. To do this, the browser must send a request over the Internet to the relevant computer and, hence, has to know the latter's IP address.

However, the URL given above does not contain an IP address, which may even change from time to time for any given device, but rather an alternative name for the computer that is easy for people to read and recognise, its so-called *host name*, i.e. www.my-english-dict.com. Through communication with specific computers (so-called *name servers*) on the Internet, the browser can find the current IP address of the computer (say, 93.184.216.34) for this host name. In fact, it is sometimes even possible to use the IP address directly in a URL instead of the host name. For example:

```
https://93.184.216.34/entry/disproof
```

The browser then sends its request for a web page as a message to the computer with the IP address 93.184.216.34. This message consists simply of a sequence of characters (numbers and letters as well as some specific control characters), which are ultimately coded as sequences of 0s and 1s. A strict system of rules, a so-called *network protocol*, determines how the message has to be constructed; that is, it provides formal rules for the language through which the computers communicate with one another. Which protocol is used is also given in the URL: the prefix "https://" indicates that the usual protocol for transferring web pages, *HTTPS* (Hypertext Transfer Protocol Secure), is being used. In the past, and in a few cases today, web pages used the variant HTTP, which provides no data encryption; it corresponds to the URL prefix "http://". The protocol prefix can usually be omitted when the URL is entered manually into the browser's address bar. The message sent by the browser over the Internet after the URL has been entered is itself a short text that specifically contains a line with the actual request, in addition to some further lines with meta information, the HTTP(S) headers (→ Section 1.3). In our case, the line containing the actual request looks as follows:

```
GET /entry/disproof HTTP/1.1
```

The keyword GET in the HTTP(S) protocol designates the method; in this case it simply requests the transfer of data from the remote computer, as opposed to, say, the modification of data on the remote system. GET is followed by the *URL path*, which is, in a sense, the actual designation of the required digital resource, here the requested web page. Next, the version of the HTTP(S) network protocol to be used is given, here 1.1. Note that this is always indicated as HTTP/1.1, even with HTTPS, since the underlying message exchange is the same in HTTP as in HTTPS, the only difference being data encryption in the HTTPS variant.

The URL path can also be derived from the URL: in the present elementary case, it is obviously just the part of the URL that follows the host name. It consists of individual segments (series of characters) that are separated from one another with slashes. There are no generally binding rules as to what the URL for a specific resource must look like. In this example, it could have read "/dictionary/entry/3325" or "/dict/disproof/showentry" instead of "/entry/disproof"; ultimately, the programmer of the Internet dictionary makes the relevant decision. In many cases, paths are chosen so that they give a rough impression of the structure of the online content being made available.

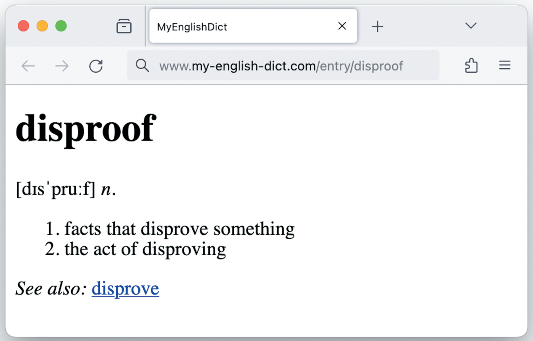
The only kinds of URLs that users normally enter manually into a browser are those with an *empty path*, that is, those whose URL consists only of a prefix like https:// and the host name: "www.google.de". The empty path is indicated in the HTTP(S) request with a simple forward slash: "/":

```
GET / HTTP/1.1
```

In typical cases, the empty path corresponds to the *home page* of an Internet presence from which the desired pages are reached through links or search functions.

A technical note for those who are interested and have prior knowledge: it may well be the case that the URL path corresponds to an actual data path on the computer responding to the request so that a path such as "/dictionary/entry/3325" refers to a piece of data with the name "3325" in the subdirectory "entry" in the directory "dictionary" on a hard disk drive, the content of which is sent back in response to the browser making the request. This is the reason for the hierarchical form of URL paths. Generally, though, there is no correspondence between the URL and the location of the data on the remote computer because the answer to a request is usually only "constructed" after the request and is not already waiting, prepared in advance, on a hard drive.

In order for the computer with the address 93.184.216.34 to be able to process the request at all, there must be a program running on it that is in a position to receive and respond to requests from other computers over the Internet. In very general terms, this kind of program is known as a *web server*. The web server then passes the request to another program, the *web application*, that is responsible for delivering the web pages of the MyEnglishDict dictionary. So it is ultimately the web application that services the *request* “GET /entry/disproof”, providing a specific description of the requested resource, in this case the code for the web page with the dictionary entry on *disproof*. This web page could, in the simplest case, look like → Fig. 1.2.[1] The code for the page, on which more below, is passed to the web server, which sends it as a *response* to the *client*, i.e. to the browser on the computer where the request originated. The response sent by the web server also follows the rules of the HTTP(S) protocol and again contains meta information (the response headers) beside the returned content proper. Note that the terms *client* and *server* are also used to refer to the computers on which client or server programs run. In the present example, we may say that the device with the web browser is a client that is making a request to the web application on the server computer with the IP address 93.184.216.34.



disproof

[dɪsˈpruːf] *n.*

1. facts that disprove something
2. the act of disproving

See also: disprove

Fig. 1.2: Minimal example of the view of an entry in an Internet dictionary.

1.2.2 HTML, CSS, and JavaScript

Yet how exactly does a web application, in its response, describe a website to a client, that is, to a web browser? The description is written in a particular language, namely Hypertext Markup Language (*HTML*). The central task of the browser is to transfer this description into the required presentation (to *render* the HTML source code, usually on a screen). The mini web page with the entry on *disproof* looks as follows:

1 This example draws on one given in the “Guidelines for Electronic Text Encoding and Interchange” (TEI 2023) on the dictionary module of the Text Encoding Initiative. The example is also used in Chapter 4.4.1 on TEI.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>MyEnglishDict</title>
   </head>
   <body>
      <h1>disproof</h1>
      <p>[dɪs'pruːf] <i>n.</i></p>
      <ol>
         <li>facts that disprove something</li>
         <li>the act of disproving</li>
      </ol>
      <p><i>See also:</i> <a href="/entry/disprove">
      disprove</a></p>
    </body>
</html>
```

The basic idea behind HTML is a strictly descriptive and hierarchically structured markup of sections of text by means of structuring information in angular brackets, so-called *tags*. Thus, the word *disproof* is marked here as a heading at the first – i.e. the highest – organisational level, by virtue of a *start tag* <**h1**> (meaning: "level 1 heading") placed in front of the word and a corresponding *end tag* </**h1**> after the word. End tags are marked by a forward slash immediately after the opening angular bracket. How exactly this structural information is rendered is a matter for the browser. Headings at level 1 are usually represented in a larger, boldface font on a separate line. The example code for the miniature web page contains further illustrations of typical HTML tags:

– a *paragraph* of text: <**p**> . . . </**p**>;
– a span of text "in an alternate voice or mood",[2] usually rendered in *italics*: <i> . . . </i>;
– an *ordered list*: <**ol**> . . . </**ol**>;
– a *list item* in that list: <**li**> . . . </**li**>;
– the web page *title*, shown in a browser's title bar or a page's tab: <**title**> . . . </**title**>;
– a hyperlink (*anchor*): <**a href**=" . . . "> . . . </**a**>. Here, the text that is actually shown in the browser is placed between the start and end tags (recognisable as a link in → Fig. 1.2 by underlining and a different colour), and the URL (or URL path) for the web page that is brought up by clicking on the link is given as a so-

2 From the HTML specification, 4.5.20 "The i element", https://html.spec.whatwg.org/#the-i-element.

called *attribute*, an additional piece of information, inside the start tag (**href** stands for *hypertext reference* and is the *name* of the attribute; the text in quotes, here the URL path, is its *value*).

Further tags structure the HTML document as a whole; thus, the whole document has to be enclosed in the <**html**> . . . </**html**> pair of tags. The initial line <**!DOCTYPE html**> is the *document type declaration* and has a special syntax; it marks the document as being written in the current version of HTML, which is HTML5. The actual content of the page to be shown in the browser is the "body" of the document and is marked by <**body**> . . . </**body**>. Core information about the web page is found in the "head" of the document and is indicated by <**head**> . . . </**head**>: in the example above, the head only contains the **title** of the page, which is shown in the tab of the browser window, plus information about the so-called *text encoding* used in the document, that is, the set of characters used and how each character is represented by a certain number. UTF-8 encoding is the most widely used encoding today, covering the characters of most of today's writing systems and being part of an ongoing standardisation effort known as the Unicode Standard. A start tag and an end tag, together with all of the content between them, represent what is known as an *element* in HTML. The tags indicate the *name* of the element while the content of the element consists of text and/or subordinate elements. Some elements cannot have content. The **meta** tag that is used here to specify the character encoding is one such *void* element; therefore, as a special syntax rule in HTML5, it must not have an end tag. It still conveys information, though, through its attribute **charset** (i.e. 'character set').

HTML code describes only the textual structure of a web page in a hierarchically structured way. Normally, HTML is combined with two further languages: The graphic and colour structure of the content is described using *Cascading Style Sheets* (CSS), including more complex aspects like animations and the definition of different presentations of site content, for example, on printers or small screens.

JavaScript is a programming language available on all modern browsers through which all of the interactive processes of a web page can be implemented directly in the browser, including comprehensive manipulation of graphics, data processing, and communication with other computers on the Internet, etc.

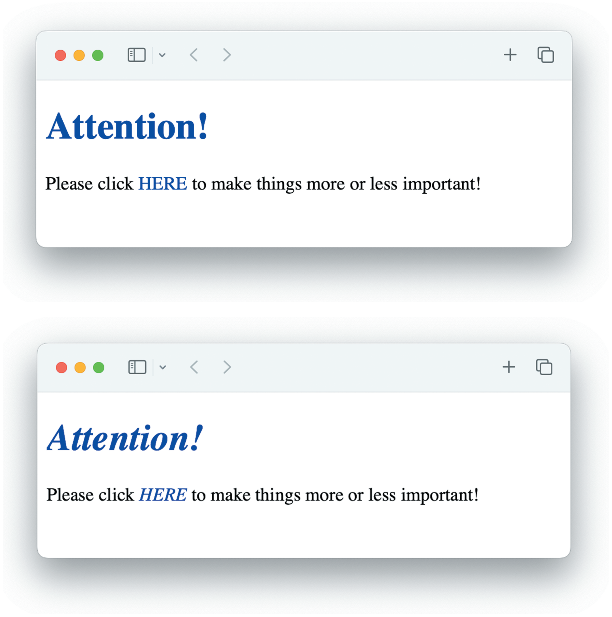
At this point, we have to be content with a miniature example to illustrate some basic ideas. In the following HTML code, which can be tested directly with a browser, CSS and JavaScript code is integrated directly:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>CSS and JavaScript Demo</title>
      <style>
```

```
            .teaser {
               color: blue;
            }
            .alert {
               font-style: italic;
            }
         </style>
      </head>
      <body>
         <h1 class="teaser">Attention!</h1>
         <p>
            Please click
            <span onclick="toggleEmphasis()"
class="teaser">HERE</span>
               to make things more or less important.
          </p>
          <script>
             function toggleEmphasis(){
                for (teaserElement of
document.getElementsByClassName("teaser")) {
                   teaserElement.classList.toggle("alert");
                }
             }
          </script>
      </body>
</html>
```

Two HTML elements have a **class** attribute with the value "teaser": the **h1** heading and a **span** element, which simply delimits a stretch of running text containing the text *HERE*. The **class** attribute assigns the *CSS class* 'teaser' to these elements. Such a CSS class is simply a kind of custom marker that can be used to define presentation-related aspects pertaining to the elements it is assigned to. In our example, this definition is done in the **style** element. We will not discuss the finer points of the CSS language here but the first CSS 'instruction' basically says that any element marked with the 'teaser' class gets a 'blue' text colour, where the predefined keyword 'blue' actually represents a certain, pure shade of blue. As a result, both elements with the 'teaser' class are indeed rendered blue by the browser, as shown in the screenshot in → Fig. 1.3. If the user clicks on the word *HERE*, all of the blue text is additionally italicised; on clicking again, the italics are removed again. This interactive behaviour is governed by the **onclick** attribute of the *HERE* **span**: If the user clicks somewhere on the text inside the **span**, the JavaScript *function* 'toggleEmphasis', which is defined in the **script** element, is executed. A function is basically a block of programming code.

The code in the ‘toggleEmphasis’ function looks at each HTML element with the class ‘teaser’ and assigns to it – or removes if already present – the CSS class ‘alert’ that, according to the CSS code in **style**, triggers italic text.



Attention!

Please click HERE to make things more or less important!

Attention!

Please click *HERE* to make things more or less important!

Fig. 1.3: Toy example of an interactive web page using CSS and JavaScript. If the user clicks on *HERE*, all of the blue text becomes italic; on clicking again, the italics are removed.

The CSS and JavaScript code is normally put in separate files so that the design and interactivity of the web page can, as far as possible, be maintained independently of the textual content. In this way, you could alter the colour and the interactive behaviour of *HERE* just by modifying these external files, which must be retrieved by the browser from the web server using dedicated URLs. In our example, the code inside the **style** and the **script** elements could alternatively be put in text files **mystylesheet.css** and **myscript.js**, respectively and referenced as follows in the head of the HTML code:

```
<link rel="stylesheet" href="/stylesheets/mystylesheet.css">
<script src="/scripts/myscript.js"></script>
```

While processing this HTML code, further HTTP(S) requests are initiated to retrieve the referenced files. This is also a common way of integrating multimedia content: The HTML element

```
<img src="/images/mypicture.jpg">
```

initiates a new request to load the image with the URL path "/images/mypicture.jpg" from the server. In this way, a request for a single complex HTML web page can prompt dozens of additional requests to download further data that are needed to render the page and enable its functionality.

Linking HTML documents to separate, external CSS and JavaScript files allows web applications to be developed in a modular way. Developers can take advantage of a huge number of software *libraries*, often freely available as open source software. In the realm of web page development, libraries are essentially just CSS and/or JavaScript files that assist in the complex task of creating standards-compliant web design, providing ready-made interactive visual components for web pages, or simplifying the implementation of complex functionality. *CSS frameworks* provide a large number of predefined CSS classes that developers can use in their HTML in order to achieve a professional and consistent look, including what is known as *responsive web design* that adapts the page layout automatically to different screen sizes and device types. So-called *front-end frameworks* have simplified web development considerably by improving on the approach that was used in our toy example: roughly speaking, instead of writing 'imperative' code that, depending on external circumstances, explicitly changes the structure of the web page and the properties of its elements, the programmer describes 'declaratively' what the page should look like depending on a set of data that defines the overall 'state' of the page.

1.2.3 Outlook

As we have seen, the web application sends HTML code over the Internet, in response to the request from the client, to the web browser where it is rendered. Interested users can trace the process described here in detail at any time on their own computer. On the one hand, browsers usually offer the option of displaying the HTML code of a page (often referred to as source code). On the other hand, most modern browsers assist programmers with inbuilt developer tools that, for example, let you view the exact content of the HTTP(S) request and response and even show details such as how long it took to determine the IP address of the web server by consulting a name server.

But where does a web application take the HTML code for an entry? Generally, this code does not remain fixed and complete ("static") on the hard drive of the web server but rather is built "dynamically" from abstract lexicographic data structures only when the client request is answered. This is explained in more detail in → Chapter 4.

Finally, it is important to emphasise that we have only examined in some detail the simplest example of web content being accessed, namely the "classic" request-response cycle, in which the user initiates a browser request to the server with an operating action, such as a mouse click on a link, thereby subsequently fetching a new HTML web page through the server's response. The limitations of this approach can be overcome in different ways. Here are three important examples:

– With a set of technologies collectively known as *Ajax*, the program code (JavaScript) on a web page can request data from a web server via HTTP(S) asynchronously, that is, in the background and without blocking any additional user interaction on the web page. The code can then process these data and modify the content of the page in any way necessary. In complex applications, this removes the slow and non-intuitive loading of a whole new web page, for example, after a button or icon has been clicked. Thus, navigating Ajax-based websites approaches the user experience of conventional desktop applications.
– A web server can deliver data to a client in the HTTP(S) protocol only when a corresponding request has previously been made by the client. The *WebSocket* protocol makes genuine bidirectional communication between the client and server possible such that a server can send data at any time to clients "of its own accord" following a particular event. For example, whenever some participant in an online chat posts a message, everybody else should receive an update of the chat history immediately; with pure HTTP(S), the only way to implement this would be through letting the browser issue requests every few seconds in order to check for updates.
– A lexicographical web application on a server can do much more than just deliver HTML pages (and associated web resources) to web browsers. There are many types of client applications that may need to use lexicographical data. Typical examples are dictionary apps on a mobile device such as a smartphone or tablet or, more generally, word processors, language learning apps, language-related games, or programs that researchers implement to process large amounts of data. Such programs might fetch lexical information from a server on a per-need basis, if storing a complete lexicographical database on the computer itself is not a viable option. In many cases, the data delivered by the server are not formatted in HTML. Instead, formats are used that are better suited for machine processing than for direct rendering in a web browser. Server applications that serve these types of clients are typically called *web services*. A common scenario for the exploitation of such web services is an aggregating server for lexicographical content that itself draws its data from a range of lexicographical resources located on other computers around the world. A request to the server about a specific word would lead to the server fetching relevant pieces of information from all of the other computers, bundling them, and then forwarding them to the client. This approach is called a *federated search*.

In all of these cases, a set of rules is needed which a client program can use to retrieve or even modify data on a specific server. This rule set or protocol is called the server's *API* (application programming interface). It has become widespread practice to simply use HTTP(S) for that purpose: individual resources – e.g. dictionary entries or collections thereof – can then be accessed through specific URLs using a variety of access modes, including the GET mode mentioned above. The machine-readable content returned in the response is typically delivered by the server in a highly structured data format, such as XML (→ Chapter 4) or JSON (a notation for structured data that JavaScript can directly understand and process), instead of HTML.

Even in the classic case of an online dictionary running in a browser, it would not be unusual to use the Ajax approach sketched above, using JavaScript code to fetch the lexicographical data currently requested by the user through an API provided by the server. The data obtained this way are then processed by JavaScript code on the web page to construct HTML elements that are inserted into the currently shown web page in order to render a human-readable view of the entry without even loading a completely new web page.

In all of this, the increasingly ubiquitous availability of the Internet is erasing the boundaries between online and offline content. Specifically, this could mean that a core set of data is available on a local device while an application can automatically search online for updates and other associated content depending on the availability of an Internet connection – without the user knowing the origin of the data.

1.3 Logging

In what follows, the term "logging" summarises, very generally, the recording of information about the internal state of a technical system as well as the interaction of users (or of other technical systems) with the system.

The information recorded is stored as *log data* in the form of individual datasets (often called "records"), usually with an exact timestamp so that the chronological sequence of actions and the state of the system can be reconstructed for relevant aspects. Additional metadata may supplement these datasets, for example, a classification of the meaning of the datasets (debug information, warning, serious error, etc.) or the name of the system component that generated the dataset. The log data can be grouped and filtered using this metadata to allow for better informed analyses of the system's behaviour, e.g. for debugging purposes.

For an Internet dictionary, two technical systems that generate log messages are of principal interest: the actual dictionary web application and the web server through which the web application communicates with computers making requests. In the concrete technical realisation of the overall system, both can also be subsystems of a single integrated system. In the following illustration, we shall proceed like in → Section 1.2

from the latter situation in order to be able to provide a concise overview. As such, we shall treat the Internet dictionary as a monolithic (server) system that communicates with a human user, mediated through their web browser. In a process of communication of this kind, data are transferred by a variety of protocols.

During interactions with web applications, metadata are inevitably created as an integral part of the protocols that the interactions rely upon. In addition to the requested URL, each request to the Internet dictionary involves a whole range of further information being transferred by the web browser in different HTTP(S) headers, for example:

– its own IP address (*Host*),
– an identification of the browser type (*User-Agent*),
– preferred data formats for direct display (*Accept*),
– preferred language (*Accept-Language*),
– the URL of the last retrieved page (*Referer*),
– a wish (not) to leave a user profile on the server (*DNT*, "do not track").

When retrieving a URL – for example, by entering a search term in a search field or by clicking on a link – different *parameters* may be sent to the server. In this way, the values that users have entered in a form on a web page (date of access, search criteria, personal settings) can be transferred to the web application. Depending on the method of the HTTP(S) request being used, either these parameters appear in a so-called query string as part of the URL in the address line of the browser (GET method) or they are sent opaquely for the dictionary user as part of the actual HTTP(S) message (POST method).

The dictionary web application can also send further data to the browser in addition to the information explicitly requested by the user. These data – so-called *cookies* – are stored locally by the browser and transferred back, on request, and usually unnoticed. Often cookies serve to identify the user through a unique token, typically after they have registered on a website to have a list saved of the entries they have already searched for or other information that has to be made available once another page has been retrieved. As such, cookies can be understood in many cases as a form of logging in the browser, but with the particular characteristic that these logged datasets can be evaluated by the dictionary web application itself while it is running. There are a variety of processes by which to send cookies to a browser, for example, using the HTTP(S) protocol.

The two most important uses for log information are, first, to analyse problems when program errors or general technical errors occur in the functioning of the dictionary application and, second, to analyse user behaviour and their interactions with the dictionary web application. The analysis of technical problems will not be discussed further here since it depends very strongly on the specific implementation of particular web applications. However, a whole chapter in this volume is dedicated to the analysis of user behaviour (→ Chapter 9). For that reason, the emphasis in what follows will be more on details about the type of metadata that can be gained for this purpose.

The dictionary application communicates first of all with a technical system that is identifiable through its IP address (HTTP(S) 'Host' request header). However, these IP addresses are often allocated dynamically and are, therefore, not associated permanently with a particular device. Many devices also operate behind a so-called shared gateway, which, for example, processes the whole outgoing communication of an organisation through a single IP address. In this way, a simple reference to an IP address does not make it possible to reliably identify a particular device (and therefore a single user). This uncertainty can be countered in part through further log information. In addition to the client's IP address, information about the type of the user's browser can be taken into account. Conclusions can also be drawn from the URL of the last page visited and the time of retrieval. While this approach generally works well for small groups of users sharing a common IP address, it often fails to reliably identify users from larger groups. For some research questions on user interaction, it may not be necessary, though, to actually identify specific users. Instead, it may suffice to focus on the behaviour of groups of users (identified as a group by common metadata features) or on single-step interactions such as the consecutive retrieval of two pages regardless of which user interacted in this scenario.

Reliable observations of a specific user (*tracking*) become possible when the user has registered on the dictionary application (i.e. they are assigned a unique identifier) or when the application silently assigns a unique identifier (e.g. a cookie) to the browser used. Using either of these unique identifiers, all of the interactions of the user can be read from the log data as long as the identifiers are stored in the logs.

Of course, not everything that is technically possible in terms of tracking users is legally permitted. For example, the specific tracking of user behaviour outlined above is generally not allowed in the European Union without the explicit and conscious consent of the user. Various legal regulations describe and limit the types of communication data gathered and their use, above all:

– the EU's General Data Protection Regulation (Regulation 2016/679, GDPR),
– the EU's Privacy and Electronic Communications Directive (Directive 2002/58/EC, ePrivacy Directive),
– data protection acts in EU member states.

In an institutional context, there are often additional and, in part, more specific provisions and guidelines (based on the aforementioned laws) determining which interaction data can be legally and ethically stored and analysed as log data and in which form. There are also appointed individuals with mandates for data protection who can provide help and support.

1.4 Versioning

The reader of a print dictionary is not dependent as a matter of principle on the support of a technical system to be able to use the storage medium of the book. However, the perception of a dictionary that appears in electronic form is not possible without recourse to a suitable device to display it and to navigate through it. The specific type of device – whether it is an electronic translation device, a mobile phone, or, more generally, a computer system – plays no role in our considerations here. What is important is the basic principle common to them all, namely that the presentation of the stored information that can be read by people always has to be generated first from the stored representation of the data. Unlike a book, the content of which is fixed and immutable after the printing process, the underlying data that are stored electronically can be changed dynamically or be replaced relatively easily. The display device will then show the user the updated information (e.g. a revised dictionary entry). There is a series of processes and technologies designed to deal with the new challenges arising from this variability, which will be presented briefly in this section and in → Section 1.5. We begin with the problem that systematic access to different versions of dictionary entries needs to be possible for dictionary creators and dictionary users alike.

Even if this so-called versioning is not a specific Internet technology or a widespread concept in lexicography, it does play a certain role in Internet lexicography, which justifies our treatment of the ideas that lie behind it.

Versioning of digital data is an idea that originated in software development. There, we talk of the life cycle of a piece of software. A program is developed, tested, launched on the market, used, and revised. The revision results in various versions of the same program. The "life" of the program comes to an end when its further development and support are discontinued – which does not mean that the program is no longer in use. In software development, tools that support the managing of versions – in particular of a program's source code – have the following purposes and functions: all changes are recorded and possibly commented on, as appropriate, and earlier stages of development (versions) of the software are archived automatically. It is then possible to go back to them as needed (cf. Baerisch 2005).

The idea of a life cycle has been transferred to documents in the digital world (cf. Lobin 2004). The typical document conceived in this way is a product description or instruction manual that keeps pace with the further development of the product; that is, it must be adapted without being written completely afresh. In this case, we can speak of multiple versions of this document that have to be managed so that the authors of the document retain an overview of them.

In the world of printed texts, there is a comparable concept: the *edition*. A text can appear in several editions. It can be reproduced unchanged from edition to edition but it can also be changed to a greater or lesser extent. The authors usually give very brief information in a foreword about the changes to the text that characterise the new edition.

The most important differences between an edition (of a book) and a version (of software or a document) are as follows:

- The time period between two editions normally amounts to one or more years. The gaps between two versions of a piece of software or a document accessible online are typically considerably shorter.
- The scope of an edition usually extends to the whole printed work (the book is in its fifth edition); in software development, a whole software package can be versioned but also a single module. As far as documents are concerned, the versioning may apply to single chapters, sections, or – in the case of lexicographic works – entries, or even just parts of entries.
- The documentation of changes in a new version (the so-called "change log") is usually more detailed than the "foreword to the new edition" in a book.

Internet lexicography involves two different types of documents and two different types of "users". On the one hand, it is possible to view a whole dictionary as a single document; on the other hand, a single module, typically a dictionary entry, but also supporting texts can each be conceived as individual documents. These different perspectives on granularity correspond to different user perspectives: while dictionary users will generally consider a dictionary as a fixed set of entries, for lexicographers the focus is often on a single entry (and possibly on closely related entries) and on the entry's individual stages of development over time.

As a result, the following applies to managing versions when compiling an Internet dictionary:

- It makes sense for lexicographers to version at the level of individual entries. Indeed, this is necessary when several lexicographers are working on the same entry. It has to be possible to recreate older stages of development of an entry and to compare the different versions with one another. A particular version has to be accessible with an unambiguous name or identifier, for example, a version number. Further metadata can be helpful in addition to this name, including the name of the individual who created this version, the time at which the version was created, and a description of the change to this version compared to the previous one.
- For the user of the "finished" product, i.e. a dictionary, which is typically accessed through a browser, versioning at the level of the whole work is often sufficient. In this case, a description should be provided of the important changes from the previous version – and cumulatively from the version before last, and so on. An indication of when this version was made available is also helpful. Corresponding supporting texts should also belong to the "product contents" of the dictionary. As a rule, there is no expectation that earlier versions of the dictionary or individual entries should be accessible since the resources required for that are very exten-

sive. So-called "wikis" (e.g. Wikipedia and Wiktionary) remain the exception. The version management of individual entries is an integral part of these systems and, thus, is available to all users (on the grounds that users can, in principle, also create or edit entries; → Chapter 8).

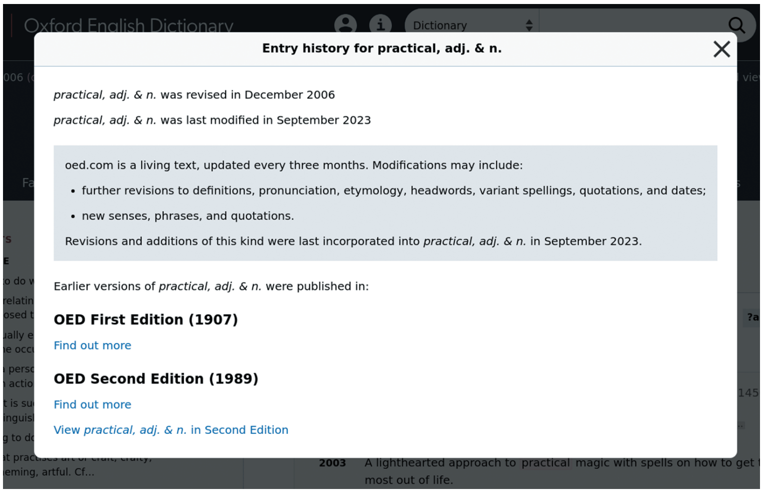
From this, we can derive the following recommendations for the planning of a dictionary to be published on the Internet.

Whenever a lexicographic process is to be planned or undertaken (→ Chapter 3), fundamental decisions need to be taken about data management. One of these decisions is whether different instances of a document should be created at the individual stages of the lexicographic process and how they should be dealt with. With every new instance of a document, previous iterations can either be discarded or conserved. If a database management system is to be employed for conserving data (→ Chapter 4.2.2), then it is important to be aware that this type of software does not automatically support the management of different versions. Each change to the stored data overwrites the previous version. For version management in these cases, the storing of data has to be conceived in such a way that any changes made to the data result in new datasets with appropriate metadata instead of simply modifying existing datasets. However, this requires greater technical effort, which has to be taken into account when planning the project. Alternatively, it is possible to employ a wiki system, where, as we have seen, version management is already built in. A third alternative is to combine an editing system with a version control system (VCS), as is typically the case in software development. Subversion (https://subversion.apache.org/) and Git (https://git-scm.com/) are common examples of such VCS among many others. Using dedicated VCSs can require extensive technical knowledge. For example, so-called "version conflicts" may have to be resolved if the VCS does not provide exclusive locking of resources to prevent two lexicographers from simultaneously and independently working on the same entry. Should version conflicts arise, they would have to be resolved in such a way that a single common version exists after merging the conflicting entry versions.

Editing cycles also have to be taken into account when planning the publication of Internet dictionaries. Here, wiki systems are again the simplest solution. Each change is immediately visible online, and changes can be undone relatively easily. However, the process of checking can be very complicated and time-consuming when a large number of changes are involved. In online reference works that are compiled with a limited set of editors, a dictionary entry will only be published after rigorous checking and approval. Updates to individual entries may not be made public as soon as they are approved. Rather, updates to the dictionary could be made in bulk at certain intervals. The model of the edition in print lexicography is an extreme case of this: the whole work is published afresh after a period of several years or even decades. The other extreme is the publication of individual updated and approved entries. The practice for updating most online dictionaries will probably lie somewhere in between. Possible options are updates at particular intervals (e.g. monthly or weekly) or when a specific number of approved entries, either new or revised, are ready for publication.

Finally, we provide two examples of versioning in large online lexicographic projects:

– OED ONLINE: in the online version of the "Oxford English Dictionary" (https://www.oed.com/), a link is provided in the top left corner of each entry stating the most recent year of its revision. When following the link, a more detailed summary of the entry's revision history appears, summarising all major revisions and the last minor revision of the entry. The information in the revision history refers either to editions of a volume (i.e. "OED First Edition 1907"), to one of the supplementary (published) volumes, or to an "online version" (→ Fig. 1.4). We are unaware of the exact internal version management practised by the OED editors.
– DWDS: the "Digital Dictionary of the German Language" ("Digitales Wörterbuch der deutschen Sprache") is conceived as a lexical information system that encompasses several dictionaries, linguistic corpora, and statistical tools (cf. Klein/Geyken 2010). Some of these dictionaries are regularly updated, including the "Etymological Dictionary of the German Language" ("Etymologisches Wörterbuch des Deutschen", also known as the PFEIFER-DWDS). Work on the print version has long since been completed, with three editions published between 1989 and 1995. However, the principal author, Wolfgang Pfeifer, worked continuously on revising existing entries and compiling new ones for the online version until his death in 2020. Until that point, his dictionary in the DWDS was updated around twice a month, with each version of the PFEIFER-DWDS being assigned its own version



Entry history for practical, adj. & n.

practical, adj. & n. was revised in December 2006

practical, adj. & n. was last modified in September 2023

oed.com is a living text, updated every three months. Modifications may include:

- further revisions to definitions, pronunciation, etymology, headwords, variant spellings, quotations, and dates;
- new senses, phrases, and quotations.

Revisions and additions of this kind were last incorporated into *practical, adj. & n.* in September 2023.

Earlier versions of *practical, adj. & n.* were published in:

OED First Edition (1907)

Find out more

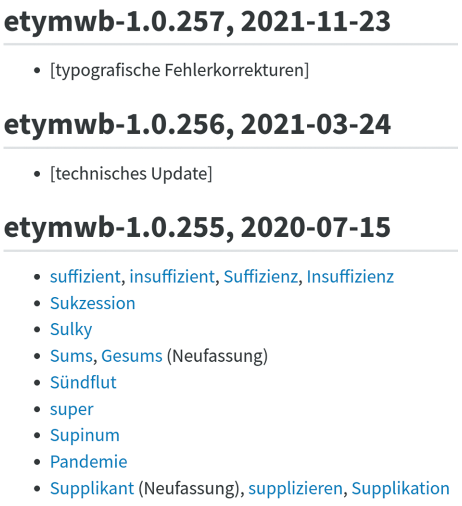
OED Second Edition (1989)

Find out more

View *practical, adj. & n.* in Second Edition

Fig. 1.4: Publication history of the entry for *practical* in the OED.

number. The version number has three parts so that more significant changes can be distinguished from more minor ones. A change in the format of the data (e.g. more detailed tagging of information within the entries) led to a new version, even if these changes might not always be visible to the user. All versions are archived and can be retrieved as necessary, although this option is not offered on the website. The change log for this resource is documented on a separate page (https://www.dwds.de/wb/etymwb/changes; → Fig. 1.5). An editing system is used in the DWDS to compile the entries of the main dictionary, which relies on the Git VCS. In this way, all versions of an entry are automatically archived and can be consulted on demand.



etymwb-1.0.257, 2021-11-23

- [typografische Fehlerkorrekturen]

etymwb-1.0.256, 2021-03-24

- [technisches Update]

etymwb-1.0.255, 2020-07-15

- suffizient, insuffizient, Suffizienz, Insuffizienz
- Sukzession
- Sulky
- Sums, Gesums (Neufassung)
- Sündflut
- super
- Supinum
- Pandemie
- Supplikant (Neufassung), supplizieren, Supplikation

Fig. 1.5: Version information for the Pfeifer-DWDS.

1.5 Persistence and identity

Moving beyond project-internal version management, the possibility of updating the data available on the server at any time raises questions about the longevity (*persistence*) and identity of electronic data. The term persistence designates the property of an object to remain unchanged over a long period of time. This property applies to print dictionaries by their very nature. The carrier medium of paper can last for several centuries when stored appropriately without the fixed written form changing. However, the usual storage media for electronic documents typically demonstrate a much shorter lifespan. In part, this is due to the materials used. Chemical changes can occur in the synthetic materials that are used as the carrier medium or as a protective layer (e.g. in magnetic tapes and optical storage media, such as CDs or DVDs). The storage cells in some non-volatile semiconductors (e.g. flash storage in memory sticks and cards) degenerate when they are written onto; their likelihood of failure increases every time they are written onto. For this reason, this type of semiconductor storage has integrated components for recognising and correcting errors as well as reserve storage for rescuing data from defective areas.

Understood more broadly, the concept of persistence can also be applied to the processes used for storing data, that is, to the technological methods, tools, and agreed conventions for storing and receiving data on the storage medium. In the case of a printed or handwritten book, the process consists of mechanically fixing the (agreed) written symbols on paper (writing, printing, stamping with an appropriate tool) and the direct recognition of these written symbols (optical and haptic reading). In electronic storage processes, storage and reception take place with the aid of technical devices. Consequently, stored data cannot be accessed by humans without technological tools. This results in the persistence of the storage process being strongly dependent on the availability of the necessary storage and reading devices as well as on the data encoding used (that is, the representation conventions agreed upon) being supported by the display device. In order to be able to read an old magnetic tape with typesetting instructions for a particular dictionary, not only does the magnetic tape need to be available (and as intact as possible) but also a suitable tape reader and, in some circumstances, a further device or program to extract the typesetting data from the data stream of the tape player.

As a rule, no detailed distinction is made between the aspects of persistence outlined above. Instead, the term persistence generally refers to the theoretical and temporally non-specific availability and usability of a dataset. Here, there is a tendency to abstract from the specific storage technology being used (storage medium and process). In particular, the storage technology actually used on the server side is ultimately irrelevant for the user accessing data over a network.

Because of the ease with which electronic data can be altered, it becomes possible to publish corrections, revisions, or new entries at any time; improve the access structures dynamically; or even extend the types of lexicographic information (→ Chapter 3). Nonetheless, if these possibilities are used as part of versioning (→ Section 1.4), this has far-reaching consequences for the way the dictionary and its parts can be cited. Depending on the time when they access it, a dictionary user will see a very particular version of a dictionary entry. In order to cite that version of the entry, they could provide the URL and the exact time at which the page was retrieved. Taken together, this information would represent a version-specific indicator. Nonetheless, both details are arbitrary: a URL is not a fixed indicator (it can, in theory, be changed by the provider of the dictionary at any time) and the time will, as a rule, be one of any number of times that all refer to the same entry version since edited entries are not updated that frequently. In addition, Internet dictionaries do not usually provide for a time-specific query that would make it possible to download the entry again in the form in which it appeared at a particular moment in time, unlike collaborative platforms like WIKTIONARY, as already explained, which offer a version history for every entry. In order to avoid the arbitrariness of the retrieval date, it is preferable to indicate the date and time on which the specific version of the dictionary entry was published.

Yet this does not resolve the problem of the URL lacking persistence. For this reason, there are now several services that provide *persistent identifiers (PIDs)*, including:

- DOI (Digital Object Identifier, https://www.doi.org/),
- ePIC handles (persistent identifiers for eResearch, https://www.pidconsortium.eu/),
- URN (Uniform Resource Name, e.g. URN:NBN at the German National Library, https://nbn-resolving.org/),
- PURL (Persistent Uniform Resource Locators, https://purl.archive.org/).

PIDs have the function of providing a stable abstract address for an electronic resource. The PID can be resolved in order to derive the actual address from the abstract one. This is done by looking up the correct allocation of the PID in a directory that lists the allocation of all PIDs to "traditional" URLs. As such, the persistence of a PID is based on the guarantee that the consortia or institutions concerned will ensure a reliable correspondence to "classical" URLs. The dictionary providers themselves must, in turn, take responsibility for the accuracy and accessibility of this URL if they wish to offer PIDs to their users. If they alter the URL for their dictionary, they must ensure that the corresponding mapping of PIDs to URLs in the PID directory also changes.

PIDs are agnostic when it comes to changes in the content of the resources to which they refer. They can be used to identify single versions of the whole dictionary in a persistent way, or an individual dictionary entry, or a dynamic resource, that is, one that changes with time or depending on context. In the case of individual entries, the same number of PIDs are needed as there are versions of an entry, with each PID referring to a single version of the entry. The version number of the entry (e.g. the date of publication) must be retained in the URL when it is resolved. In the case of a dynamic resource, only one PID is needed per entry. When it is resolved, the URL should lead to the most up-to-date version of the entry. A versioning of the entry can be provided on the web page for the Internet dictionary independently of the PID, as discussed in → Section 1.4; it would then not be possible to provide direct addresses for individual versions using a PID.

It is also possible to mix the two ways of working with PIDs that we discussed in relation to versioning and unique identification. Thus, a PID might refer to the dictionary as a whole in its current form while provision is made for individual entries to have version-specific PIDs to allow for more accurate citation.

Embracing an Internet archive such as www.archive.org brings forth a plethora of advantages compared to being solely reliant on persistent URLs. The major benefit lies in the preservation of web content over time. While persistent URLs may succumb to the inevitability of “link rot”, an Internet archive acts as a digital time capsule, capturing and storing historical versions of websites. This capability not only safeguards against broken links but also allows users to access and reference content that may have undergone alterations or been removed entirely from its original source.

In addition to mitigating the risks of link deterioration, Internet archives provide a robust solution for the continuity of access. Persistent URLs are effective only as long as the original website remains available. In cases of temporary downtime or permanent cessation, Internet archives often serve as a reliable alternative, ensuring that users can still retrieve valuable information from archived versions of web pages. This accessibility during downtimes contributes to the resilience of information dissemination and proves especially beneficial for researchers, educators, and the general public seeking reliable sources beyond the limitations of persistent URLs.

1.6 Concluding remarks

At the beginning of this chapter, we addressed the radical new possibilities available to producers and users of lexicographic products as a result of computer and network technology. These were set out with great clarity at a very early stage in their development (cf. de Schryver 2003). This also includes the observation that it makes little sense in many contexts to distinguish between the relatively traditional use of such products as a technological continuation of print dictionaries and other ways of using digital lexicographic resources. Thus, the following definition by Nesi (2000: 839) has remained valid for over two decades:

> The term *electronic dictionary* (or ED) can be used to refer to any reference material stored in electronic form that gives information about the spelling, meaning, or use of words. Thus a spell-checker in a word-processing program, a device that scans and translates printed words, a glossary for on-line teaching materials, or an electronic version of a respected hardcopy dictionary are all EDs of a sort, characterised by the same system of storage and retrieval.

Increasingly, lexicographic and encyclopaedic information about words is presented by search engines in a particular part of the screen – often next to links to corresponding entries in various Internet dictionaries – or it is even available at any time in applications, for example, by double-clicking on any word. Dictionaries are being used in ever greater numbers of applications from the field of natural language processing, unseen by users, including in language-learning and correction programs.

In all of these contexts, the authors of lexicographic works, together with their specialist authority, and the works themselves are becoming increasingly invisible to users. Googling words is now a frequent substitute for consulting a recognised dictionary. Even the physical handling of dictionaries is disappearing from our consciousness thanks to computer programs presenting results from the process of “consulting” digital dictionary data in a form that has already been further edited. In this way, the digital revolution of lexicography can be viewed not only from a technological perspective but also from a sociological one. As early as 1997, Niklas Luhmann was already offering a diagnosis in the context of computer-aided communication technology in his abstract systems-theoretical distinction between loosely coupled elements of a *medium* (Luhmann 1997: 309–310):

> Mit all dem ist die soziale Entkopplung des medialen Substrats der Kommunikation ins Extrem getrieben. In unserer Begrifflichkeit muß das heißen, daß ein neues Medium im Entstehen ist, dessen Formen nun von den Computerprogrammen abhängig sind. [With all this, the social decoupling of the medial substrate of communication is pushed to the extreme. In our conceptualisation, it must mean that a new medium is coming into being, the form of which is dependent on computer programs.]

Bibliography

Further reading

Fischer, Peter/Witt, Andreas (2014): Best Practices on Long-Term Archiving of Spoken Language Data. In: Ruhi, Şükriye/Haugh, Michael/Schmidt, Thomas/Wörner, Kai (eds.): *Best Practices for Spoken Corpora in Linguistic Research*. Newcastle: Cambridge Scholars Publishing, 162–182. *Despite the divergent thematic focus, a good overview of the problems of and possible solutions to the long-term storage and availability of language-related data with extensive further reading*.

MDN Web Docs. Mountain View, California, USA: Mozilla Foundation. Online: https://developer.mozilla.org/. *Very comprehensive and up-to-date open-source, collaborative project that documents web technologies. Alongside detailed technical reference documentation for experts, MDN web docs provides extensive learning resources for students and beginners. The documentation is primarily available in English, but translations of most documents are available in a number of large languages (at the time of writing, Chinese, French, Japanese, Korean, Portuguese, Russian, and Spanish).*

Literature

Academic literature

Baerisch, Stefan (2005): *Versionskontrollsysteme in der Softwareentwicklung*. Bonn. https://www.gesis.org/fileadmin/upload/forschung/publikationen/gesis_reihen/iz_arbeitsberichte/ab_36.pdf [last access: April 25, 2024].

de Schryver, Gilles-Maurice (2003): Lexicographers' Dreams in the Electronic Dictionary Age. In: *International Journal of Lexicography* 16, 143–199.

Klein, Wolfgang/Geyken, Alexander (2010): Das digitale Wörterbuch der deutschen Sprache (DWDS). In: *Lexicographica* 26, 79–93.

Lobin, Henning (2004): Textauszeichnungssprachen und Dokumentgrammatiken. In: Lobin, Henning/Lemnitzer, Lothar (eds.): *Texttechnologie. Perspektiven und Anwendungen*. Tübingen: Stauffenburg, 51–82.

Luhmann, Niklas (1997): *Die Gesellschaft der Gesellschaft*. Frankfurt am Main: Suhrkamp.

Nesi, Hilary (2000): Electronic Dictionaries in Second Language Vocabulary Comprehension and Acquisition: the State of the Art. In: Heid, Ulrich, et al. (eds.): *Proceedings of the Ninth Euralex International Congress, EURALEX 2000, Stuttgart, Germany, August 8th–12th, 2000*. Stuttgart: Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, 839–847.

TEI Consortium (eds.): *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Version 4.7.0. Last updated November 16, 2023, revision e5dd73ed0. TEI Consortium. https://www.tei-c.org/release/doc/tei-p5-doc/en/html/index.html [last access: April 15, 2024].

Dictionaries and reference works

DWDS = *Digitales Wörterbuch der deutschen Sprache. Das Wortauskunftssystem zur deutschen Sprache in Geschichte und Gegenwart. Ed. by Berlin-Brandenburgische Akademie der Wissenschaften*. Online publishing. https://www.dwds.de/ [last access: April 25, 2024].

OED Online = *Oxford English Dictionary online*. Ed. by Michael Proffitt. Online publishing. https://www.oed.com/ [last access: April 25, 2024].

Pfeifer-DWDS = *Etymologisches Wörterbuch des Deutschen. Digitalisierte und von Wolfgang Pfeifer überarbeitete Version im Digitalen Wörterbuch der deutschen Sprache*. Online publishing. https://www.dwds.de/d/wb-etymwb [last access: April 25, 2024].

Wikipedia = *Wikipedia, The Free Encyclopedia. San Francisco*. Ed. by Wikimedia Foundation. Online publishing. https://www.wikipedia.org/ [last access: April 25, 2024].

Wiktionary = *Wiktionary, the free dictionary*. Ed. by Wikimedia Foundation. Online publishing. https://en.wiktionary.org/ [last access: April 25, 2024].

Images

Fig. 1.1 “Nessie II”: https://upload.wikimedia.org/wikipedia/commons/thumb/c/cd/Nessie_II.JPG/800px-Nessie_II.JPG. Friflash, Wikimedia Commons, licensed under Creative Commons Attribution-ShareAlike 4.0, URL: https://creativecommons.org/licenses/by-sa/4.0/legalcode.