Introduction

1

Abstract

Health and Life studies are well known for the huge amount of data they produce, such as high-throughput sequencing projects (Stephens et al., PLoS Biol 13(7):e1002195, 2015; Hey et al., The fourth paradigm: data-intensive scientific discovery, vol 1. Microsoft research Redmond, Redmond, 2009). However, the value of the data should not be measured by its amount, but instead by the possibility and ability of researchers to retrieve and process it (Leonelli, Data-centric biology: a philosophical study. University of Chicago Press, Chicago, 2016). Transparency, openness, and reproducibility are key aspects to boost the discovery of novel insights into how living systems work (Nosek et al., Science 348(6242):1422–1425, 2015).

Keywords

Bioinformatics · Biomedical data repositories · Text files · EBI: European Bioinformatics Institute · Bibliographic databases · Shell scripting · Command line tools · Spreadsheet applications · CSV: comma-separated values · TSV: tab-separated values

Biomedical Data Repositories

Fortunately, a significant portion of the biomedical data is already being collected, integrated and distributed through Biomedical Data Repositories, such as European Bioinformatics Institute (EBI) and National Center for Biotechnology Information (NCBI) repositories (Cook et al. 2017; Coordinators 2018). Nonetheless, researchers cannot rely on available data as mere facts, they may contain errors, can be outdated, and may require a context (Ferreira et al. 2017). Most facts are only valid in a specific biological setting and should not be directly extrapolated to other cases. In addition, different research communities have different needs and requirements, which change over time (Tomczak et al. 2018).

Scientific Text

Structured data is what most computer applications require as input, but humans tend to prefer the flexibility of text to express their hypothesis, ideas, opinions, conclusions (Barros and Couto 2016). This explains why scientific text is still the preferential means to publish new

F. M. Couto, *Data and Text Processing for Health and Life Sciences*,
Advances in Experimental Medicine and Biology 1137,
https://doi.org/10.1007/978-3-030-13845-5_1

discoveries and to describe the data that support them (Holzinger et al. 2014; Lu 2011). Another reason is the long-established scientific reward system based on the publication of scientific articles (Rawat and Meena 2014).

Amount of Text

The main problem of analyzing biomedical text is the huge amount of text being published every day (Hersh 2008). For example, 813,598 citations[1] were added in 2017 to MEDLINE, a bibliographic database of Health and Life literature[2]. If we read 10 articles per day, it will take us takes more than 222 years to just read those articles. Figure 1.1 presents the number of citations added to MEDLINE in the past decades, showing the increasing large amount of biomedical text that researchers must deal with.

Moreover, scientific articles are not the only source of biomedical text, for example clinical studies and patents also provide a large amount of text to explore. They are also growing at a fast pace, as Figs. 1.2 and 1.3 clearly show (Aras et al. 2014; Jensen et al. 2012).

Ambiguity and Contextualization

Given the high flexibility and ambiguity of natural language, processing and extracting information from texts is a painful and hard task, even to humans. The problem is even more complex when dealing with scientific text, that requires specialized expertise to understand it. The major problem with Health and Life Sciences is the inconsistency of the nomenclature used for describing biomedical concepts and entities (Hunter and Cohen 2006; Rebholz-Schuhmann et al. 2005). In biomedical text, we can often find different terms referring to the same biological concept or entity (synonyms), or the same term meaning different biological concepts or entities (homonyms). For example, many times authors improve the readability of their publications by using acronyms to mention entities, that may be clear for experts on the field but ambiguous in another context.

The second problem is the complexity of the message. Almost everyone can read and understand a newspaper story, but just a few can really understand a scientific article. Understanding the underlying message in such articles normally requires years of training to create in our brain a semantic model about the domain and to know how to interpret the highly specialized terminology specific to each domain. Finally, the multilingual aspect of text is also a problem, since most clinical data are produced in the native language (Campos et al. 2017).

Biomedical Ontologies

To address the issue of ambiguity of natural language and contextualization of the message, text processing techniques can explore current biomedical ontologies (Robinson and Bauer 2011). These ontologies can work as vocabularies to guide us in what to look for (Couto et al. 2006). For example, we can select an ontology that models a given domain and find out which official names and synonyms are used to mention concepts in which we have an interest (Spasic et al. 2005). Ontologies may also be explored as semantic models by providing semantic relationships between concepts (Lamurias et al. 2017).

Programming Skills

The success of biomedical studies relies on overcoming data and text processing issues to take the most of all the information available in biomedical data repositories. In most cases, biomedical data analysis is no longer possible using an in-house and limited dataset, we must be able to efficiently process all this data and text. So, a common question that many Health and Life specialists face is:

[1] https://www.nlm.nih.gov/bsd/index_stats_comp.html
[2] https://www.nlm.nih.gov/bsd/medline.html

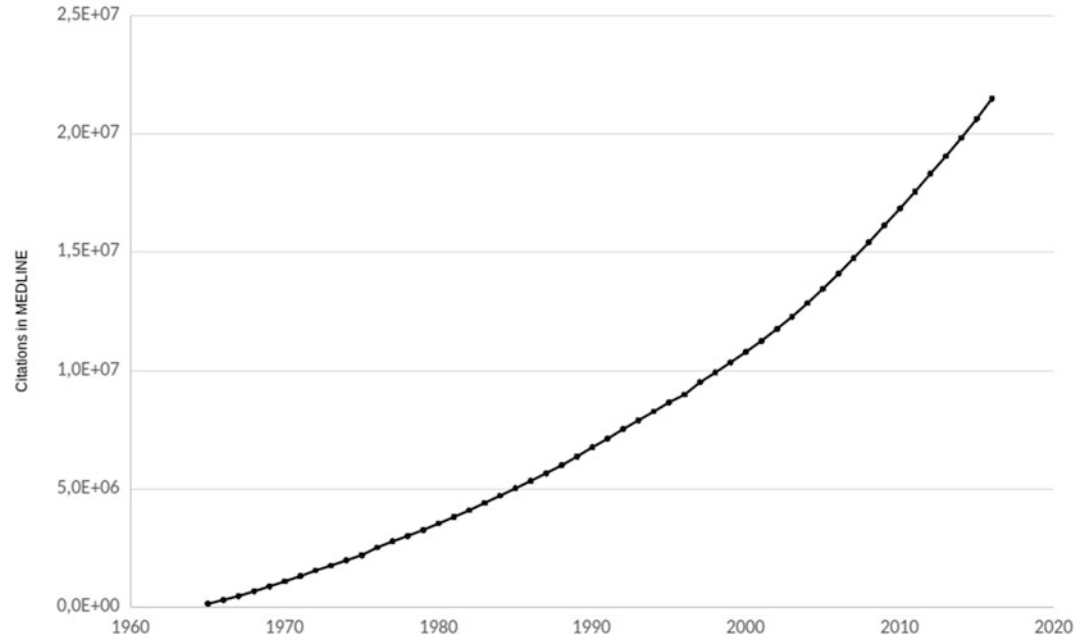

Fig. 1.1 Chronological listing of the total number of citations in MEDLINE (Source: https://www.nlm.nih.gov/bsd/)

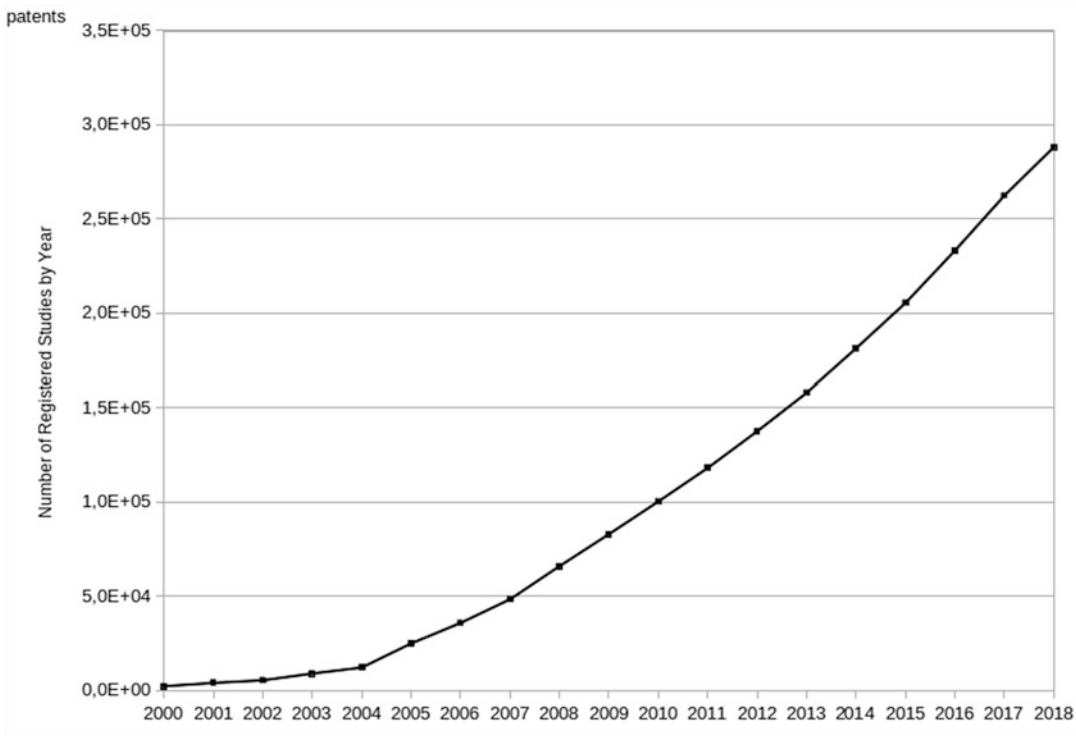

Fig. 1.2 Chronological listing of the total number of registered studies (clinical trials) (Source: https://clinicaltrials.gov)

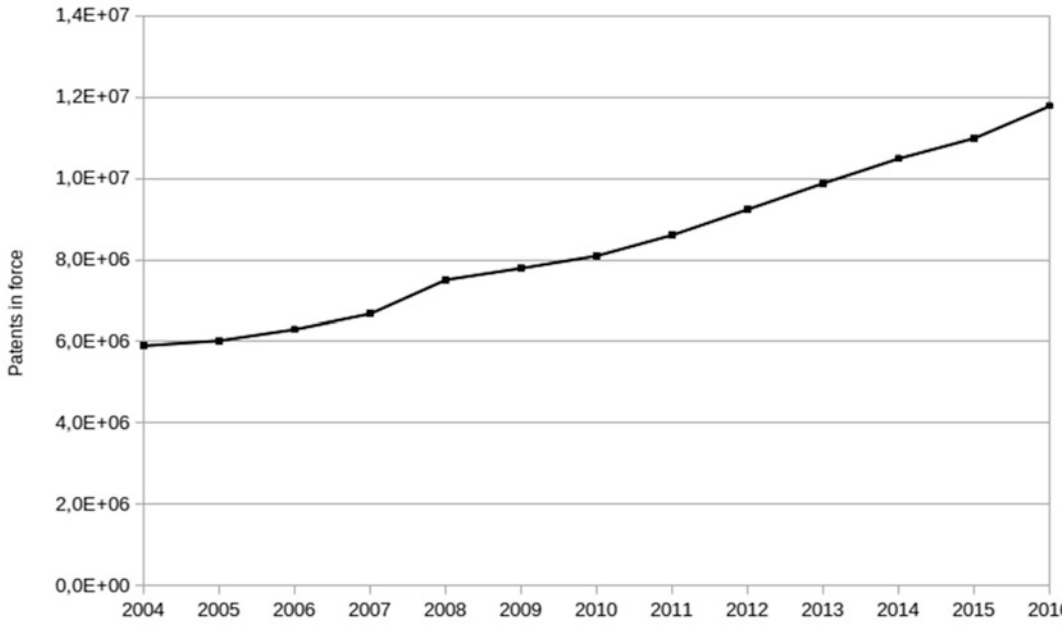

Fig. 1.3 Chronological listing of the total number of patents in force (Source: WIPO statistics database http://www.wipo.int/ipstats/en/)

How can I deal with such huge amount of data and text the necessary expertise, time and disposition to learn computer programming?

This is the goal of this book, to provide a low-cost, long-lasting, feasible and painless answer to this question.

Why This Book?

State-of-the-art data and text processing tools are nowadays based on complex and sophisticated technologies, and to understand them we need to have special knowledge on programming, linguistics, machine learning or deep learning (Holzinger and Jurisica 2014; Ching et al. 2018; Angermueller et al. 2016). Explaining their technicalities or providing a comprehensive list of them are not the purpose of this book. The tools implementing these technologies tend to be impenetrable to the common Health and Life specialists and usually become outdated or even unavailable some time after their publication or the financial support ends. Instead, this book will equip the reader with a set of skills to process text with minimal dependencies to existing tools and technologies. The idea is not to explain how to build the most advanced tool, but how to create a resilient and versatile solution with acceptable results.

In many cases, advanced tools may not be most efficient approach to tackle a specific problem. It all depends on the complexity of problem, and the results we need to obtain. Like a good physician knows that the most efficient treatment for a specific patient is not always the most advanced one, a good data scientist knows that the most efficient tool to address a specific information need is not always the most advanced one. Even without focusing on the foundational basis of programming, linguistics or artificial intelligence, this book provides the basic knowledge and right references to pursue a more advanced solution if required.

Third-Party Solutions

Many manuscripts already present and discuss the most recent and efficient text mining techniques and the available software solutions based on them that users can use to process data and text (Cock et al. 2009; Gentleman et al. 2004; Stajich et al. 2002). These solutions include stand-alone applications, web applications, frameworks, packages, pipelines, etc. A common problem with these solutions is their resiliency to deal with new user requirements, to changes on how resources are being distributed, and to software and hardware updates. Commercial solutions tend to be more resilient if they have enough customers to support the adaptation process. But of course we need the funding to buy the service. Moreover, we will be still dependent on a third-party availability to address our requirements that are continuously changing, which vary according to the size of the company and our relevance as client.

Using open-source solutions may seem a great alternative since we do not need to allocate funding to use the service and its maintenance is assured by the community. However, many of these solutions derive from academic projects that most of the times are highly active during the funding period and then fade away to minimal updates. The focus of academic research is on creating new and more efficient methods and publish them, the software is normally just a means to demonstrate their breakthroughs. In many cases to execute the legacy software is already a non-trivial task, and even harder is to implement the required changes. Thus, frequently the most feasible solution is to start from scratch.

Simple Pipelines

If we are interested in learning sophisticated and advanced programming skills, this is not the right book to read. This book aims at helping Health and Life specialists to process data and text by describing a simple pipeline that can be executed with minimal software dependencies. Instead of using a fancy web front-end, we can still manually manipulate our data using the spreadsheet application that we already are comfortable with, and at the same time be able to automatize some of the repetitive tasks.

> In summary, this book is directed mainly towards Health and Life specialists and students that need to know how to process biomedical data and text, without being dependent on continuous financial support, third-party applications, or advanced computer skills.

How This Book Helps Health and Life Specialists?

So, if this book does not focus on learning programming skills, and neither on the usage of any special package or software, how it will help specialists processing biomedical text and data?

Shell Scripting

The solution proposed in this book has been available for more than four decades (Ritchie 1971), and it can now be used in almost every personal computer (Haines 2017). The idea is to provide an example driven introduction to shell scripting[3] that addresses common challenges in biomedical text processing using a Unix shell[4]. Shells are software programs available in Unix operating systems since 1971[5], but nowadays are available is most of our personal computers using Linux, macOS or Windows operating systems.

> But a shell script is still a computer algorithm, so how is it different from learning another programming language?

[3] https://en.wikipedia.org/wiki/Shell_script
[4] https://en.wikipedia.org/wiki/Unix_shell
[5] https://www.in-ulm.de/~mascheck/bourne/#origins

It is different in the sense that most solutions are based on the usage of single command line tools, that sometimes are combined as simple pipelines. This book does not intend to create experts in shell scripting, by the contrary, the few scripts introduced are merely direct combinations of simple command line tools individually explained before.

The main idea is to demonstrate the ability of a few command line tools to automate many of the text and data processing tasks. The solutions are presented in a way that comprehending them is like conducting a new laboratory protocol i.e. testing and understanding its multiple procedural steps, variables, and intermediate results.

Text Files

All the data will be stored in text files, which command line tools are able to efficiently process (Baker and Milligan 2014). Text files represent a simple and universal medium of storing our data. They do not require any special encoding and can be opened and interpreted by using any text editor application. Normally, text files without any kind of formatting are stored using a *txt* extension. However, text files can contain data using a specific format, such as:

CSV : Comma-Separated Values[6];
TSV : Tab-Separated Values[7];
XML : eXtensible Markup Language[8].

All the above formats can be open (import), edited and saved (export) by any text editor application. and common spreadsheet applications[9], such as LibreOffice Calc or Microsoft Excel[10]. For example, we can create a new data file using LibreOffice Calc, like the one in Fig. 1.4. Then we select the option to save it as CSV, TSV, XML

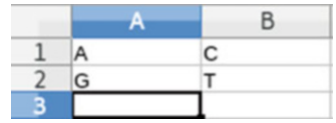

Fig. 1.4 Spreadsheet example

(Microsoft 2003), and XLS (Microsoft 2003) formats. We can try to open all these files in our favorite text editor.

When opening the CSV file, the application will show the following contents:

```
A,C
G,T
```

Each line represents a row of the spreadsheet, and column values are separated by commas.

When opening the TSV file, the application will show the following contents:

```
A C
G T
```

The only difference is that instead of a comma it is now used a tab character to separate column values.

When opening the XML file, the application will show the following contents:

```
...
<Table ss:StyleID="ta1">
<Column ss:Span="1" ss:Width="
    64.01"/>
<Row ss:Height="12.81"><Cell><
    Data ss:Type="String">A</Data
    ></Cell><Cell><Data ss:Type="
    String">C</Data></Cell></Row>
<Row ss:Height="12.81"><Cell><
    Data ss:Type="String">G</Data
    ></Cell><Cell><Data ss:Type="
    String">T</Data></Cell></Row>
</Table>
...
```

Now the data is more complex to find and understand, but with a little more effort we can check that we have a table with two rows, each one with two cells.

When opening the XLS file, we will get a lot of strange characters and it is humanly impossible to understand what data it is storing.

[6] https://en.wikipedia.org/wiki/Comma-separated_values
[7] https://en.wikipedia.org/wiki/Tab-separated_values
[8] https://en.wikipedia.org/wiki/XML
[9] https://en.wikipedia.org/wiki/Spreadsheet
[10] To save in TSV format using the LibreOffice Calc, we may have to choose CSV format and then select as field delimiter the tab character.

This happens because XLS is not a text file is a proprietary format[11], which organizes data using an exclusive encoding scheme, so its interpretation and manipulation could only be done using a specific software application.

Comma-separated values is a data format so old as shell scripting, in 1972 it was already supported by an IBM product[12]. Using CSV or TSV enables us to manually manipulate the data using our favorite spreadsheet application, and at the same time use command line tools to automate some of the tasks.

Relational Databases

If there is a need to use more advanced data storage techniques, such as using a relational database[13], we may still be able to use shell scripting if we can import and export our data to a text format. For example, we can open a relational database, execute Structured Query Language (SQL) commands[14], and import and export the data to CSV using the command line tool `sqlite3`[15].

Besides CSV and shell scripting being almost the same as they were four decades ago, they are still available everywhere and are able to solve most of our data and text processing daily problems. So, these tools are expected to continue to be used for many more decades to come. As a bonus, we will look like a true professional typing command line instructions in a black background window ! ☺

What Is in the Book?

First, the Chap. 2 presents a brief overview of some of the most prominent resources of biomedical data, text, and semantics. The chapter discusses what type of information they distribute, where we can find them, and how we will be able to automatically explore them. Most of the examples in the book use the resources provided by the European Bioinformatics Institute (EBI) and use their services to automatically retrieve the data and text. Nevertheless, after understanding the command line tools, it will not be hard to adapt them to the formats used by other service provider, such as the National Center for Biotechnology Information (NCBI). In terms of semantics, the examples will use two ontologies, one about human diseases and the other about chemical entities of biological interest. Most ontologies share the same structure and syntax, so adapting the solutions to other domains are expected to be painless.

As an example, the Chap. 3 will describe the manual steps that Health and Life specialists may have to perform to find and retrieve biomedical text about *caffeine* using publicly available resources. Afterwards, these manual steps will be automatized by using command line tools, including the automatic download of data. The idea is to go step-by-step and introduce how each command line tool can be used to automate each task.

Command Line Tools

The main command line tools that this book will introduce are the following:

- `curl`: a tool to download data and text from the web;
- `grep`: a tool to search our data and text;
- `gawk`: a tool to manipulate our data and text;
- `sed`: a tool to edit our data and text;
- `xargs`: a tool to repeat the same step for multiple data items;
- `xmllint`: a tool to search in XML data files.

Other command line tools are also presented to perform minor data and text manipulations, such as:

- `cat`: a tool to get the content of file;

[11] https://en.wikipedia.org/wiki/Proprietary_format

[12] http://bitsavers.trailingedge.com/pdf/ibm/370/fortran/GC28-6884-0_IBM_FORTRAN_Program_Products_for_OS_and_CMS_General_Information_Jul72.pdf

[13] https://en.wikipedia.org/wiki/Relational_database

[14] https://en.wikipedia.org/wiki/SQL

[15] https://www.sqlite.org/cli.html

- `tr`: a tool to replace one character by another;
- `sort`: a tool to sort multiple lines;
- `head`: a tool to select only the first lines.

Pipelines

A fundamental technique introduced in Chap. 3 is how to redirect the output of a command line tool as input to another tool, or to a file. This enables the construction of pipelines of sequential invocations of command line tools. Using a few commands integrated in a pipeline is really the maximum shell scripting that this book will use. Scripts longer than that would cross the line of not having to learn programming skills.

Chapter 4 is about extracting useful information from the text retrieved previously. The example consists in finding references to *malignant hyperthermia* in these *caffeine* related texts, so we may be able to check any valid relation.

Regular Expressions

A powerful pattern matching technique described in this chapter is the usage of regular expressions[16] in the `grep` command line tool to perform Named-Entity Recognition (NER)[17]. Regular expressions originated in 1951 (Kleene 1951), so they are even older than shell scripting, but still popular and available in multiple software applications and programming languages (Forta 2018). A regular expression is a string that include special operators represented by special characters. For example, the regular expression `A|C|G|T` will identify in a given string any of the four nucleobases adenine (A), cytosine (C), guanine (G), or thymine (T).

Another technique introduced is tokenization. It addresses the challenge of identifying the text boundaries, such as splitting a text into sentences. So, we can keep only the sentences that may have something we want. Chapter 4 also describes how can we try to find two entities in the same sentence, providing a simple solution to the relation extraction challenge[18].

Semantics

Instead of trying to recognize a limited list of entities, Chap. 5 explains how can we use ontologies to construct large lexicons that include all the entities of a given domain, e.g. humans diseases. The chapter also explains how the semantics encoded in an ontology can be used to expand a search by adding the ancestors and related classes of a given entity. Finally, a simple solution to the Entity Linking[19] challenge is given, where each entity recognized is mapped to a class in an ontology. A simple technique to solve the ambiguity issue when the same label can be mapped to more than one class is also briefly presented.

[16] https://en.wikipedia.org/wiki/Regular_expression

[17] https://en.wikipedia.org/wiki/Named-entity_recognition

[18] https://en.wikipedia.org/wiki/Relationship_extraction

[19] https://en.wikipedia.org/wiki/Entity_linking