Design patterns to improve Usability of Text Analytic Tools for Digital Humanists

Jan Nehring[1]

Abstract: Usability of text analytics tools is still a big issue. This paper presents design patterns for tooling for text analytics for Digital Humanities that are specially suited for users who don't know how to program and therefore enhance usability especially for this target group. The first pattern evolves around availability of the system as a web API. Further we propose a clear and easy workflow to handle data formats because combining different NLP tools often requires significant and complicated work with data formats. Next we present a format to define workflows or NLP pipelines that are only possible because of the aforementioned principles. We suggest to enable clear workflows between programmers and non-programmers to make clear which tasks can be done with and without programming knowledge. Last we propose a system to allow for reuse of code.

Keywords: Usability; Design Patterns; Research Infrastructures

1 Introduction

There are many text analytics tools for the digital humanitities but their usability is still a big issue [Bu16; Gn17]. One can use a Graphical User Interface (GUI). These are usually easy to use and therefore well suited for users who cannot code. Mostly GUIs are desigend for a specific research question. Adapting a Graphical User Interface to a new research question can require software development work. On the other hand one can use a programming language, e.g. Python and the popular NLTK framework for text analytics. Learning how to program has a steep learning curve but once one knows how to program this is the most flexible approach. In the domain of Digital Humanitites, programming languages and command line tools also are often too complicated [BW15].

In the Digital Humanities users of such text analytics software often cannot program themselves so implementing a text analytics workflow in a programming language is difficult for them. On the other hand research projects often explore new territory and a suitable GUI has not been created yet for the desired analytic. Therefore in this work we propose characteristics for a new type of tooling that is easier to use and learn then a programming language and more flexible then a GUI. On the same hand it will be less flexible then a programming language and more difficult to use then a GUI.

[1] German Research Centre for Artificial Intelligence, Speech and Language Technology, Alt-Moabit 91c, 10559 Berlin, Germany,jan.nehring@dfki.de

2 Background

There are different research infrastructures for the Digital Humanitites, for example CLARIN [HK14] or DARIAH[BS16]. Often these infrastructures offer graphical user interfaces, for example Weblicht [HHZ10] in case of Clarin. The amount of available tools for text analytics is large also, with major players being the Natural Language Processing Toolkit [LB02], Gate [CMB11] or UIMA [FL04].

Other authors investigate design patterns for research infrastructures for digital humanities. [Gn17] asked digital humanists for factors and criterias of the success of research infrastructures. The result was the following list which is translated from German: Data exchange, ease of use, flexibility, possibilities of integration, compatibility, scalability, stability, dissemination, availability. [Th17] adds the criterias adhere to standards, integrate stackholders in development process and documentation. Another user survey revealed that DH tools should make it possible to approach work in new ways [GO12]. Further they find out that the tools need to be be easy to use so they can be adopted by a broad audience [GO12]. [NS18] proposes to design text analytics tools with different user levels in mind. Users of different levels have different technical expertise. When this idea is considered in the design from the beginning then it allows for clear workflows: Users without programming skills know it advance which tasks they can perfrom and which not. Further it allows for a division of work between programmers and non programmers.

The FREME framework [Sa15a; Sa15b] is a text processing framework intending to bring together language tools and data in a unified approach. It was not specifically designed for the Digital Humanities but for industry use cases. Nevertheless we believe that the FREME framework is well suited for digital humanists also. The concepts mentioned in this paper are incorporated in the FREME framework. [SDN17] gives a detailed overview over the architecture of FREME. FREME is available as open source[2] and the API is publicly available and free to use at the time this paper is written[3]. The documentation homepage[4] shows more information about FREME.

In FREME NLP services and other services are called Enrichment Services (e-Services). Every e-Service performs one NLP tasks like Named Entity Recognition, Sentiment Analysis, and others. In the course of the FREME project, the following e-Services were developed:

- e-Entity for named entity recognition, classification and linking to the LD cloud
- e-Link to fetch additional information from the Linked Data cloud
- e-Translation for statistical machine translation
- e-Terminology for terminology annotation

[2] https://www.github.com/freme-project
[3] https://freme-project.github.io/api-doc/simple.html
[4] https://freme-project.github.io

Other projects like Digitale Kuratierungstechnologien or Qurator further use the FREME framework and add additional e-Services.

The Natural Language Interchange Format (NIF) [He13] defines a set of rules for the creation of NIF compatible language tools. We believe that NIF is a suitable data format for language tools for Digital Humanists. NIF is an RDF/OWL based format that defines a common vocabulary to describe NLP annotations. NIF addresses the interoperability of NLP tools on three layers: It is based on Linked Data (structural layer) and a selection of ontologies to describe common NLP terms and concepts (conceptual layer). NIF aware applications are accessible via REST services (access layer).

3 Proposals for design

We want to propose a system that is easier to learn than a programming language. This section explains important design patterns of this system. The following section will discuss these concepts.

3.1 Availability as a web of APIs

The proposed system should be reachable as an API via the internet as a Service Oriented Architecture [BCC08] which means that it consists of distributed and independent services. This is not a new approach but it is an important building block for the subsequent design patterns. This is already a widespread approach in digital research infrastructures, e.g. in CLARIN-D or the DARIAH project.

3.2 Self-contained services

The data flow in NLP pipelines is often complicated. Converting the output of one tool to the input format of the next tool in the pipeline often requires significant effort. Further this data conversion often requires programming work.

We want to overcome this complexity by using self-contained high level pipeline steps which consist of a low level pipeline themselves. For example a service performs Named Entity Recognition (NER). The input is plain text and the output is plain text annotated with Named Entities. Named Entity Recognition itself requires a low level pipeline consisting of sentence detection, tokenization and other NLP tools. The complexity of this low level pipeline is hidden in the self-contained high level pipeline step. It is called self contained because all the processing steps for NER are contained in the step and the NER system does not need information generated from preceeding pipeline steps. The NER service needs only plain text as input.

Every NLP tool has a data format that is used for data exchange inside of the tool. We suggest using a data format that is able to express all necessary annotations. Further it should adhere to standards and be able to embed linked data. The Natural Language Processing Interchange Format (NIF) [He13] points in a good direction for a suitable system data format. It uses the Resource Description Format (RDF) as data format. Therefore Linked Data can be integrated naturally in the data format. So when a NLP tools accepts plain text as input, it actually accepts NIF and uses the plain text embedded in NIF as input for the text analytic. There are also similar formats like for example the Web Annotation Data Model [5] but here we suggest NIF.

Using NIF as data exchange format it is possible to connect different analysis steps to each other very easy. But for users without deep technical expertise reading a Linked Data format is challenging. Processing it with SPARQL queries is almost impossible for them. Therefore we suggest that the input of the pipeline accepts also other formats like text documents, HTML or PDF. The system then converts these documents to NIF at the beginning of the pipeline. At the end of the pipeline is must be possible to convert NIF to another format which is better suitable for further processing, e.g. CSV. In this way it is possible to use the tool without getting in touch with the internal data format which increases usability.

3.3 Configurable workflows without programming

We believe that workflows need to be configured in a flexible way. Therefore we propose a workflow module that utilizes the aforementioned concepts of Service Oriented Architecture and easy handling of data formats.

Listing 1 shows an example how the proposed pipelining module can be configured. The example performs named entity recognition on the input text. Then it downloads all museums close to the detected locations from the Linked Open Data cloud and embeds the information in the NIF document. In the last pipeline step it converts the data from NIF to CSV.

The data format to define the pipeline is JSON. The plaintext is stored in the pipeline itself. One can also think of a model where the pipeline can be stored on the server first and then the input document is send to the pipeline.

Please note that the second step of the workflow addresses a service which is hosted at another API. So using this method one can span a web of APIs which can all be compatible to each other.

[5] https://www.w3.org/TR/annotation-model

```
{
  "document": "Welcome to Berlin, the capital of Germany!",
  "pipeline": [
  {
    "endpoint": "https://smart-service-1.de/named-entity-recognition/",
    "parameters": {
      "language": "en"
    }
  },
  {
    "endpoint": "https://smart-service-3.com/named-entity-linking/"
    "parameters": {
      "template": "fetch-museums"
    }
  },
  {
    "endpoint": "https://smart-service-1.de/data-format-conversion",
    "parameters": {
      "outformat": "csv",
      "conversion": "extract-labels-and-links"
  } ]
}
```

List. 1: Example for the workflow module

3.4 Enable clear workflows between programmers and non programmers

No matter how sophisticated a tool is tailored towards the needs of non programmers, non programmers will always encounter tasks that they are not able to solve. For example when the tool has a functionality to upload datasets, a non programmer might be able to upload a dataset to the API when the dataset already exists in the format accepted by the tool. But converting a dataset to this format can require programming knowledge and is therefore not suited for non programmers.

When designing tools for non programmers we believe it should be clear which functionalities of the tool require programming knowledge. The documentation can contain for example a table with all functionalities and the required level of expertise. This division of work should be considered in the design process of the tooling already.

3.5 Allow for reuse and sharing of code

When certain tasks can only be done by programmers then it is important to reuse the programmers work so once a programmer solved the task, other non programmers can reuse this work. The following paragraphs show an example:

Using the workflow description mentioned in section 3.3 in JSON it is possible to use Natural Language Processing without programming skills. The output of this system will be in the NIF format which is not necessarily ideal for users without a technical background. Also NIF is usually not accepted by other programs for subsequent tasks. So it should be possible to convert NIF to a different data format.

Using a SPARQL SELECT query it is possible to convert NIF to CSV. But we assume that it is too difficult for a non programmer to formulate a SPARQL SELECT query. Therefore we propose the SPARQL filter service that can store a SPARQL SELECT query together with a description on the server. The SPARQL filter service can be the last step of a pipeline. Once a programmer created the SPARQL query and stored it in the SPARQL filter everybody can use it.

Similar concepts can be used in various parts of the framework to store SPARQL CONSTRUCT queries to fetch external data from the cloud, XSLT stylesheets for XML transformations or pipeline definitions as shown in list 1.

4 Discussion

Tooling that follows above mentioned design patterns can enhance the accessiblity for people who cannot program in several ways.

Because of the availability of the tooling as a web API it is possible to use the system on every client computer, regardless of the operating system. Also the client computer does not need to be very strong, although the algorithms used might require a very strong computer. This lowers the barrier of starting to use the tool.

Further the installation of tooling, for example a python library and its required modules, is often complicated. Using above mentioned design, the user needs to install an HTTP client only. HTTP clients are widely available for free. Further and unlike text analytics tooling, they are mature software products which are easy to install. HTTP clients exist for every operating system and do not require strong computers.

The NLP services are self contained in a way that the underlying multistep pipeline is hidden in a single high level pipeline step. This allows for to view a system as a blackbox. Users need to know only that they can feed text into the system and which information they get out of it. Most internal parameters should be put to reasonable default values so the user does not need to learn the details of the system. Of course it is still possible to parametrize the NLP service using the web API but the extra work of implementing and documenting parameters automatically leads to slim nlp services with few parameters. When the goal is to reach for the highest possible performance scores, e.g. a high f-measure, then this is not a good approach. But we argue that it is more important to enable users to use the tooling in the first place then to optimize the algorithms for the last percents of performance.

The self contained services do not propagate all of their analysis to subsequent steps. Therefore it is likely that different pipeline steps perform the same preprocessing. From a computational point of view this is not efficient. This drawback shows especially in computational expensive preprocessing like Part of Speech Tagging. On the other hand this enhances the usability of the tooling in various ways. First it is very easy to take away parts of the processing pipeline. The other steps still work because no important preprocessing is destroyed by removing parts. Also adding a new analysis step is very easy. Further the tedious work of converting data formats is taken away from the user. We believe that for users without programming knowledge it is often not possible to convert a data format into another. So we believe that for users from the Digital Humanities without programming knowledge it is legit to reduce the computational efficiency for above benefits.

We decided to use NIF as the primary data format of the framework. NIF has a steep learning curve and understanding the concept of how to represent data as a graph is challenging. Furthermore processing NIF with SPARQL queries is hard also. But we argue that any data exchange format has a high complexity. Using the Turtle serialization format NIF is human readable. In addition to NIF it is crucial that users can feed data into the system in other common formats like plain text, HTML or PDF also and that the output can be converted to other formats, for example CSV.

The aforementioned concept of reusing code is important for various reasons. It allows for sharing of code to foster collaboration between researchers. It supports non programmers because over time the code databases on the API grow and non programmers might find a piece they want to use in the database. Also it is much easier to create a new piece of code by using a similar code as a template instead of starting from scratch. This sharing also allows for reproducability of research code.

5 Conclusion and Future Work

In this paper we presented design patterns for text analytics frameworks. We believe that these design patterns improve the usability of text analytics for people who cannot program. Nevertheless conducting text analysis is not easy. It is still an effort to learn using a framework with these concepts. But still it should be easier then learning how to program. Further we believe that it is not a good idea to reduce the complexity of the tooling to a point that hinders the adaption of the tool to new research questions. Our hypothesis is that a framework that follows the design principles mentioned here is a good compromise between flexibility and usability for our target group. The next step of our work is to conduct experiments with users to validate this hypothesis.

6 Acknowledgments

This research was supported by the Federal Ministry of Transport and Digital Infrastructure (BMVI) through the project SIM3S(19F2058B).

Literatur

[BCC08] Barai, M.; Caselli, V.; Christudas, B. A.: Service Oriented Architecture with Java. Packt Publishing, 2008.

[BS16] Blümm, M.; Schmunk, S.: Digital Research Infrastructures: DARIAH. In: 3D Research Challenges in Cultural Heritage II. Lecture Notes in Computer Science, Springer, Cham, S. 62–73, 2016, ISBN: 978-3-319-47646-9 978-3-319-47647-6, URL: https://link.springer.com/chapter/10.1007/978-3-319-47647-6_4.

[Bu16] Bulatovic, N.; Gnadt, T.; Romanello, M.; Stiller, J.; Thoden, K.: Usability in Digital Humanities - Evaluating User Interfaces, Infrastructural Components and the Use of Mobile Devices During Research Process. In (Fuhr, N.; Kovács, L.; Risse, T.; Nejdl, W., Hrsg.): Research and Advanced Technology for Digital Libraries. Springer International Publishing, Cham, S. 335–346, 2016, ISBN: 978-3-319-43997-6.

[BW15] Burghardt, M.; Wolff, C.: Humanist-Computer Interaction: Herausforderungen für die Digital Humanities aus Perspektive der Medieninformatik. In. 2015.

[CMB11] Cunningham, H.; Maynard, D.; Bontcheva, K.: Text Processing with GATE. Gateway Press CA, 2011.

[FL04] Ferrucci, D.; Lally, A.: UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. Natural Language Engineering 10/3-4, S. 327–348, Sep. 2004.

[Gn17] Gnadt, T.; E. Schmitt, V.; Stiller, J.; Thoden, K.: Faktoren und Kriterien für den Impact von DH-Tools und Infrastrukturen, 2017.

[GO12] Gibbs, F.; Owens, T.: Building Better Digital Humanities Tools: Toward broader audiences and user-centered designs. Digital Humanities Quarterly 6/, 2012, URL: http://www.digitalhumanities.org/dhq/vol/6/2/000136/000136.html.

[He13] Hellmann, S.; Lehmann, J.; Auer, S.; Brümmer, M.: Integrating NLP using Linked Data. In: 12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia. 2013, URL: http://svn.aksw.org/papers/2013/ISWC_NIF/public.pdf.

[HHZ10] Hinrichs, E. W.; Hinrichs, M.; Zastrow, T.: WebLicht: Web-Based LRT Services for German. In: Proceedings of the ACL 2010 System Demonstrations. S. 25–29, 2010, URL: http://www.aclweb.org/anthology/P10-4005.

[HK14] Hinrichs, E.; Krauwer, S.: The CLARIN Research Infrastructure: Resources and Tools for e-Humanities Scholars. en, Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)/, S. 1525–1531, Mai 2014, URL: http://dspace.library.uu.nl/handle/1874/307981, Stand: 12. 08. 2015.

[LB02] Loper, E.; Bird, S.: NLTK: The Natural Language Toolkit. In: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1. ETMTNLP '02, Association for Computational Linguistics, Philadelphia, Pennsylvania, S. 63–70, 2002, URL: https://doi.org/10.3115/1118108.1118117.

[NS18] Nehring, J.; Sasaki, F.: A Framework for the Needs of Different Types of Users in Multilingual Semantic Enrichment. In (chair), N. C. (; Choukri, K.; Cieri, C.; Declerck, T.; Goggi, S.; Hasida, K.; Isahara, H.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; Piperidis, S.; Tokunaga, T., Hrsg.): Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan, Mai 2018, ISBN: 979-10-95546-00-9.

[Sa15a] Sasaki, F.; Gornostay, T.; Dojchinovski, M.; Osella, M.; Mannens, E.; Stoitsis, G.; Ritchie, P.; Koidl, K.: Introducing FREME: Deploying Linguistic Linked Data. In: The 4th Workshop on the Multilingual Semantic Web: co-located with ESWC 2015, Portorož, Slovenia. 2015, URL: https://svn.aksw.org/papers/2015/MSW4/public.pdf%20http://ceur-ws.org/Vol-1532/paper6.pdf.

[Sa15b] Sasaki, F.; Gornostay, T.; Dojchinovski, M.; Osella, M.; Mannens, E.; Stoitsis, G.; Ritchie, P.; Koidl, K.: Introduction to FREME: Data meets Language meets Business. In: The 1st International Workshop on the Use of Multilingual Language Resources in Knowledge Representation Systems (MLKREP2015), Vienna, Austria. 2015, URL: https://svn.aksw.org/papers/2015/PN-ESWC/public.pdf.

[SDN17] Sasaki, F.; Dojchinovski, M.; Nehring, J.: Chainable and Extendable Knowledge Integration Web Services. In (van Erp, M.; Hellmann, S.; McCrae, J. P.; Chiarcos, C.; Choi, K.-S.; Gracia, J.; Hayashi, Y.; Koide, S.; Mendes, P.; Paulheim, H.; Takeda, H., Hrsg.): Knowledge Graphs and Language Technology. Kobe, Japan, 2017, ISBN: 978-3-319-68723-0, URL: https://drive.google.com/file/d/0BxeQvF3BluZUVWtzd2O2dGFsRUU/view.

[Th17] Thoden, K.; Stiller, J.; Bulatovic, N.; Meiners, H.-L.; Boukhelifa, N.: User-centered design practices in digital humanities - experiences from DARIAH and CENDARI. ABI Technik 37/1, np, 2017, URL: https://hal.archives-ouvertes.fr/hal-01606630%20https://pdfs.semanticscholar.org/9887/8215a5c92590800e01b4b02c5b9d6151de1c.pdf.