# impressoLOAD: Creating a Knowledge Graph Impresso Corpus

Présenté le 15 Juin 2020
à la Faculté d'Informatique et Communications
laboratoire Digital Humanities Laboratory
programme master en System de Communication
École Polytechnique Fédérale de Lausanne

pour le projet du semestre
par

Isabelle Pumford

supervisée par
Frederick Kaplan
Matteo Romanello
Maud Ehrmann
Lausanne, EPFL, 2020

# Contents

# List of Figures

# List of Tables

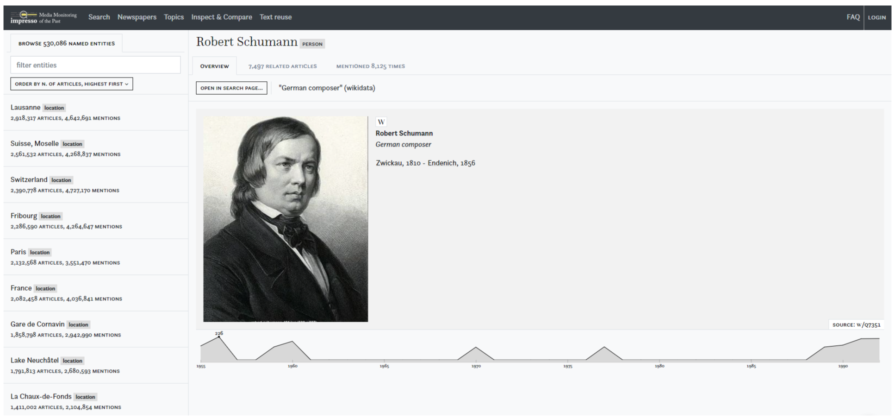
# 1 Introduction

Digitized newspapers constitute an extraordinary goldmine of information about our past, and historians are among those who can most benefit from it. Impresso[I], an ongoing, collaborative research project based at the DHLAB, has been building a large-scale corpus of digitized newspapers: it currently contains 76 newspapers from Switzerland and Luxembourg (written in French, German and Luxembourgish) for a total of 12 billion tokens. This corpus was enriched with several layers of semantic information such as topics, text reuse and named entities (persons, locations, dates and organizations). The latter are particularly useful for historians as co-occurrences of named entities often indicate (and help to identify) historical events. The impresso corpus currently contains some 164 million entity mentions, linked to 500 thousand entities from DBpedia (partly mapped to Wikidata).

Yet, making use of such a large knowledge graph in an interactive tool for historians — such as the tool impresso has been developing[II] — requires an underlying document model that facilitates the retrieval of entity relations, contexts, and related events from the documents effectively and efficiently. This is where LOAD comes into play, which is a graph-based document model, developed by Spitz and Gertz, 2016, that supports browsing, extracting and summarizing real world events in large collections of unstructured text based on named entities such as Locations, Organizations, Actors and Dates.

The goal of this project was to apply the LOAD model to the impresso corpus, adapting the LOAD implementation for the specific needs of the impresso corpus. This adaptation includes the following changes: the addition of different entity types, sources of different languages(German, French and Luxembourgish), changing the context weight relation from sentence-based to word distance-based and adapting the backend.

[I] https://impresso-project.ch/
[II] https://impresso-project.ch/app



Lausanne location
2,918,317 ARTICLES, 4,642,691 MENTIONS

Suisse, Moselle location
2,561,532 ARTICLES, 4,268,837 MENTIONS

Switzerland location
2,390,778 ARTICLES, 4,727,170 MENTIONS

Fribourg location
2,286,590 ARTICLES, 4,264,647 MENTIONS

Paris location
2,132,568 ARTICLES, 3,551,470 MENTIONS

France location
2,082,458 ARTICLES, 4,036,841 MENTIONS

Gare de Cornavin location
1,858,798 ARTICLES, 2,942,990 MENTIONS

Lake Neuchâtel location
1,791,813 ARTICLES, 2,680,593 MENTIONS

La Chaux-de-Fonds location
1,411,002 ARTICLES, 2,104,854 MENTIONS

Robert Schumann PERSON

"German composer" (wikidata)

**Robert Schumann**
*German composer*

Zwickau, 1810 - Endenich, 1856

Figure 1.1 – Example of impresso tool display for a searched entity, Robert Schumann

## 1.1 Motivation

One motivation for applying this type of graph to the impresso corpus includes the disambiguation of entities becomes less taxing when multiple entities of differing types are involved in a given context. By using a context-preserving representation of entity co-occurrences it becomes possible to utilize entity type information that would otherwise be lost and to answer similarly structured queries.

Another advantage of applying this structure to the impresso corpus is then that querying of the network would be possible through structured queries. This can be done with an interface through the command line of with a GUI created by Spitz et al., 2017 known as EVELIN[III]. This interface could be used by historians to explore the named entities and their contexts. Currently the impresso tool displays entity information, an example can be seen in Figure 1.1, a graph of related entities from a LOAD model could be displayed as well allowing historians to see related entities.

[III]https://evelin.ifi.uni-heidelberg.de/ interface for exploring LOAD model of English Wikipedia

# 2 LOAD Model

The LOAD model is an entity-centric network aimed at representing co-occurrences of named entities of arbitrary types in the *context of the surrounding terms.* A graph is formed based on an implicit network model. An implicit network model represents arbitrary types of entities with equal importance along with their co-occurring terms to retain context.

## 2.1 Node Types and Construction

The nodes in the LOAD network consist of 4 structural types: documents, sentences, named entities and terms. The documents are composed of sentences, these sentences are *bag-of-words* that contain named entities and terms. Named entities, or entities, are identified by an annotator and terms are all other significant words outside of entities. In the original LOAD implementation, four different types of entities were defined: Locations, Organisations, Actors and Dates. However, these are merely a noninclusive subset of the types of entities that can be defined.

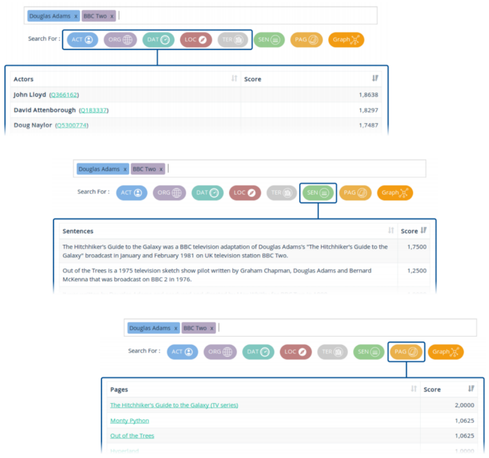
## 2.2 Edges and Edge Weights

The edge construction consists of three types, this is defined in the following definition.

**Definition 1.** (Set of edges $E$) *Given a set of nodes $V$ and some window size $c \in$ let $E \subseteq VxV$ denote the set of edges. For each $(v, w) \in E$, we require that one of the following conditions is met. (i) We have $v \in S$, $w \in D$ and $v \in d$. (ii) We have $v \in E \cup T$, $w \in S$, and $v \in w$. (iii) We have $v$, $w$, $\in E \cup T$ and there is at least one document for white $v$ and $w$ occur at most $c$ sentences apart.*

The edges between the nodes thus form an undirected graph. The weights of the edges of the graph is determined either as a binary relationship for documents to sentences and entities and terms to sentences or a weight of a distance function defined in the equations below.

$$\delta := |\varsigma(\iota_S(i)) - \varsigma(\iota_S(j))| \tag{2.1}$$



| Actors | Score |
|---|---|
| John Lloyd (Q366162) | 1,8638 |
| David Attenborough (Q183337) | 1,8297 |
| Doug Naylor (Q5300774) | 1,7487 |

| Sentences | Score |
|---|---|
| The Hitchhiker's Guide to the Galaxy was a BBC television adaptation of Douglas Adams's "The Hitchhiker's Guide to the Galaxy" broadcast in January and February 1981 on UK television station BBC Two. | 1,7500 |
| Out of the Trees is a 1975 television sketch show pilot written by Graham Chapman, Douglas Adams and Bernard McKenna that was broadcast on BBC 2 in 1976. | 1,2500 |

| Pages | Score |
|---|---|
| The Hitchhiker's Guide to the Galaxy (TV series) | 2,0000 |
| Monty Python | 1,0625 |
| Out of the Trees | 1,0625 |
| Hyperland | 1,0000 |

Figure 2.1 – Examples of using the GUI interface EVELIN

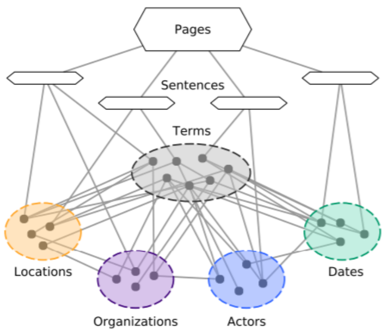


Figure 2.2 – LOAD Model

Where $\iota_S$ is an instance an entity or term within a sentence. If the two entities or terms exist in the same sentence then the distance is zero, otherwise it is progressively larger depending on sentence distance or assigned $\infty$ if outside a set sentence distance $c$.

**Definition 2.** (Undirected edge importance $\omega$) *Let $v, w \epsilon E \cup T$, then the importance weight of an edge $e = (v, w)$ between them is defined as*

$$\omega(v, w) := \sum_{i \epsilon I_v w \epsilon I_w} \exp -\delta(i, j) \tag{2.2}$$

The weights are then a sum of the negative exponential of the distance function for all instances of each node. Note that if comparing named entities, the type of named entity does not affect the weight of the edge between them.

# 3 Impresso LOAD

The original LOAD model needed to be modified to adapt to the information available in the impresso corpus. The adaptations necessary are summarized in Table 3.1 below and then elaborated on.

## 3.1 Corpus Composition

The original LOAD model was created from the entirety of English Wikipedia from June 2, 2015 as input. This included 4.6M English Wikipedia articles that contained at least one entity and 53.5M sentences that contained at least one annotation. The impresso corpus, on the other hand, consists of 76 newspapers (50 Swiss and 26 Luxembourgish) with almost 48M content items, its equivalent to articles. Only content items of type articles, obituaries or unsegmented pages are kept. The languages in this corpus are German, French and Luxembourgish.

## 3.2 Backend

The backend for the original LOAD model was composed of two different Mongo databases. One contained the sentences of the corpus and the other contained the annotated named entities. The Wikipedia page identifiers were extracted from the Mongo sentence DB during the first run of the model and then subsequently stored in a text document and read from this for later runs.

With the impresso corpus the content item identifiers are first extracted from Solr. This allows for the content item ids to be filtered out by article, obituary or page type. Originally, it was thought that the annotations could also be extracted from Solr but unfortunately, because the structure of the index is driven by the needs of the impresso tool, the named entities were not mapped to locations within the content item ids here. This meant that to know the offset of the annotations within the content item this had to extracted from MySQL and then loaded into S3. S3 was also used to read from the lemmatized tokens of each content item, here they

Table 3.1 – Comparison of LOAD implementations

| Attribute | Original LOAD Model | impresso LOAD Model |
|---|---|---|
| Corpus Composition | Wikipedia articles in English | French, German and Luxembourgish newspapers from last two centuries. Transcribed using Optical Character Recognition (OCR). |
| Backend | Two MongoDBs, one of sentences and one of annotations. | Solr DB for extracting the content item ids, S3 for obtaining the named entities and tokenized documents. |
| Entity Types | Location, Actor, Organization and Date | Preliminary implementation: Location and Actor, near future: Person, Organization, Producer, Time and Location |
| Term Extraction | All named entities are masked and terms extracted from sentences then stemmed | Lemmatized tokens already created, PoS kept: noun, number, proper nouns, verbs and adjectives. |

are given in sentence form but this is ignored in the final implementation, due to not knowing how reliable the annotated sentences are. The tokens also have their part of speech labeled, the nouns, proper nouns, adjectives, numbers and verbs are kept. If the token is a named entity, it is disregarded because the named entities come from the MySQL.

## 3.3 Entity Types

LOAD is an acronym for the original named entity types used by Andreas: Location, Organization, Actor and Date. The initial implementation of the LOAD model on the impresso corpus only used location and person entity types because at the time that was all that was available from the annotator, however, in the near future the person, organization, producer, time and location named entity could be used.

## 3.4 Term Extraction

As mentioned before, the Wikipedia corpus utilized by Andreas consisted of only the raw sentences and the annotations. During his implementation to extract the terms from a sentence that contained entities, he used the sentences and after eliminating the stop words, masked over the named entities and extracted the stems of the remaining words. These are then called the terms of the sentence. With the impresso corpus it was not necessary to process

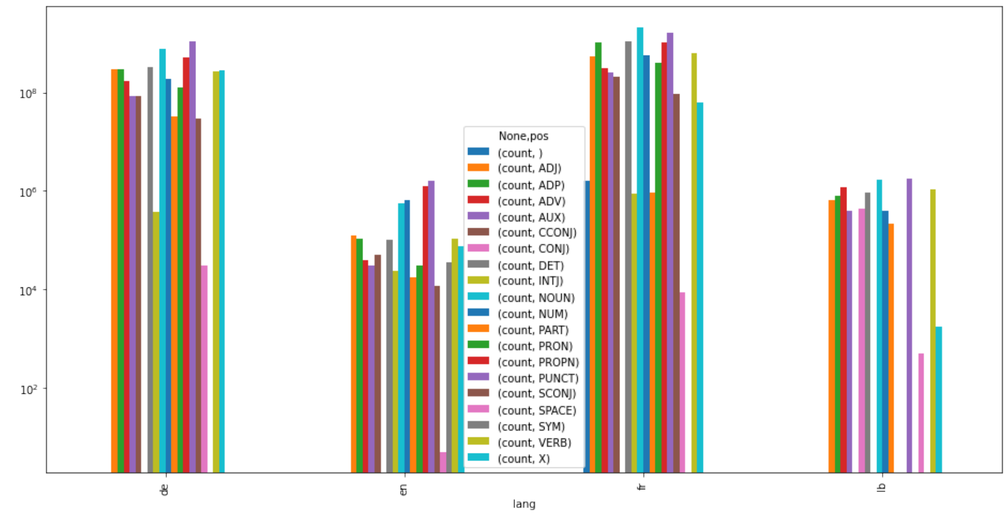


Figure 3.1 – Part of Speech tags distribution in each language of the corpus using a logarithmic scale. The PoS tags in the legend refer to spacy's PoS tagset https://spacy.io/api/annotation

the raw sentences because the lemmatized tokens already existed.

Using these lemmatized tokens provides another additional benefit, the corpus size is able to be reduced from a total of 14,6 billion to 6 billion tokens. This is done by using the part of speech tags and keeping only the nouns, proper nouns, numbers, verbs and adjectives. Note that even among the kept part of speech tags, only those within the context of named entities will be kept. This considerably reduces the number of nodes in the graph and the resulting querying time. An illustration of the distribution of tokens in each language in the corpus can be found in Figure 3.1.

# 4 impresso LOAD Pipeline

Both the original LOAD model and the impresso LOAD adaptation have been implemented in Java [I]. The image in Figure 4.1 gives an overview of the current impresso LOAD pipeline.

This overview can be broadly given in the following steps:

1. Before the construction of the network, the content item ids are extracted from the Solr DB. This is done by querying the DB using a cursor for all content items from a given newspaper, sorting in ascending order, only including the articles, obituaries and pages. The the cursor marks where the last query stopped so that the DB is queried until all content items are read. These content item ids are written to text files corresponding to their newspaper and year.

2. To begin the creation of the network, the main function in ParallelExtractNetworkImpresso instantiates an S3 Client and two caches. These caches are then passed to the MultiThreadHubImpresso and n-numbered MultiThreadWorkerImpresso. These two caches are composed of <String, JSONObject> key and content pairs. The string is the content item id and the JSON object are either the tokens or annotations for this content item. These are thread-safe caches shared between the workers and the hub.

3. The hub is then in charge of allocating the pageids (content item ids that have been mapped to a hashmap to integers) and the annotation ids to each worker. It also facilitates the populating of the newspaper and entity caches. Every time a new newspaper-year combination is reached in the hub tells the current worker requesting a new page id to populate the newspaper and entity caches. This is accomplished by the worker using the S3 Client to read the JSON file associated with the newspaper/year to the caches. The caches have a maximum number of entries and begin to "kick out" entries on a Least Recently Used (LRU) basis. This prevents too much memory from being used.

4. Each worker requests the content item id (in the form of the page id) from the hub. Then it uses the content id to create a ImpressoContentItem object. This object is composed

[I] https://github.com/dhlab-epfl-students/impresso-LOAD

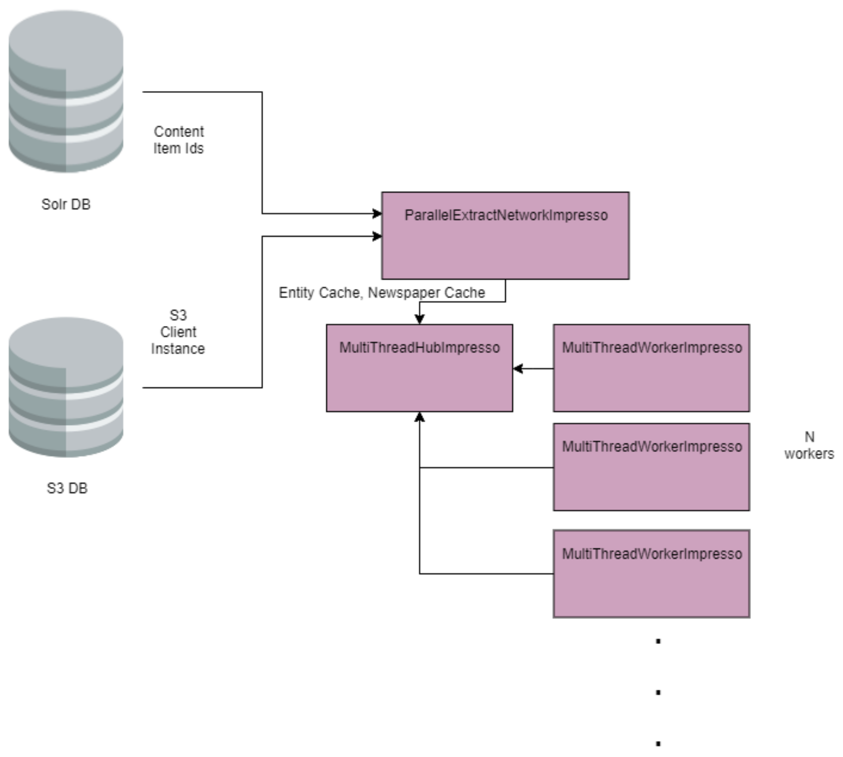

Solr DB
Content Item Ids
S3 DB
S3 Client Instance
ParallelExtractNetworkImpresso
Entity Cache, Newspaper Cache
MultiThreadHubImpresso
MultiThreadWorkerImpresso
MultiThreadWorkerImpresso
MultiThreadWorkerImpresso
N workers

Figure 4.1 – impresso LOAD Pipeline

of basic meta data about the content item, year, id and language as well as a list of Token objects. This list of Token objects also contains the Entity objects, a subclass of Token, read from the entity cache. The rest of the tokens are read from the newspaper cache.

5. This list of Token objects is then broken down into artificial sentences of token length 7. Each Token within this sublist is added to an Annotation object list. Each Annotation object is created with the annotation id, obtained from and coordinated by the hub, and the sentence id. Finally, the workers complete their work on the content item by writing the edges created within the content item.

# 5 Future Work

This final section will detail the necessary future work to fully complete the implementation of the LOAD model on the impresso corpus.

## 5.1 Introduction of Non-artificial Sentences

Currently artificial sentences are constructed by grouping together Token objects, of entity type or not, into groups of 7, which seemed a reasonable value. This was done because the sentences from OCR were segmented using spacy vanilla sentence segmentation and due to the noise from OCR it was decided to not trust the annotated sentences. To accomplish this grouping, the offset of tokens within an ImpressoContentItem object must have the offset read from the processed content items converted from sentence offset to content item offset. The named entity annotations are then read from the other S3 file and sorted withing the ImpressoContentItem object before the segmentation into 7 occurs.

In the future, the sentences will hopefully simply be found by a trusted annotator and the named entity location offsets can then be adapted so that they fit within a sentence. I think a good implementation of this would be to create a Sentence Class to hold the Token objects and have the ImpressoContentItem contain a list of sentences that can be sorted.

## 5.2 Hierarchy of Content Items

In the original LOAD implementation the documents from the Mongo DB were given a numerical id. This id is used to identify the page relative to other pages and used in the page comparator as an integer. I would in the future change this comparator to be fine with strings, so that the direct comparison between pages can be easily made. Currently a HashMap object of <Integer, String> maps these values to each other.

## 5.3 Query adaptation

The most immediate future goal would be to adapt the current GraphQueryInterface class from Andreas so that the command line could be used to query the created graph. This ideally would then also lead to the adaptation of EVELIN mentioned in Chapter 2 as a GUI interface that historians could used to query the LOAD graph.

## 5.4 Corpus Testing

Finally, a goal would be to expand the current newspapers that are being added to the graph. Testing thus far has been done on the NZZ, GDL, luxwort and indeplux newspapers. This should be expanded to include the other 72 newspapers in the impresso corpus as well.

# Bibliography

Spitz, A., Almasian, S., & Gertz, M. (2017). EVELIN: exploration of event and entity links in implicit networks (R. Barrett, R. Cummings, E. Agichtein, & E. Gabrilovich, Eds.). In R. Barrett, R. Cummings, E. Agichtein, & E. Gabrilovich (Eds.), *Proceedings of the 26th international conference on world wide web companion, perth, australia, april 3-7, 2017*, ACM. https://doi.org/10.1145/3041021.3054721

Spitz, A., & Gertz, M. (2016). Terms over LOAD: leveraging named entities for cross-document extraction and summarization of events (R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, & J. Zobel, Eds.). In R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, & J. Zobel (Eds.), *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval, SIGIR 2016, pisa, italy, july 17-21, 2016*, ACM. https://doi.org/10.1145/2911451.2911529