Connective-Lex.info

A Web App for a Multilingual Connective Database

Bachelor Thesis by Felix Dombek
(Matriculation No. 739122)

Computational Linguistics B.Sc.,
Department of Linguistics,
University of Potsdam

Supervisors:
Prof. Dr. Manfred Stede
Dr. Tatjana Scheffler

UNIVERSITÄT POTSDAM, 30.7.2017

Zusammenfassung

Die vorliegende Arbeit beschreibt die Konzeption und Umsetzung einer Webanwendung zum Durchsuchen und Vergleichen mehrerer verschiedensprachlicher, elektronischer Lexika von Diskurskonnektoren. Dies sind Wörter oder kurze Phrasen, welche Satzteile verbinden und in einen Sinnzusammenhang stellen. Dazu wurden in einer vorbereitenden Tätigkeit die unterschiedlichen Formate der fünf bisher verfügbaren Lexika durch XML-Transformationen aneinander angeglichen und Zuordnungstabellen für die verschiedenen genutzen Annotationsschemata erstellt. Darauf aufbauend wurde eine Single-Page-Webapplikation entwickelt, die bereits unter http://connective-lex.info abrufbar ist. Sie stellt Informationen über die Lexika bereit, macht sie durchsuchbar und bietet verschiedene Filtermöglichkeiten für die Datensätze. Die Implementierung ermöglicht es, jederzeit Lexikonupdates einzuspielen oder neue Lexika hinzuzufügen, die sofort in die Darstellung der App übernommen werden.

Contents

1 Introduction

This thesis presents the work done to integrate several independent machine-readable lexicons, or word databases, of the discourse connectives of different languages into Connective-Lex.info: a publicly available web application which allows anyone to browse the lexicons, search them, and display information on the words contained in them.

The idea for this web application originated in the European TextLink Action[1] as DiMLex, the Diskurs-Marker-Lexikon for German connectives (Stede 2002), was complemented by multiple similar lexicons for other European languages. One of the key requirements for the web app was that it must automatically work with any new lexicons in DiMLex format which might be added later. The app therefore accepts any new or updated lexicons as input and immediately incorporates them into its presentation. It displays syntactic and semantic (sense) information for each entry. Several lexicons can be browsed at once, thus correlations between connectives of different languages can be displayed, either by viewing synonymity annotations or by using specific filter settings to view a subset of all multi-language entries. Language-independent info from the lexicons has been incorporated into the web app to a large extent, and even some language-specific features of the source lexicons can be displayed.

1.1 Discourse Connectives

Discourse connectives, also called discourse markers, are words or short utterances which set two phrases into a relation, such as “because”, “while”, and “even though”. They are explicit signals for relationships between phrases and are thus essential for building coherent rhetorical discourse out of individual propositions. Connectives belong to different word classes, but are most frequently conjunctions (coordinating or subordinating), followed by discourse-marking adverbs/adverbials. For the purposes of exhaustive coverage of discourse-marking words, some prepositions must be considered as well, although there is no universal consensus on whether to include them in this category as their arguments are not clauses (cf. Stede 2002, p. 6).

Each connective can mark one or more discourse relation senses, and there exist several linguistic frameworks which aim to comprehensively classify these senses. Among the first such frameworks is the widely used Rhetorical Structure Theory (RST) (Mann & Thompson 1988); newer frameworks are Asher and Lascarides’ (2003) Segmented Discourse Representation Theory (SDRT) and the Penn Discourse Treebank (Prasad et al. 2008) with its associated tagset, whose current version, PDTB3 (Webber et al. 2016), is the one used for DiMLex and hence also for this web app. The exact rhetorical senses that are postulated and that can be represented by a conforming annotation vary a lot from framework to framework, but they generally include, for instance, narrative senses (such as “narration”, “explanation”), comparative senses (“contrast”, “concession”) as well as causal, conditional, and temporal senses.

1 http://textlink.ii.metu.edu.tr

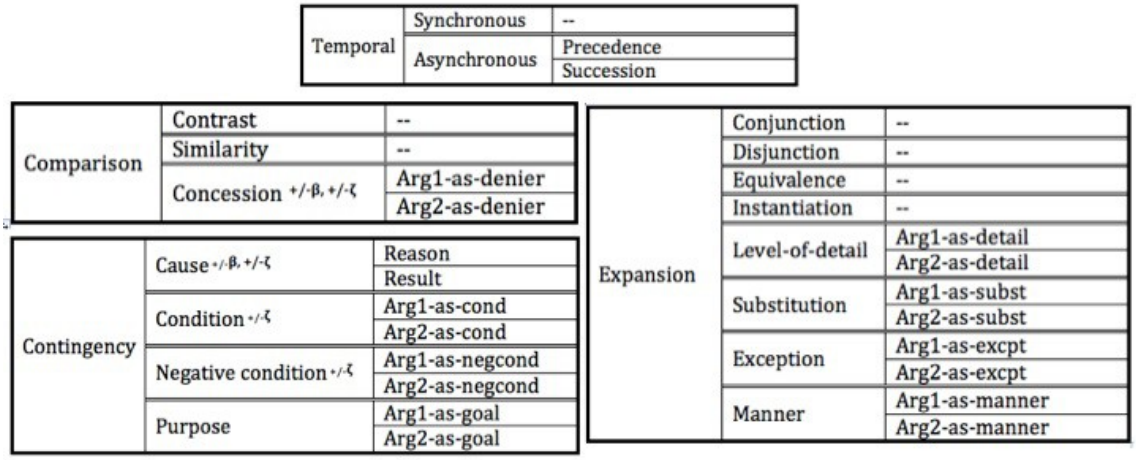

Figure 1: The PDTB3 sense hierarchy.

It should be noted that many rhetorical relations in texts are not stated explicitly through discourse connectives, but are instead established implicitly from context. The web app which is the topic of this thesis, however, is not concerned with such implicit relations as the lexicons that were used only contain information about connectives as lexical items.[2]

Speech and language technology needs machine-readable resources that facilitate automatic processing of language, and the lexicons provide such resources. In computational linguistics, the formal analysis of discourse connectives has applications in several disciplines such as text generation and rhetorical structure parsing, which both rely on solid models of discourse relations. Building on the text understanding which results from "shallow", i.e. non-syntactic discourse parsing, research has also demonstrated the use of rhetorical structure in machine translation (e.g. Danlos & Roze 2011), sentiment analysis (e.g. Bhatia et al. 2015), and argumentation mining (e.g. Peldszus & Stede 2013). DiMLex has in fact already been used in practical applications of text generation and text understanding (Stede 2002).

1.2 The Penn Discourse Treebank 3 Tagset

The PDTB3 sense tagset is structured hierarchically: At the root there are four *classes*, often written in uppercase (COMPARISON, CONTINGENCY, EXPANSION, TEMPORAL), each of which contains several *types* which are the actual sense names. If the sense is directional, or asymmetric, then the type has exactly two *subtypes* which specify the directionality of the relation. The full relation hierarchy is shown in Figure 1 (taken from Webber et al. 2016). The two argument clauses of the connective are called Arg1 and Arg2, where Arg2 is always the phrase which syntactically contains the connective, regardless of their actual order in the text. Some of the relations can be additionally annotated with features as conveying a belief (+Belief, +β) or

2 A slight exception from this rule, as will be shown in the paragraphs on the Portuguese lexicon LDM-PT, lies in some of its connectives being marked *AltLex*: In PDTB3 parlance, these are non-connectives, usually phrases, which serve a connective function in a certain context (Prasad et al. 2007).

containing a speech act (+SpeechAct, +ζ), which "may or may not be associated with one of the defined arguments of the relation. Features […] should not be seen as a property of the relation, rather of the arguments" (ibid., p. 27). Nevertheless, these features are annotated for some of the connectives in the lexicons, and therefore also appear in the filter settings of the web app. An example for a CONTINGENCY:Cause+β:Result is "The blinds are down, therefore she isn't at home" (which is not a logical inference, but a belief resulting from world knowledge). There is only one example for a speech act annotation in the lexicons, here translated from Portuguese: "I agree with the return to the official time which was in effect before 1992. By the way, in my opinion the time should never have changed.", an example for "aliás" ("by the way") having the reading COMPARISON:Concession+ζ:Arg2-as-denier+ζ in LDM-PT (Mendes & Lejeune 2016).

2 The Lexicons

Developed initially at TU Berlin (Stede & Umbach 1998) and then at Universität Potsdam (Stede 2002), DiMLex had for a long time been the only of its kind, until the French LexConn (Roze et al. 2010), the Italian Lexicon of Connectives LICO (Feltracco et al. 2016), and the Portuguese Lexicon of Discouse Markers LDM-PT were created to serve a similar function for their languages. A fifth lexicon, PDTB-DiMLex, stands out a bit because it was entirely machine-generated from PDTB corpus sources (Prasad et al. 2008) by Dr. Tatjana Scheffler; it contains, in DiMLex format, the list of English connectives with the senses that are extractable from the annotations in that corpus.

All of these lexicons have several things in common. They all aim to provide a complete, machine-readable database of the connectives of their respective language. They all use XML as their file format to encode lexical, syntactic, and semantic information about the connectives. However, there are also significant differences. The actual XML formats are completely different in each of them except for the two DiMLexes; no single XML parser and validator for a specific Document Type Definition (DTD) could read them. They also use different tagset frameworks for both word classes and semantic senses. In order to facilitate file parsing for the actual app, it is crucial that the formats are normalized to a single, canonical format and that the word class and sense annotions are mapped to a fixed tagsets.

We naturally chose our own DiMLex format as the target, because it is already in active use and is easily extendable by us so that particular annotation elements from the other lexicons could be retained in the normalized version as far as desired. We also initally thought that LICO already existed in this format, which turned out not to be the case. For DiMLex, the normalization part therefore consisted only of bugfixing. For LexConn, LICO and LDM-PT, we employed XSL transformations to convert the lexicons into the DiMLex format, and created

mappings from their respective part-of-speech (POS) and sense tagsets to the ones used for the web app. For PDTB-DiMLex, only the tagset had to be converted.

2.1 DiMLex

The DiMLex lexicon provided the canonical format for the web app to use as its input. There is a freely available version of it on GitHub[3] which is automatically generated from a larger, non-public version which contains information from proprietary sources. DiMLex contains 274 German connectives, and the structure of its public edition is shown in Listing 1.
Elements which are processed by the app are set in bold. The three positions marked with asterisks (*) are the positions where certain new elements could be inserted that are described below. The places marked with ellipses (…) indicate that the preceding tag could occur multiple times in that position.

The document is headed by a DTD doctype declaration which names the root element, `dimlex`, and the path to a DTD which can be used to validate the document. Inside the root element, there is a list of `entry` elements, one for each connective in the lexicon. The meaning of the different tags and attributes of each entry will be explained with some level of detail, as this is the central structure that was used for all lexicon conversions and in the web app.

`entry`

A single lexicon entry, i.e. a lexical item. Its lexicon-wide unique ID and the base word are attributes of this tag.

`orths`

Contains the list of orthographical variants. For LICO and LexConn, some closely related lexical variants are listed here as well instead of having their own entry.

`orth`

Contains a single orthographical or lexical variant of the connective. The `type` attribute is either *cont* or *discont*, indicating whether the connective consists of two disconnected parts. The `canonical` attribute has a value of *1* for the most wide-spread form and *0* for the others.

`part`

A part of the connective. A *cont* `orth` has only one `part`; a *discont* one has two. A `part` is of `type` *phrasal* if it consists of more than one word, otherwise it is of `type` *single*.

`ambiguity`

Contains information about whether the connective has non-connective uses, and whether its connective uses have readings with different senses. The ambiguity can also be inferred

3 https://github.com/discourse-lab/dimlex

```
<!DOCTYPE dimlex
  SYSTEM "dimlex.dtd">
<dimlex>
   <entry id="k1" word="aber">
      <orths>
         <orth type="cont" canonical="1" onr="k1o1">
            <part type="single">aber</part>
            ...
         </orth>
         <orth type="cont" canonical="0" onr="k1o2">
            <part type="single">Aber</part>
         </orth>
         ...
      </orths>
      <ambiguity>
         <non_conn freq="3" anno_N="21">1</non_conn>
         <sem_ambiguity>1</sem_ambiguity>
      </ambiguity>
      <focuspart>0</focuspart>
      <non_conn_reading>
         <example type="ADV" tfreq="940">aber und abermals</example>
         ...
      </non_conn_reading>
      <stts>
         <example type="KON" tfreq="349">Mein Auto faehrt nicht
        besonders schnell, aber das Auto von Juergen.</example>
      </stts>
      (*)
      <syn>
         (*)
         <cat>konnadv</cat>
         <integr/>
         <ordering>
            <ante>0</ante>
            <post>1</post>
            <insert>0</insert>
         </ordering>
         <sem>
            (*)
            <pdtb3_relation sense="concession-arg2-as-denier" freq="7" anno_N="18"/>
            ...
         </sem>
         ...
      </syn>
      ...
   </entry>
   ...
</dimlex>
```

Listing 1: The DiMLex XML structure.

from other properties of the entry. The non-connective readings are deliberately kept out of the scope of the web app, which is only concerned with connective readings.

`focuspart`

This field has value *1* if the connective can be used as a focus particle or focus marker, i.e. if it can set its argument clause into the focus of the sentence, and a value of *0* otherwise.

non_conn_reading

Contains examples of non-connective readings. Because only connective readings of words are taken into account, this field is ignored for the purposes of this thesis.

stts

In the DiMLex lexicon, this field contains a list of examples for connective readings of a word, some of which are extracted from the TIGER corpus (Brants et al. 2004). The acronym STTS stands for Stuttgart-Tübingen TagSet, which is the set of syntactic categories with which these examples are annotated. The examples are taken over into the web app, but because the STTS tagset is different from the other syntactic annotation of the entries, they are only incorporated at word level instead of at syntactic or sense level.

syn

The entry contains one `syn` (syntactic info) element for each word class that the connective occurs as. All important syntactic and sense information is located under these nodes.

cat

Specifies the syntactic category using a custom tagset. This is an abridged overview taken from the DiMLex documentation on GitHub:[4]

- *konnadv* adverb ("anschließend")
- *padv* adverb with prepositional part ("außerdem")
- *konj* coordinating conjunction ("und")
- *subj* subordinating conjunction ("weil")
- *v2emb* V2 embedder ("vorausgesetzt")
- *postp* postponer ("weshalb")
- *appr* preposition ("anstatt")
- *appo* postposition ("wegen")
- *apci* circumposition ("um … willen")
- *einzel* isolated ("dass")

These categories are largely meant to describe specifically German syntactic features; we use a reduced, more general tagset in the actual app and map these categories onto the simpler ones.

integr

This field is always empty in the public version of DiMLex.

ordering

For word classes which are not prepositions, this field contains the sub-fields `ante`, `post`, `insert`, and, rarely, `desintegr`. They have a value of *1* if the word, together with its argu-

4 https://github.com/discourse-lab/dimlex/blob/master/DimLex-documentation.md

ment clause, can occur at the indicated position in a sentence, or a value of *0* otherwise. For prepositions, the `ordering` tag is replaced by a `praep` tag, which has similar child nodes `ante`, `post`, `circum`, and one or more `case` nodes, where the latter indicate the cases which the preposition can assign to its noun phrase (*gen*, *dat*, or *acc*).

`sem`

The actual semantic sense node. Each `syn` can have one or more `sem` nodes as children. Inside, there is a `pdtb3_relation` with a `sense` attribute whose value is a rhetoric sense name from the PDTB3 framework. In DiMLex, each `sem` has exactly one semantic sense, but other lexicons can have more than one relation listed here if these relations actually occur at the same time, as parts of a single reading. For multiple mutually exclusive senses, separate `sem` nodes are used instead.

As we looked into the formats of the other lexicons, it became clear that some new tags had to be allowed in the DiMLex format in order to not lose certain information. Some of the other lexicons contain tags for synonyms, references to DiMLex, and examples in different positions, which we do not want to lose during conversion. Therefore, in the places marked with asterisks (*), the following new element types can optionally be added at `entry` level, at `syn` level, or at `sem` level according to the specificity of the source annotation:

`synonyms`

Contains a list of `synonym` children which contain the synonym as text and the optional attributes `lexicon` and `entry-id` for references to other entries, possibly in another lexicon. This tag can also be used for translations which have no entry in another lexicon, by using a `language` attribute with a language name as its value.

`examples`

Contains a list of `example` children whose text is a sentence that uses the connective, possibly as a specific word class or in a specific sense according to the location of the `examples` tag.

Some more information on the DiMLex format, especially regarding some of the fields which are of no importance for this project and which are not described in all detail in this thesis, can be found in the DiMLex documentation.

A DTD was created for the DiMLex format to check this lexicon and the converted versions of the other lexicons for syntax errors. All the lexicons have to validate against this DTD in order to work with the web app.

2.1.1 Bugfixes and Tagset Mapping

With the DiMLex format designated to be the canonical input format for the app, no format conversion had to be performed for the DiMLex lexicon. However, there were numerous incon-

sistencies and some bugs. I used XSLT to create lists of all occurring field values in DiMLex and collected all those items that needed to be cleaned. Among these were IDs used twice, wrong attribute values, double tags, and similar issues. The resulting list of proposed changes then had to be applied to the non-public version of DiMLex, because it is the master version from which the public version is automatically generated.

Next followed the handling of the sense annotations. DiMLex's sense tagset is the same as the app sense tagset, PDTB3, but the notation used is a short notation which omits the class. Because it is desirable to have the class available in the app, a sense mapping was created from the short notation to the full notation. The POS tagset was also mapped from the custom tagset that is used in DiMLex to a simpler one taken from LexConn, which contains the important word classes without language-specific peculiarities. It only contains *cco* (coordinating conjunction), *csu* (subordinating conjunction), *adv* (adverb/adverbial), and *prep* (preposition). All POS tags which have no corresponding tag in this reduced set, such as *einzel*, are mapped to a catch-all *other* type.

2.2 PDTB-DiMLex

PDTB-DiMLex was automatically constructed from the Penn Discourse Treebank corpus and contains 101 English connectives. It is very clean, and no format conversion or bugfixing had to be done. However, the tagsets used in that corpus, and therefore in this lexicon, are different compared to DiMLex: Its POS tagset is the Penn Treebank POS tagset (PTB2), and the sense tagset is PDTB2, the previous version of the PDTB tagset. These tagsets were therefore mapped to the app tagsets using their respective annotation guide documents. PDTB2 is slightly more fine-grained than version 3, but has a very similar hierarchical structure and largely the same tags. The mapping could therefore be performed without introducing ambiguities and without any significant loss in specificity.

2.3 LexConn

LexConn, the *base lexicale des connecteurs discursifs du français* (lexical database of French discourse connectives) is the result of a master's thesis by Roze (2009) at the Université Paris Diderot and is described in English in Roze et al. (2010). It is the only used lexicon which includes an associated XSL transformation script so that it can be displayed in a browser.[5] It contains 328 connectives and is structured very differently from DiMLex. The largest difference is that different readings of a word are separated from each other, i.e. words that are ambiguous on the syntactic or semantic level have multiple entries. In DiMLex, multiple readings are simply annotated with multiple `syn` or `sem` tags as shown in Listing 1. Another big difference is that is uses the SDRT framework for its sense annotations.

5 http://www.linguist.univ-paris-diderot.fr/~croze/D/Lexconn.xml is displayed as a nicely formatted table instead of the XML source.

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="./tableau.xsl"?>
<liste>
	<connecteur cat="csu" id="c31" relations="parallel" type="coord" position-
sub="postposée">
		<forme>
			au même titre que
		</forme>
		<forme>
			au même titre qu'
		</forme>
		...
		<commentaire>
			Aucune occurrence de &quot;au même titre que p1, p2&quot; trouvée
dans FRANTEXT.
		</commentaire>
		<exemple>
			Le rétro comme antimode ou comme non-mode : cela ne désigne pas la
fin de la mode mais sa phase humoristique ou parodique, (au même titre que/de même
que) l'anti-art n'a jamais fait que reproduire et élargir la sphère artistique en y
intégrant la dimension humoristique.
		</exemple>
		...
		<synonyme connecteur="c145"/>
		...
	</connecteur>
	...
</liste>
```

Listing 2: The LexConn XML structure.

A typical entry is shown in Listing 2. An ellipsis (…) again denotes a possible repetition of the previous tag. The listing also shows the presence of a `synonyme` tag that refers to another LexConn entry, and two more linguistic annotations: The `type` attribute specifies the type of the rhetorical structure that the relation imposes on its argument clauses, which is different from the syntactic structure and can take the values *coord*, *sub*, and *unknown*, and the `position-sub` attribute only occurs for subordinating conjunctions and specifies the grammatical order of the argument clauses (*antéposée* or *postposée*; three are marked *postposée?*). A similar attribute, `position-adv`, is present in adverb entries and can be *initiale* or *interne*. The tags annotated in the LexConn entries were exactly the broad categories that were needed for the web app, and could be copied without change. Similar to DiMLex, each entry could have multiple orthographic variants, listed here as `forme` tags, of which the first could be regarded as the canonical variant.

2.3.1 Format Conversion

For all format conversions, XSL transformation scripts were employed that read in an XML document, transform its structure, and output the resulting transformed document. XSLT is itself specified in XML format, but is a fully-fledged declarative programming language, although it isn't very useful beyond its domain of XML transformations. The open source tool Saxon HE 9.7 (.NET)[6] was used to run the XSLT scripts.

6 http://saxon.sourceforge.net

To convert LexConn into the DiMLex format, an XSL transformation script was created which first groups connecteur entries by the first `forme` text. This makes sure that all connecteur entries for a particular word were correctly merged into a single result entry. Its `sem_ambiguity` value could then be established from the count of entries in that group, as only in ambiguous connectives the group would have a cardinality larger than one. The forms were then transcribed into `orth` tags, and uppercase variants were added. Phrasal forms were distinguished from single forms by the presence of at least one space. A special case was created for handling non-continuous forms, which unfortunately were not overtly marked as they are in DiMLex. Discontinuous forms in LexConn look like this: *d'un côté (...d'un autre côté)*. The forms therefore had to be parsed to be recognized and split up correctly, this example resulting in one part *d'un côté* and another *d'un autre côté*. For discontinuous forms, all four possible capitalization variants (a b, A b, a B, A B) were created, because in many cases, there could be a sentence boundary between the parts.

Next, the syntactic and semantic information was translated into the DiMLex structure. This was done by iterating through the group entries, first grouped by category, and then through the entries of the resulting groups, which are the individual relations. At this stage, the examples and synonyms were taken over into the target structure. In LexConn, those elements are necessarily attached to a specific reading, so they are also inserted into the DiMLex structure as children of `sem`. For synonyms, it was deemed very practical if the actual word would appear in the target tag, where LexConn lists synonyms only by ID. The reference given by the synonym ID was therefore resolved, and the word with this ID was copied into the tag. This is quite helpful for the web app, especially performance-wise, because it ensures locality of data (at the cost of adding some redundancy). Resolving such a reference in the app, while programmatically simple, would take considerably more time than simply looking at a single tag.

The `position-sub` and `position-adv` attributes are currently not translated into the DiMLex format. They could roughly correspond to the ordering tags in DiMLex, but more information is needed to ensure that they in fact encode exactly the same kind of information and can be freely converted. Other DiMLex fields, such as the `focuspart`, `non_conn*`, and `stts` fields, were intentionally left blank by the conversion, as they did not have corresponding data in LexConn.

As LexConn uses the SDRT set of discourse relations, they must be mapped to PDTB3. However, this is not a task for XSLT. For one, the mapping files should be easily editable, which XSLT files arguably are not. Also, it would be good to have all mappings in one place and no need to re-run the XSL transformation in case the mapping is updated. As the mapping therefore has to be done by the app, the tagset was not changed in the current step. Instead, I inserted an `sdrt_relation` tag at the position where DiMLex usually has its `pdtb3_relation`. The relation `type`, a value unique to LexConn, is also written into this tag as an attribute.

The sense mapping itself was initially supposed to be based on preliminary work done by Colinet (2015) during a short-term project at Uni Potsdam which was aimed at combining LexConn with DiMLex under the moniker DiMLexConn, but during review of this work, it was found that no mapping from either tagset to the other was available. The SDRT-to-PDTB3 sense mapping was therefore decided to not be a part of the present thesis.

In order for the app to work, there has to be some mapping from SDRT to PDTB3, and I tried to work out one that is useful, even if possibly incomplete. Obtaining a full mapping is complicated by the fact that the tagsets are very different and that no simple annotation guide for all SDRT relations was available during this work. It is clear at this stage that no one-to-one mapping can exist, therefore introducing the need to potentially map SDRT relations to lists of possible PDTB3 candidate senses. This has an unwanted side effect in that, for some words from LexConn, the app would display inappropriate PDTB3 senses alongside valid ones. This is an inherent issue of incompatible tagsets and could only be resolved by creating PDTB3 annotations for the LexConn entries on a word-by-word basis.

There are SDRT annotations in the non-public version of DiMLex, and an XSLT script was developed which collects the cooccurrences of tags and creates a mapping from that, of which an example entry and an explanation are shown in Listing 3. Specifically, the script recorded, for each SDRT sense, the PDTB3 senses that are used for the same words. The result is an XML file with a nested list:

- The list of SDRT senses is at the root.
    - For each SDRT sense: List of PDTB3 senses used for the same words, including an occurrence count.
        - For each PDTB3 sense: List of words for which both tags were used together.
            - For each word: Is the SDRT sense unique for this word or was this word annotated with multiple SDRT senses?
              —If unique: All PDTB3 senses that were also used for the word are valid mappings for the SDRT sense.
              —If not unique: The PDTB3 senses are candidates that have to be checked for validity.

While quite clear in theory, it turned out that this method has several issues. First, in DiMLex, SDRT is annotated per word and not per reading, rendering this method quite useless for any ambiguous connectives. Second, the existing SDRT annotations do not necessarily capture all senses that a word can have. Together, these circumstances result in a very high amount of false positives (matches that are not really good equivalents) and also some missing mappings. Even if these two problems were solved, some SDRT/PDTB3 annotation pairs which validly occur together in the DiMLex data might actually be a very bad fit for a general mapping. This is why occurrence counts were collected which helped sort out rarely occurring pairs.

```
<entry sdrt="alternation">
   <counts>
      <pdtb3 name="negative-condition-arg2-as-consequent" count="3">
         <src entry-id="k8" word="andernfalls" other="opposition continuation "/>
         <src entry-id="k8" word="andernfalls" other="opposition continuation "/>
         <src entry-id="k138" word="sonst" other="opposition continuation "/>
      </pdtb3>
      <pdtb3 name="disjunction" count="3">
         <src entry-id="k29" word="beziehungsweise" unique="true"/>
         <src entry-id="k231" word="entweder...oder" other="elaboration "/>
         <src entry-id="k117" word="oder" other="elaboration "/>
      </pdtb3>
      <pdtb3 name="specification-arg2-as-detail" count="1">
         <src entry-id="k29" word="beziehungsweise" unique="true"/>
      </pdtb3>
      <pdtb3 name="conjunction" count="1">
         <src entry-id="k29" word="beziehungsweise" unique="true"/>
      </pdtb3>
      <pdtb3 name="negative-condition-arg1-as-consequent" count="1">
         <src entry-id="k234" word="es sei denn" unique="true"/>
      </pdtb3>
   </counts>
</entry>
```

Listing 3: Mapping DiMLex (non-public) SDRT annotations to PDTB3 by example of "alternation". The entry shows that "beziehungsweise" and "es sei denn" are SDRT-annotated only with "alternation". Therefore, all of their PDTB3 annotations ("disjunction", "specification-arg2-as-detail", "conjunction", "negative-condition-arg1-as-consequent") are potential matches to which "alternation" could be mapped. In reality, using all such matches leads to a huge number of dubious mappings. Looking at the counts, however, it turns out that "alternation" most often occurs with "disjunction". Such a mapping was then used for the preliminary SDRT-PDTB3 mapping table.

The resulting mappings were then sanitized by looking at these counts and also into the meaning of the annotations, removing and adding annotations according to the guidelines at hand (Reese et al. 2007), but more work has to be done to obtain a viable, mostly correct mapping file. Therefore, the current sense mapping table from SDRT to PDTB3 is a preliminary version.

2.4 LICO

The Lexicon of Italian Connectives LICO was developed at Fondazione Bruno Kessler in Trento, Italy, by Feltracco et al. (2016) and contains 173 connectives. It was actually based on DiMLex and was initially thought to be available in the same format. However, instead of the base version, the authors took a specific transformation of DiMLex as the template for LICO. This DiMLex transformation is a reformatting specifically to target ConAno, a rhetorical structure annotation tool (Stede & Heintze 2004), and the format is known as ConAnoLex. Obtaining a ConAno-compatible version of LICO might also have been achieved by applying the existing XSL transformation by which DiMLex is converted to ConAnoLex. The root tag was later changed from `conanolex` to `lico`, so it is now definitely incompatible to ConAno.

LICO uses the PDTB3 tagset with fully specified sense names and a custom POS tagset with word class names in Italian. Its format is shown in Listing 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<lico>
  <entry id="1">
    <orth type="cont">
     <part type="phrasal">a causa di</part>
     ...
    </orth>
    ...
    <syn type="prepositional">
      <sem>
        <coh-relation>CONTINGENCY:Cause:reason</coh-relation>
        <example>A causa del maltempo il St. Gotthard è rimasto chiuso.</example>
        ...
      </sem>
      ...
    </syn>
    ...
    <commento>DimLex.xml/id="k19"</commento>
    ...
  </entry>
  ...
</lico>
```

Listing 4: The LICO XML structure.

2.4.1 Format Conversion

As we needed LICO in the canonical DiMLex format, the need for a retransformation script from the ConAno format back to DiMLex arose. An XSL transformation script was written which converts LICO to DiMLex. The actual structural transformation was very straightforward, especially since ambiguous connectives were already realized as multiple `syn` and `sem` elements within the same entry, and because the `orth` structure requires no special parsing as was the case with LexConn.

A special complication emerged from the synonym IDs used in LICO. The `commento` tags contain a lexicon name (this used to be "ConAnoConnectorLexicon.xml", pointing to the DiMLex version used as the source), an ID and, sometimes, a sense name. The IDs identified entries in the ConAnoLex version of DiMLex. However, during the transformation from DiMLex to ConAnoLex, all IDs are reassigned sequentially. This means that for each new version of DiMLex, whenever this transformation is applied, all ConAnoLex IDs change. As the ConAnoLex IDs were not fixed, it was vital to have IDs referencing the original DiMLex source, and the IDs had to be converted. This was done by creating two map files via XSLT: one containing all ConAnoLex entries with word and ID, and another one containing all DiMLex entries in the same format. An XSLT script was then created which used these two map files to process LICO, obtaining the word for the existing ConAnoLex ID from the first map file and mapping it to the corresponding DiMLex ID using the second map file. The updated IDs were then written back to LICO. The resulting updated LICO version was made available to the original authors and has since replaced the previous version.

This updated LICO file was the basis for the translation to DiMLex. Generating `orth` capitalization variants was done with the same code used for LexConn, which is designed to be

reusable. The synonym notation was parsed and output to `synonym` tags. The same kind of synonym ID resolving as with LexConn was performed to include synonym words, instead of just IDs, in the output. The POS tagset was directly mappable to the reduced app tagset. The sense tagset, being PDTB3, did not have to be mapped.

2.5 LDM-PT

The Portuguese Lexicon of Discourse Markers LDM-PT was created at the University of Lisbon by Mendes and Lejeune (2016) and is the only lexicon which does not use XML as its native format. Instead, it consists of a Microsoft Excel spreadsheet containing 228 connectives, one per row, from which an XML file can be created with two intermediate steps: First, the table has to be exported as a tab-separated text file similar to CSV; second, the tabbed-CSV file must be processed with a Perl script which converts it to XML. This script is part of the LDM-PT distribution. I fixed some issues with it which prevented the resulting XML from being accepted by the Saxon XSLT processor because it omitted some closing tags and included unwanted information in its output such as the column headers.

LDM-PT uses the PDTB3 tagset and has all three levels – class, type, and (if applicable) subtype – realized as different tags. Its POS tagset is almost identical to LexConn and therefore already largely the same as that which the app uses.

Similarly to LexConn, different readings of a word are separated into multiple entries. An example for an entry is shown in Listing 5. Because each entry is based on a row in the .xlsx spreadsheet, and the Perl script does not filter out empty cells, each entry contains exactly the same tags, but some of them are empty. It is clearly visible that there can be no more than one orthographical variant with no more than two parts, no more than two modifiers, one synonym, three examples, and one comment. It can also be seen that the tag names that the Perl script generates are very close to DiMLex.

There are some syntax-related fields which DiMLex does not yet have, namely `type`, `mood`, `tense`, `modifier1` and `modifier2`. However, except for `type`, those are empty in most entries. The `type` can be either of *primary connective*, *secondary connective*, or *altlex*. Primary connectives are those connectives which are fully grammaticalized function words, and secondary connectives are usually short phrases which always have connective meaning[7] (Rysová & Rysová 2015). *AltLex* (alternative lexicalization) stands out from the others because it is a term from the PDTB3 annotation guidelines (Prasad et al. 2007) and does, in fact, not denote a connective. Instead, it refers to such phrases that have a connective meaning in a particular context, but can't easily be exchanged with real connectives in other contexts. To be precise, a phrase is annotated as *AltLex* in the PDTB if there is a discourse relation between two clauses where no explicit connective (primary or secondary) is found, but it is also impossible

7 In LDM-PT, however, most multi-word connectives are also tagged as primary connectives. The *secondary connective* type is used only rarely.

```
<dmarkers>
	<dmarker word="a fim de que" id="dm1">
		<orth1 type="cont">
			<part1 type="phrasal">a fim de que</part1>
			<part2 type=""></part2>
		</orth1>
		<syn>
			<type>primary connective</type>
			<cat>csu</cat>
			<context>
				<mood>subjunctive</mood>
				<tense></tense>
			</context>
			<modifier1></modifier1>
			<modifier2></modifier2>
		</syn>
		<sem>
			<relation1>contingency</relation1>
			<relation2>purpose</relation2>
			<relation3>arg2-as-goal</relation3>
		</sem>
		<synonym lexicon="dimlex-en" entry-id="22">so that</synonym>
		<examples>
			<example1 source="CRPC">Por fim , a Comissão sugere um sistema de
etiquetagem das viaturas a fim de que o cliente possa fazer uma escolha com melhor
conhecimento de causa e passar a rolar mais por menos dinheiro , sem poluir o ambiente
, ao volante do que se chama já a viatura do futuro . </example1>
			<example2 source=""></example2>
			<example3 source=""></example3>
		</examples>
		<comment></comment>
	</dmarker>
</dmarkers>
```

Listing 5: The LDM-PT XML structure.

to insert an implicit connective because this would be redundant. Here is an example: “After trading at an average discount of more than 20% in late 1987 and part of last year, country funds currently trade at an average premium of 6%. *AltLex*[The reason:] Share prices of many of these funds this year have climbed much more sharply than the foreign stocks they hold.” (ibid., p. 7). The causal relation is expressed here by “the reason”, which is clearly not a connective, so the relation is lexicalized by an expression alternative to a connective.

The other LDM-PT-unique fields are very rarely filled out. The `mood` field can be either *subjunctive* or *infinitive*, `tense` is either *infinitive*, *inflected infinitive* or *participle*, `modifier1` only occurs twice in the whole lexicon and contains stronger forms of the base word (*“só após”* for *“após”* (“after”) and *“muito embora”* for *“embora”* (“though”)), and `modifier2` is always empty.

2.5.1 Format Conversion

Similarly to the LexConn conversion, the split-up entries for ambiguous connectives had to be merged by grouping them, first by word, then by word class. Of the described fields unique to this lexicon, only `type` was taken over into the DiMLex representation, as a new `type` attribute for the `entry` tag, so that it can be displayed by the app. Neither the sense tagset nor the POS

tagset had to be converted via a mapping, the app works with them out-of-the-box. Some used POS tags, e.g. *verb*, are not specifically represented in the app, but are found under the catch-all *other* name. No mapping is necessary for this, as the app automatically represents all unknown tags as *other*. All in all, the lexicon conversion could be completed using the general techniques that were already applied to the other lexicons, mostly those from the LexConn conversion.

3 The Web App

Connective-Lex.info is not just a static website, but a web application consisting of a frontend and a backend. The frontend is the part which runs in the browser and is visible to the user, and the backend is the part which runs on the server. Both frontend and backend are fully commented. The web app is built on the principles that define a single-page web application (SPA), a concept which sees wide adoption in current web development. This marks a shift in paradigm from earlier web app designs and is enabled by modern, standardized web technologies, namely HTML5, CSS 3, and JavaScript, the latter including Ajax for dynamic communication with the server without reloading the page. Key characteristics of modern SPAs, all of which also apply to Connective-Lex.info, are:

- There are no page changes or reloads. This means that the whole app lives inside a single HTML page which displays all app windows and interfaces.
- All rendering is done by the client. This means that the server no longer generates HTML pages or snippets to be displayed, as traditional PHP backends often do, but instead all display elements are created in the browser at runtime.
- Communication between client and server primarily uses JSON. This point is sometimes paraphrased by the catch-phrase “data-on-the-wire”. Related to the previous point, as it is not desirable to transfer HTML markup, transfers simply consist of data in JavaScript Object Notation (JSON), a concise text format which is both human- and machine-readable and is the native notation for JavaScript objects.
- The app dynamically loads data that it needs to update its views from the server via Ajax and updates the document via DOM manipulation, i.e. editing the tree structure of the HTML page.

The frontend of Connective-Lex.info displays the interface to the user and loads the lexicon data from the backend. Using the downloaded lexicons, it handles all user queries autonomously by running searches on the data and generating the resulting display elements on the fly.

On the other side of the SPA sits the backend which provides services to the frontend. SPAs usually rely on the Representational State Transfer (REST) principles for exchanging data and state between frontend and backend via JSON. The Connective-Lex.info backend provides an interface which complies with the REST paradigm, although it is so simple that many key elements of so-called RESTful services are not applicable here. Specifically, many of

the principles of REST revolve around the management of application state and its transfer from client to server with each request, so that the server ideally doesn't need to store any session state for the current users to keep track of where each user currently is in the app. This is a non-issue in this web app, because no state has ever to be transferred to the server. In fact, the backend is completely oblivious about the users and their actions in the app. It mostly acts as a static service which provides the list of lexicons and the lexicon data on request. It also generates JSON files from the lexicons' XML source files for simpler transfer and handling on the client side. The web app and this document are attached on a CD-ROM.

3.1 Files, Directories, & Infrastructure

Connective-Lex.info is currently hosted on a Linux web server at Webspace4All.eu and all data and code combined occupy about 7 MB of total space. The server runs the Apache server software for HTTP serving and supports PHP 7.0, the language for server-side scripting used by the backend. (It also supports other common technologies such as SQL which are not used by this web app.) The following directory structure exists on the server:

/httpdocs/connective-lex.info/

This is the main site directory and contains the part of the web app which is served via HTTP. This includes the public parts of the backend and the whole frontend. Its files and subdirectories are explained in detail in the sections on backend and frontend. This directory corresponds to the domain http://connective-lex.info.

/logs/

This is the directory for the Apache and PHP access logs and error logs. It is automatically generated by the server.

/php_include/

This directory contains the non-public parts of the web app backend, i.e. those parts which may not be exposed to the public. The settings file and internal parts of the backend logic reside here. This directory was specified as an *include_path* in the PHP settings in the admin interface of the hosting provider so that scripts in the main directory can use its contents without specifying a path.

/xml/

This is the data directory. It does not only contain the lexicon source XML data, but also all JSON files that the frontend may load by request. This includes lexicons, metadata, and the two tagset mapping files, one for POS tags and one for semantic senses. They each contain a JSON map from source tag to target tag, or to a list of possible target tags if no unique mapping is possible. This directory is not directly exposed to the public, but files in it are.

```
{
  "lexiconName": "DiMLex",
  "lexiconNameFull": "Diskurs-Marker-Lexikon",
  "version": "2.0",
  "originalReleaseDate": "1998",
  "currentReleaseDate": "2017-01-17",
  "languageEnglish": "German",
  "languageNative": "Deutsch",
  "locale": "de-DE",
  "islrn": "030-081-432-891-7",
  "homepage": "https://github.com/discourse-lab/dimlex",
  "parseInfo": {
    "posTagset": "dimlex",
    "senseTagset": "pdtb3-short"
  },
  "licenseInfo": {
    "copyrightDate": "2002-2017",
    "licenseName": "Creative Commons Attribution-NonCommercial-ShareAlike 4.0
International License",
    "licenseUrl": "https://creativecommons.org/licenses/by-nc-sa/4.0/"
  },
  "authorInfo": {
    "publisher": "Universität Potsdam, Department of Linguistics",
    "authors": [
      {
        "authorName": "Manfred Stede",
        "organisation": "Universität Potsdam, Department of Linguistics"
      },
      {
        "authorName": "Tatjana Scheffler",
        "organisation": "Universität Potsdam, Department of Linguistics"
      }
    ],
    "publications": [
      {
         "cite": "Manfred Stede. \"DiMLex: A Lexical Approach to Discourse Markers\"
In: A. Lenci, V. Di Tomaso (eds.): Exploring the Lexicon - Theory and Computation.
Alessandria (Italy): Edizioni dell'Orso, 2002.",
         "url": "http://www.ling.uni-potsdam.de/~stede/Papers/lenci02.pdf"
      }
    ]
  }
}
```


Listing 6: The metadata format by example of the DiMLex metadata.

3.2 Metadata

Complementing the actual lexicon data, I have created a metadata file for each lexicon which contains information about it in JSON format. Their contents were partially filled out by the creators of the respective lexicon. This data is used by the app in several ways, most importantly to display the names of the lexicons and their languages to the user. Some other metadata such as homepage, license, and author info is accessible via an *About* dialog. The metadata file for DiMLex is shown in Listing 6.

Most of the fields are self-explanatory. There are two technicalities. The `parseInfo` field contains the names of the tagsets used by the lexicon so that they can be mapped to the app tagsets. The `locale` is a language code that helps software handle international text correctly,

e.g. with regard to sorting rules or character case conversions. It is not used by the app at this time, but could become useful if the app is extended in the future. The ISLRN is the *International Standard Language Resource Number* (Choukri et al. 2011)[8], a web directory in which DiMLex and LDM-PT are listed. Most of the fields are optional, only `lexiconName` and `languageEnglish` are absolutely necessary.

3.3 The PHP Backend

The backend exists to serve the lexicon files to the frontend. It offers two RESTful services:

- A lexicon listing service, which supplies information about available lexicons and some associated data for each lexicon, such as the filename for metadata, date of last change, and similar info.
- A file service, which supplies JSON files requested by the frontend.

The backend consists of the following files, which reside in the `/httpdocs/connective-lex.info/` location unless another path is given:

filelist.php

Provides the REST interface for the lexicon listing service. It is called from a browser as http://connective-lex.info/filelist.php.

getfile.php

Provides the interface for the file service. It is called as http://connective-lex.info/getfile.php and takes a filename as an HTTP GET parameter, specified like this: http://connective-lex.info/getfile.php?file=DiMLex-2017.json. The filename is checked so that no paths can be submitted; this ensures that the backend is safe and only files from the data directory can be requested.

/php_includes/LexiconXmlToJsonConverter.php

This file contains the LexiconXmlToJsonConverter class with the implementation of the XML-to-JSON conversion and a secondary SettingsLoader class which reads the settings file and exposes the settings for the other components to use. This file cannot reside in the public directory because its classes may not be called from outside the backend.

/php_includes/settings.json

The settings file. It contains some settings which are read by the SettingsLoader class and define the behaviour of the other classes, as explained below.

Behind the scenes, the lexicon listing service not only creates an inventory of lexicon files on the server, but it also converts them from XML to JSON as needed. XML files are only converted if they are found to be newer than their corresponding JSON files, which are always

8 http://islrn.org

stored for later use. The conversion therefore only runs if a lexicon file is updated or a new one is added. There is an optional `force` parameter, provided as http://connective-lex.info/filelist.php?force=true, which overrides the standard behaviour and forces a reconversion of all XML files.

The following settings in `settings.json` are applicable to the lexicon listing service:

xmlDirectory (*Path*)

Specifies the path of the data directory relative to the main directory where `filelist.php` and `getfile.php` are located. For the current server file system structure, it must be `../../xml`.

autoConvert (*true*/*false*)

True if new XML files should automatically be converted to JSON whenever a file listing is requested. This enables fully automatic updates of lexicon files without having to force a new conversion.

prettyPrintJson (*true*/*false*)

Controls whether the JSON output of the scripts should be printed in a nicely readable format (true) or as short as possible (false). To minimize the amount of data that has to be transferred, this option can be set to false. If this option is changed, a forced reconversion is necessary to update the JSON files accordingly. For reference, the amount of data saved if this option is disabled is detailed in Table 1. Not only lexicon data is affected, but also all JSON output from the lexicon listing service. The lexicon sizes for minified JSON are about ⅔ of the source XML size and about ¼ of the pretty-printed size.

Lexicon	***XML size (KB)***	***Pretty-printed JSON size (KB)***	***Minified JSON size (KB)***	***Minified % of XML/PP-JSON***
DiMLex	298	813	200	67.1 % / 24.6 %
PDTB-DiMLex	105	317	71	67.6 % / 22.4 %
LexConn	400	964	283	70.8 % / 32.8 %
LICO	174	435	126	72.4 % / 29.0 %
LDM-PT	285	630	206	72.3 % / 32.7 %

Table 1: The sizes of the lexicon files in XML, pretty-printed JSON, and minified JSON, as well as the size ratio of the minified JSON to the XML and the pretty-printed JSON.

allowForceReparse (*true*/*false*)

Indicates whether a forced reparse of all lexicons is allowed by use of the `force` parameter. There is usually no reason to disallow it, but in case this parameter should ever be abused, such as during a denial-of-service (DoS) attack, forced reparses can be disallowed to save processing power on the server.

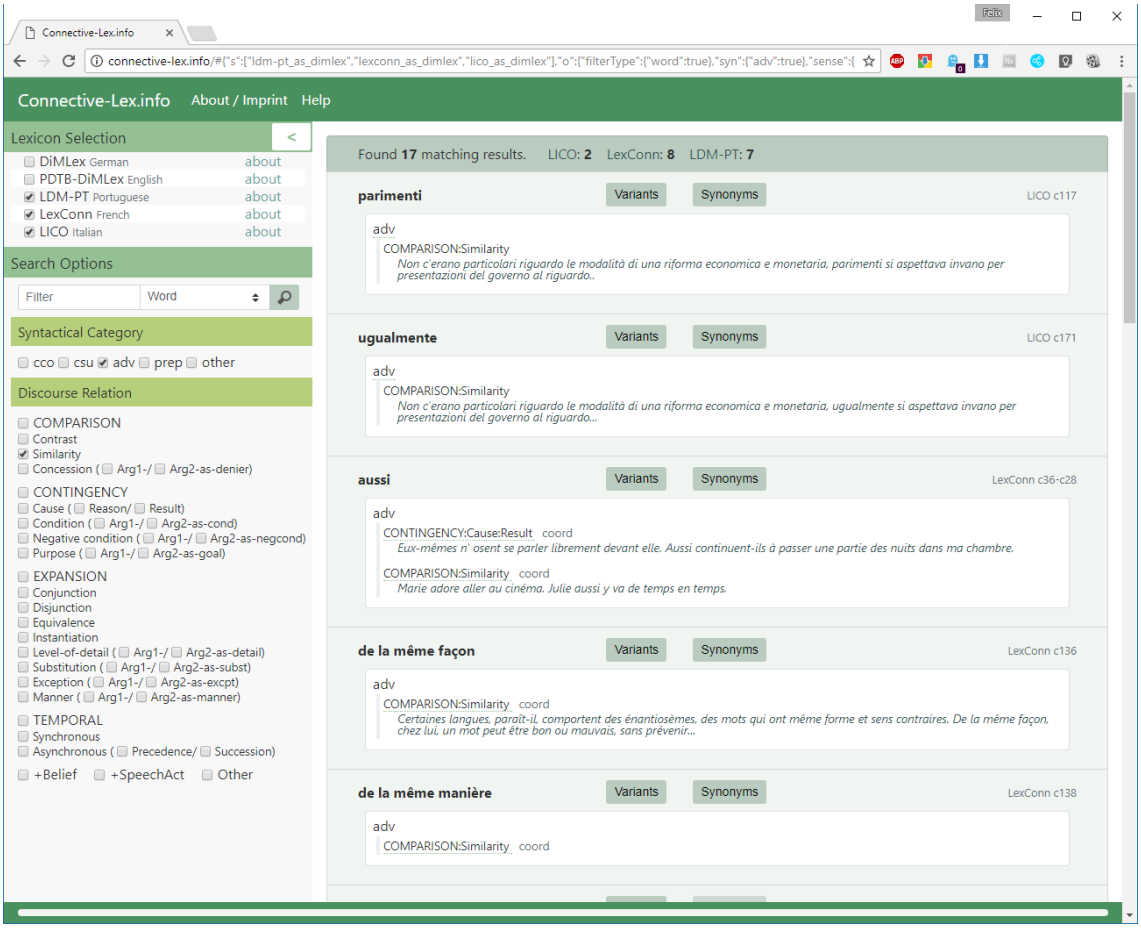

Figure 2: A screenshot of the Connective-Lex.info app displaying Italian, French, and Portuguese adverbs that express similarity.

3.4 The Single-Page Frontend

The frontend, shown in Figure 2, was built using techniques that were inspired by a popular SPA framework, Angular[9]. The build system it needs to produce webpages makes it somewhat complicated to set up, and without this build system, it is impossible to modify pages which makes it unsuitable if the app must be easily extendable in the future. Therefore, I mimicked the way SPAs are written with this framework, but developed Connective-Lex.info without any build system. Several JavaScript and CSS libraries were used instead:

- Bootstrap 4.0.0-alpha.6, a layout and display components library[10]
- jQuery 3.2.1, a library for Ajax and document manipulation, inter alia[11]
- JsRender/JsViews/JsObservable, a set of template and data binding libraries to bind to and display dynamic data[12]
- Lodash 4.17.4 for object manipulation[13]

9 https://angular.io
10 https://v4-alpha.getbootstrap.com. It is totally usable despite still being in alpha status.
11 https://jquery.com
12 https://www.jsviews.com
13 https://lodash.com

- lz-string for string compression[14]

Code used from other sources is indicated in the source code files via comments. The frontend has the following directory structure, located within the main directory:

index.html

The file in which all of the web app display elements live. It is automatically displayed when http://connective-lex.info is visited.

css/

The stylesheet directory with two files, `c24.css` and `simple-sidebar.css`. The former contains general page and element styles, the latter is concerned with the sidebar and its behaviour.

js/

The script directory which contains the five .js JavaScript files that make up the frontend logic.

lib/

This directory contains all the library files that are used by the app.

As it follows Angular programming patterns and jargon, the front end logic is split up into *services* and *components*. Services are classes which provide backend access or supply other data and reside in c24-services.js, and components are classes which display data by having templates bound to them in a Model-View-ViewModel (MVVM) design pattern style. They reside in `c24-components.js`.

The services are the *LexListService* for accessing the backend's lexicon list service, the *LexFileService* for accessing the backend's file service, and the *LocationDataService* which reads and writes application state to and from the browser's location bar. The latter enables sharing so-called deep links to specific search results.

The components use the services to aquire their data. The *LexSelectorComponent* initially loads the list of lexicons from the LexListService, looks up the metadata via the LexFileService and displays the *Lexicon Selection* section of the sidebar. It also handles lexicon loading and storage of the data. The *OptionsComponent* loads the initial options from the URL with which the app was loaded via the LocationDataService and displays the *Search Options* section of the sidebar. The *ResultsComponent* stores and displays the results after a search.

The lexicons are preprocessed immediately after loading by the *LexiconPreprocessor* helper class to adapt the object structure a bit so that lexicon entries can be more easily displayed by the template system when the result list is shown. Preprocessing also includes the tagset mapping for which `syn-maps.json` and `sense-maps.json` are loaded from the file

14 http://pieroxy.net/blog/pages/lz-string/index.html

service. Each tag in each entry is the looked up in the correct map as indicated by the metadata and replaced by the fitting tag(s) in the respective app tagset.

Searches are carried out by another helper class, *ResultsFilter*, which checks lexicon entries against the current filter settings.

The visible part of the web app is split up into

- a top navigation bar containing links to an *About and Imprint* dialog and a *Help* dialog,
- a sidebar containing all options including lexicon selection and filter settings,
- a large area for the result list,
- a small bottom bar with a progress indicator for file downloads and search processes.

If many search results are displayed, then the actual display will still take a noticeable amount of time during which the progress bar can't display any progress, because the rendering blocks all other processes. The interface of Connective-Lex.info is explained in more detail in the *Help* dialog that can be opened from the web app.

Via the settings file, `c24-settings.js`, some program options can be set for all users. The `compressUrlData` setting can be set to *true* to obtain a bit shorter, but totally unreadable deep links. The `storeSensesInUrl` setting can be deactivated if the sense names in deep links turn out to be too long. Finally, `useBootstrapTooltips` prevents the use of the Bootstrap library to display tooltips. Although the Bootstrap library generally works very well, there are still some bugs in the tooltips part, which seemed to be harmless during testing (spurious errors are logged to the browser's console and tooltips sometimes fail to close automatically). If the bugs turn out to be too annoying, all tooltips can be reverted to their native look.

The web app currently runs well on desktop systems with very modern browsers from March 2017 and newer. The dependance on modern browers is due to the use of some modern JavaScript features and notations which were only introduced recently (promises, classes, big-arrow functions). Supported browsers are recent versions of Chrome, Firefox, Edge, Safari, and Opera. Users of outdated browsers, such as Internet Explorer, see a big red error message because the app isn't running, even though I don't deliberately test for browser features. The message contains the list of supported browsers as specified above.

3.5 Adding New Lexicons

Connective-Lex.info works with any new lexicons in DiMLex format, and also makes it very easy to update existing lexicons. The lexicon XML file has to be put into the backend's data directory, `/xml/`. Additionally, a metadata file has to be provided which must have the same name as the lexicon, just that the extension must be `.meta` instead of `.xml`. Both file names should not contain any other dots and should also not contain spaces. The metadata structure should be copied from an existing metadata file, and the new info should then be filled in. If a POS or sense mapping is necessary because the new lexicons use different tagsets, the files

`syn-maps.json` and `sense-maps.json`, also in the data directory, must be edited to include a mapping from the lexicons tagsets to the app tagsets. These files can also be edited if a non-suitable mapping is discovered for an existing entry.

3.6 Outlook

This section contains some suggestions for improvement which could be added to the webapp at a later stage. These can be divided into linguistic improvements and usability improvements. Here are some usability improvements which could be worth trying:

- *Responsive Design*: It is possible to create a more mobile-friendly version without changing much code: only just the stylesheets need to be updated a bit to include alternative styles for smaller display areas.
- *Result Prerendering*: As the individual lexicon entries never change during execution of the app, it would be possible to create all the HTML needed for displaying each result at the beginning when the lexicons are loaded, or alternatively when an entry is first included in the result list. The effect would be much shorter rendering times, especially when many results have to be displayed. Getting this right is somewhat tricky, though.
- *Collation for filter words:* This means that the app could be updated to accept alternative spellings in filter words, such as omitting diacritics. For instance, it is already hard to enter a “ç” or an “ñ” on German keyboards; diaereses pose a problem on American keyboards (a workaround being the use of Alt codes). This issue will be more pressing if lexicons for languages with more diacritics are added, such as data from the Prague Discourse Treebank (Rysová et al. 2016).
- *Forward/backward navigation*: It would be nice to be able to navigate back to previous results using the browser’s navigation toolbar.

Some possible extensions that would be useful from a linguistic perspective include:

- *More filter options*: It is currently possible to filter by words, POS tags, senses, and source lexicon. However, several lexicons include other data, such as the type of connective, ordering info, or the cases that are assigned by prepositions, for example. Some of these attributes might be useful to filter by.
- *Sorting*: Results are currently sorted by the order in which the lexicons were loaded and then by the order in which the entries appear in the lexicons. Giving users the ability to sort them by other criteria would be a good addition.
- *Inline translation*: No one can speak all the languages for which lexicons might be added to the app. It would be extremely helpful to link the app to a translation web

service such as Google's[15] so that translations for words or examples can be viewed in-place without changing the page.

- *Statistics*: It might prove valuable to include some statistics in the form of tables or diagrams into the *About* dialog of each lexicon, providing insights on the kinds of words contained, such as the occurrence counts of POS or sense tags and other statistical info.

4 Conclusion

With Connective-Lex.info up and running, the biggest immediate impact on its usefulness lies in the availability of lexicon data for different languages. As new lexicons can easily be added, and existing ones amended with more data such as examples, the quality and amount of the data that the web app makes available is expected to increase over time. For DiMLex specifically, it would be a big improvement if examples were attached at reading level instead of word level. It is also hoped that other universities or work groups will will produce similar lexicons for other languages, which can then be incorporated into the app.

The web app provides increased readability and accessability for the lexicons. Its interface is not only useful for finding information on particular connectives, but can also aid lexicon developers in that it makes errors in the data quite easy to find, e.g. by looking at the entries that are found for *Other* POS or sense filter settings or simply by scanning through the entries that are displayed. All in all, the goals that were set for this project have been fully met. Further work is needed on the SDRT-to-PDTB3 mapping, and it would be beneficial for all users of the app if some of the suggested improvements could be implemented in the future.

15 https://translate.google.com

Bibliography

Nicholas Asher and Alex Lascarides. *Logics of Conversation*. Cambridge University Press, 2003.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. *Better Document-Level Sentiment Analysis from RST Discourse Parsing*. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. Lisbon, 2015.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. *TIGER: Linguistic Interpretation of a German Corpus*. In: Journal of Language and Computation (2) (pp. 597-620). 2004.

Khalid Choukri, Jungyeul Park, Olivier Hamon, and Victoria Arranz. *Proposal for the International Standard Language Resource Number*. 2011.

Margot Colinet. *Comparison of French and German Lexicons of Discourse Connectives.* Report for the STSM for TextLink. Universität Potsdam, 2015.

Laurence Danlos and Charlotte Roze. *Traduction (automatique) des connecteurs de discours*. In: Actes de TALN 2011. Montpellier, 2011.

Anna Feltracco, Elisabetta Jezek, Bernardo Magnini, and Manfred Stede. *LICO: A Lexicon of Italian Connectives.* CLiC-it/EVALITA. 2016.

William C. Mann and Sandra A. Thompson. *Rhetorical Structure Theory: Toward a Functional Theory of Text Organization*. In: Interdisciplinary Journal for the Study of Discourse 8 (3) (pp. 243–281). 1988.

Amália Mendes and Pierre Lejeune. *LDM-PT: A Portuguese Lexicon of Discourse Markers*. In: L. Degand, C. Dér, P. Furkó, and B. Webber (eds.). Conference Handbook TextLink – Structuring Discourse in Multilingual Europe Second Action Conference (pp. 89-92). Budapest: Debrecen University Press, 2016.

Andreas Peldszus and Manfred Stede. *From Argument Diagrams to Argumentation Mining in Texts: A Survey*. In: International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), 7(1) (pp. 1–31). 2013.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie Webber. *The Penn Discourse Treebank 2.0 Annotation Manual*. 2007.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Libio Robaldo, Aravind Joshi, and Bonnie Webber. *The Penn Discourse TreeBank 2.0*. In: Proceedings of the Sixth International Language Resources and Evaluation Conference (pp. 2961-2968). 2008.

Brian Reese, Julie Hunter, Nicholas Asher, Pascal Denis, and Jason Baldridge. *Reference Manual for the Analysis and Annotation of Rhetorical Structure (Version 1.0)*. Departments of Linguistics and Philosophy, University of Texas at Austin, 2007.

Charlotte Roze. *Base lexicale des connecteurs discursifs du français.* Master thesis, Université Paris Diderot. Retrieved from http://utilisateurs.linguist.univ-paris-diderot.fr/~croze. 2009.

Charlotte Roze, Laurence Danlos, and Phillipe Muller. *LEXCONN: a French Lexicon of Discourse Connectives.* In: Proceedings of Multidisciplinary Approaches to Discourse (MAD 2010). Moissac, 2010.

Magdaléna Rysová, Pavliná Synková, Jiří Mírovský, Eva Hajičová, Anna Nedoluzhko, Radek Ocelák, Jiří Pergler, Lucie Poláková, Veronika Scheller, Jana Zdeňková, and Šárka Zikánová. *Prague Discourse Treebank 2.0*. Data/software, ÚFAL MFF UK. Retrieved from http://hdl.handle.net/11234/1-1905. Prague, 2016.

Manfred Stede and Carla Umbach. *DiMLex: A Lexicon of Discourse Markers for Text Generation and Understanding*. In: Proceedings of the 17th International Conference on Computational Linguistics (2). Association for Computational Linguistics. 1998.

Manfred Stede. *DiMLex: A Lexical Approach to Discourse Markers*. In: A. Lenci, V. Di Tomaso (eds.): Exploring the Lexicon - Theory and Computation. Alessandria (Italy): Edizioni dell'Orso, 2002.

Manfred Stede and Silvan Heintze. *Machine-Assisted Rhetorical Structure Annotation.* In: Proceedings of the 20th International Conference on Computational Linguistics. Association for Computational Linguistics. 2004.

Bonnie Webber, Rashmi Prasad, Alan Lee, and Aravind Joshi. *A Discourse-Annotated Corpus of Conjoined VPs*. In: Proceedings of LAW X – The 10th Linguistic Annotation Workshop. Association for Computational Linguistics (pp. 22-31). 2016

Erklärung

Ich versichere, dass ich (Name: _____)

die Arbeit (Titel Seminar/Semester:_____

_____

selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln (z. B. Nachschlagewerke oder Internet) angefertigt habe. Alle Stellen der Arbeit, die ich aus diesen Quellen und Hilfsmitteln dem Wortlaut oder dem Sinne nach entnommen habe, sind kenntlich gemacht und im Literaturverzeichnis aufgeführt. Weiterhin versichere, ich, dass weder ich noch andere diese Arbeit weder in der vorliegenden noch in einer mehr oder weniger abgewandelten Form als Leistungsnachweise in einer anderen Veranstaltung bereits verwendet haben oder noch verwenden werden.

Die „Richtlinie zur Sicherung guter wissenschaftlicher Praxis für Studierende an der Universität Potsdam (Plagiatsrichtlinie) - Vom 20. Oktober 2010“, im Internet unter http://uni-potsdam.de/ambek/ambek2011/1/Seite7.pdf, ist mir bekannt.

Es handelt sich bei dieser Arbeit um meinen ersten/zweiten Versuch.

_____

Ort, Datum Unterschrift