
# CONCRETE

## CONCEPT CREATION TECHNOLOGY
## PROJECT NUMBER: 611733

SMALL OR MEDIUM-SCALE FOCUSED RESEARCH PROJECT
ICT - FUTURE AND EMERGING TECHNOLOGIES (FET)

---

# Deliverable 6.2:
# First report on framework and data

Jožef Stefan Institute (JSI) Version: 1.1, final

---

**Executive summary**
This document reports on the status of data and data-related services in ConCreTe. The developed software components provide access to two important knowledge bases: ConceptNet and WordNet. The components enable programmatic interaction with these two data resources directly from the project's common software platform, named ConCreTeFlows. In addition, a collection of datasets was gathered and prepared in accordance with the specifications proposed in Deliverable D6.1. The datasets are from the domains of recipes, morals and proverbs, as well as single sentences of very diverse textual sources (fairy tales, scientific papers, news articles) for which a collection and management infrastructure was set up. The document presents also the proposed processes and tools for data access and annotation, both programmatic and manual.

---

| **Dissemination Level** | | |
|---|---|---|
| PU | Public | X |
| PP | Restricted to other programme participants (including the Commission Services) | - |
| RE | Restricted to a group specified by the Consortium (including the Commission Services) | - |
| CO | Confidential, only for members of the Consortium (including the Commission Services) | - |

## Revision history

| Document administrative information | |
|---|---|
| Project acronym: | ConCreTe |
| Project number | 611733 |
| Deliverable number: | D6.2 |
| Deliverable full title: | First report on framework and data |
| Deliverable short title D6.2: | First report on framework and data |
| Document identifier: | ConCreTe-del-D6.2-FirstReportData-final-v1.1 |
| Lead partner short name: | Jožef Stefan Institute (JSI) |
| Report version: | 1.1, final |
| Report preparation date: | 30/09/2014 |
| Dissemination level: | PU |
| Nature: | R = report |
| Lead author: | Martin Žnidaršič |
| Co-authors: | Dragana Miljković, Matic Perovšek, Senja Pollak, Janez Kranjc, Darko Cherepnalkoski, Nada Lavrač |
| Status: | Draft |

Changes to this document are detailed in the change log table below.

## Change log

| Date | Editor | Summary of changes made |
|---|---|---|
| 25/07/2014 | Martin Žnidaršič | Sectioning |
| 05/08/2014 | Dragana Miljković, Matic Perovšek, Senja Pollak | Datasets description |
| 20/08/2014 | Martin Žnidaršič | Data annotation section |
| 25/08/2014 | Dragana Miljković | Data widgets introduction |
| 02/09/2014 | Martin Žnidaršič | Data widgets text, Introduction, Conclusion |
| 17/09/2014 | Nada Lavrač | Quality control and final revision |
| 23/09/2014 | Sue White | Language corrections |

# Contents

# 1 Introduction

This document reports on the status of data and data-related services in ConCreTe at the end of the first year of the project. The described data resources and tools were developed to meet the anticipated further needs of the project, as at the time of development there were still very few project requirements and needs explicitly defined. An exception to this is the WordNet access component, which was explicitly identified as useful and necessary by several members of the project consortium.

The document is structured as follows. Section 2 presents the resources, while Section 3 presents the developed tools for data access and annotation.

# 2 Data services and datasets

This section describes prototype data services and datasets that were prepared for the use in ConCreTe: prototype data services are presented in Section 2.1, and the datasets in Section 2.2.

## 2.1 Data access services

This section is devoted to the description of software components that provide access to two important data resources directly from the common software platform of the project. These two resources, ConceptNet[1] and WordNet[2], are often used in computational creativity processes as sources of common knowledge and as means for understanding written text. Request to provide access to ConceptNet was supported as an exemplary case, while the access to WordNet was required by several consortium members at the main project meeting of this reporting period.

In the following, we use the term *widget* to denote any software component used in computational creativity workflows of the workflow management and execution platform ConCreTeFlows, which is used as the infrastructural backbone of the ConCreTe project.

### 2.1.1 ConceptNet data widget

ConceptNet is a semantic network of common sense knowledge that contains more than 1.6 million edges connecting more than 300,000 nodes. ConceptNet uses many data sources, including: Wikipedia, Open Mind Common Sense (OMCS), sister projects to OMCS in Portuguese [1] and Dutch [3], English Wiktionary, DBpedia, ReVerb, WordNet, etc. ConceptNet considers numerous languages: English, Portuguese, Korean, Japanese, Dutch, Spanish, French, Arabic, Chinese and many others. ConceptNet has a closed class of *Relations*, expressing connections between *Concepts*.

Below we list the current set of relations (with the questions they provide an answer to):

IsA — What kind of thing is it?
HasA — What does it possess?
PartOf — What is it part of?
UsedFor — What do you use it for?
AtLocation — Where would you find it?
CapableOf — What can it do?
MadeOf — What is it made of?
CreatedBy — How do you bring it into existence?
HasSubevent — What do you do to accomplish it?
HasFirstSubevent — What do you do first to accomplish it?

[1] http://conceptnet5.media.mit.edu/
[2] http://wordnet.princeton.edu/

HasLastSubevent — What do you do last to accomplish it?
HasPrerequisite — What do you need to do first?
MotivatedByGoal — Why would you do it?
Causes — What does it make happen?
Desires — What does it want?
CausesDesire — What does it make you want to do?
HasProperty — What properties does it have?
ReceivesAction — What can you do to it?
DefinedAs — How do you define it?
SymbolOf — What does it represent?
LocatedNear — What is it typically near?
ObstructedBy — What would prevent it from happening?
ConceptuallyRelatedTo — What is related to it in an unknown way?
InheritsFrom — (not stored, but used in some applications)

ConceptNet is a useful resource of common knowledge and facts, therefore it is often used in solutions to computational creativity problems [10, 9]. As the knowledge from ConceptNet will likely be used in some of the workflows of ConCreTe, a set of prototype components for the use of this resource were developed.

In ConCreTeFlows, the access to data in ConceptNet is managed with three widgets:

- Conceptnet Create Search Query
- Conceptnet Execute Query
- Conceptnet Get Triplets

The first widget, the *Conceptnet Create Search Query*, shown in Figure 1, offers a graphical user interface for designing a properly formatted *Search* query for ConceptNet. The ConceptNet's API supports three kinds of queries: (I) *Lookup*: accesses a known ConceptNet resource, (II) *Search*: outputs the concepts corresponding to the search word, and (III) *Association*: enables searching for concepts similar to the ones provided in the query. The *Search* action is usually the initial and most important one, thus the query is currently of this kind.

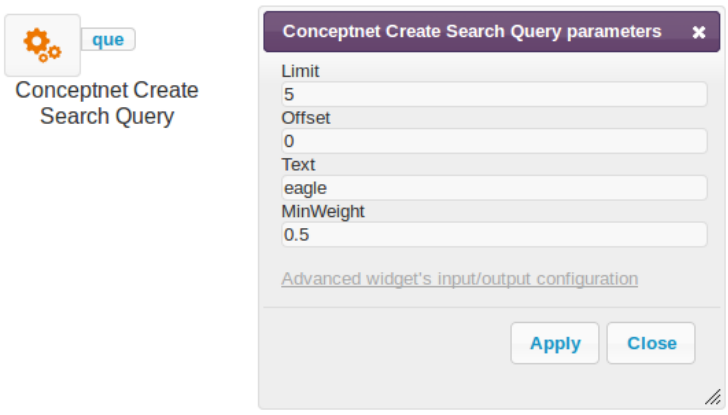


Figure 1: The *Conceptnet Create Search Query* widget and its parameters.

ConceptNet allows ten attributes to be used in the design of queries, out of which the *Conceptnet Create Search Query* widget supports the most important four as widget parameters[3]:

[3]The parameters can be easily and on-the-fly turned into inputs, if desired: right click on the widget; select *Parameters* and click *Advanced widget's input/output configuration* where any parameter can be dragged from the parameter box into an input box.

- Limit: the limit of the number of returned results
- Offset: the number of results in the response that are skipped from the start (useful for gradual collection of results)
- Text: the search word
- MinWeight: minimal allowed weight of an edge

The *Conceptnet Execute Query* widget conducts the communication with the ConceptNet API, where it executes the query from its input. The query typically comes from the *Conceptnet Create Search Query* widget as its output, but this is not mandatory. Alternatively, the query can be created also manually, as a free text input in the *Create String* widget. In this way, the query can be arbitrary and can use also the features that are not supported by the *Conceptnet Create Search Query* widget. De-coupling of the first two ConceptNet widgets in such a way allows for the creation and use of alternative widgets for ConceptNet query production, which might be useful in specific use cases. The output of the *Conceptnet Execute Query* widget is the entire response of ConceptNet expressed in JavaScript Object Notation (JSON).

An example of such a response for a single object (edge) looks as follows:


```
{
  "edges": [
    {
      "context": "/ctx/all",
      "dataset": "/d/dbpedia/en",
      "end": "/c/en/bird",
      "endLemmas": "bird",
      "features": [
        "/c/en/bald_eagle /r/IsA -",
        "/c/en/bald_eagle - /c/en/bird",
        "- /r/IsA /c/en/bird"
      ],
      "id": "/e/7b95cadd534a6524c1537d29614ef6398345b24b",
      "license": "/l/CC/By-SA",
      "nodes": [
        "/r/IsA",
        "/c/en/bird",
        "/c/en/bald_eagle"
      ],
      "rel": "/r/IsA",
      "relLemmas": "",
      "score": 15.790956,
      "source_uri": "/or/[/and/[/s/activity/omcs/omcs1,_possibly_free_text/,/s/
      contributor/omcs/GARY/]/,/and/[/s/activity/omcs/omcs1,_possibly_free_text/
      ,/s/contributor/omcs/Jake512/]/,/and/[/s/activity/omcs/omcs1,_possibly_
      free_text/,/s/contributor/omcs/guru1/]/,/s/dbpedia/3.7/]",
      "sources": [
        "/s/activity/omcs/omcs1,_possibly_free_text",
        "/s/contributor/omcs/GARY",
        "/s/contributor/omcs/Jake512",
        "/s/contributor/omcs/guru1",
        "/s/dbpedia/3.7"
      ],
      "start": "/c/en/bald_eagle",
      "startLemmas": "bald eagle",
      "surfaceText": "[[Bald eagles]] are [[birds]]",
```

```
    "text": [
      "bald eagle",
      "bird",
      ""
    ],
    "timestamp": "2014-04-16T22:39:31.374Z",
    "uri": "/a/[/r/IsA/,/c/en/bald_eagle/,/c/en/bird/]",
    "weight": 3.087463
  },
```


As the JSON response of ConceptNet is not easy to comprehend and use in other components, we added another widget named *Conceptnet Get Triplets* which filters the JSON response for *concept-relation-concept* triplets and outputs them as a Python list, either with or without labels of their type and language. Again, the decoupling of the triplet filtering part allows for the other components to access and use the entire JSON response of ConceptNet.

An example workflow with the three ConceptNet widgets and some exemplary inputs and outputs is shown in Figure 2.

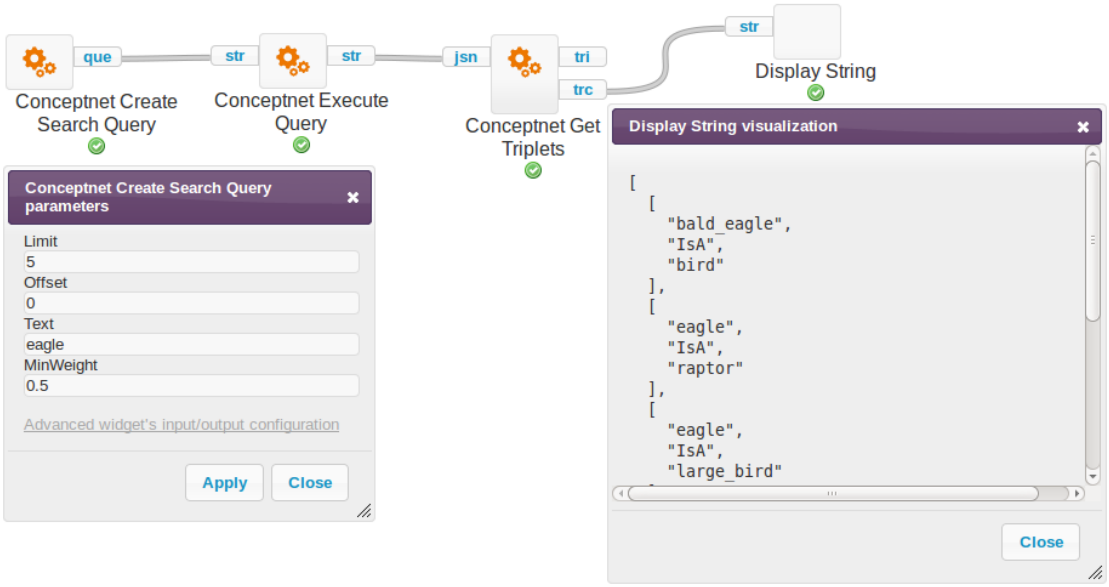


Figure 2: Three ConceptNet widgets in a typical workflow configuration.

### 2.1.2 WordNet data widget

WordNet [4] is a lexical database, where words that denote a specific concept are grouped into synonym sets, called *synsets*. The concepts are enriched with semantic information and linked with other synsets by various relations. In WordNet the main relation between words is synonymy, where two or more words represent the same concept. The most frequent relation between synsets is the super-subordinate relation (hyperonymy / hyponymy relation). This means that a more general concept, like {education}, is related to the more specific ones, like {school} and {faculty}. Moreover, WordNet contains meronymy relation (part-whole) between synsets, for example: {chair} and {legs}. Adjectives are also organized based on antonymy relation, like {wet} and {dry}. For more information about the nature, structure and aims of WordNet see [4] and [5].

WordNet is a valuable resource in the field of language technologies. It is used in ontology research [6], for resolving disambiguation problems [8], metaphora research [11], etc.

Due to WordNet's considerable potential for computational creativity research, we have developed access widgets in ConCreTeFlows. Access to WordNet is enabled by two ConCreTeFlows widgets that access WordNet through the *nltk*[4] Python library:

- WordNet query, and
- WordNet synset.

The *WordNet query* widget expects a search string as a parameter (or input), it queries WordNet through the *nltk* library with the string and outputs a Python list with IDs of corresponding synsets (see Display1 visualization in Figure 3).

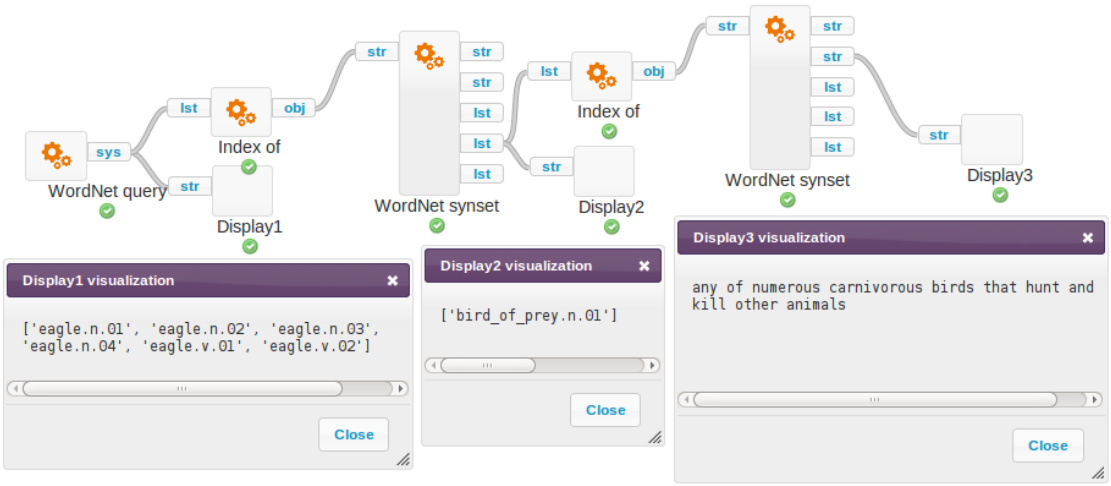


Figure 3: Example workflow of WordNet widgets.

The second widget, *WordNet synset* expects a single synset ID as its input and outputs:

- the **name** of the synset object,
- its **definition**,
- a list of **examples**,
- a list of **hypernyms** and
- a list of **lemmas**.

As the first widget outputs a list and the second one expects a single ID, there must be a standard *Index of* widget used in between to select a single list element by its index (see Figure 3). The *WordNet synset* widgets can also be used sequentially, so that the output of one such widget is the input to another one as in the situation shown in Figure 3.

The WordNet widgets and the ConceptNet widgets have simple and compatible inputs and outputs, thus they can be used in combination, like in the workflow shown in Figure 4 where we search for *mouse* in WordNet and then use its hypernym in a query to ConceptNet.

## 2.2 Datasets

This section presents a collection of prototype datasets that were prepared for the use in ConCreTe. Their primary role as prototype cases is to serve for demonstration purposes in experimental implementations and to spur discussion on further data requirements.

[4] http://www.nltk.org/

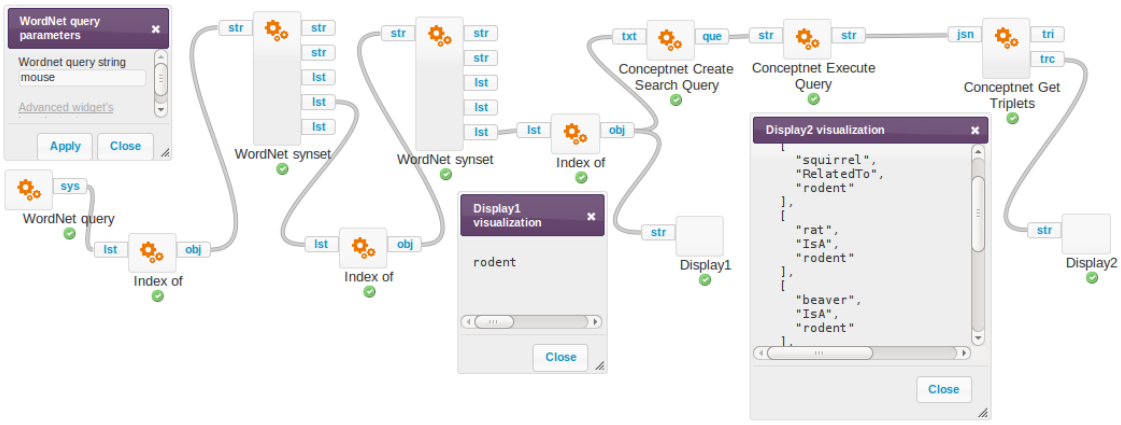


Figure 4: Example workflow of a combination of WordNet and ConceptNet widgets.

The selection of datasets is either due to previously documented cases of using such data in computational creativity tasks or simply due to their estimated potential for use in the ConCreTe project. Most of these datasets follow strict formatting specifications that were set in Deliverable D6.1 and are available online. As exemplary datasets, they are typically of a small size and are very diverse. Two of them contain recipes, a type of content that is expected to be well suited as targets for component exchange or blending. Creation of recipes [12, 2, 15, 14] is one of the classic domains in computational creativity. Next, there are datasets of fables' morals and proverbs, which are both condensed representations of valuable common knowledge. Finally, there are three datasets of single sentences from three diverse domains: fairy tales, science papers and news articles. These are intended for experimental analysis of the specific and common features that are characteristic of these diverse types of text.

### 2.2.1 Cocktail recipes dataset

The *Cocktail recipes dataset* is an RDF dataset of cocktail recipes. In its raw form it is an XML-RDF text file. It is available directly at:

`http://concreteflows.ijs.si/static/creativity/CCdata/cocktailrecipes.rdf`

and contains entries such as:

```
<amf:Recipe rdf:about="http://conceptcreationtechnology.eu/resource/data#cr1">
  <amf:Fn>Mai Tai</amf:Fn>
  <amf:Ingredient>dark rum</amf:Ingredient>
  <amf:Ingredient>light rum</amf:Ingredient>
  <amf:Ingredient>lime juice</amf:Ingredient>
  <amf:Ingredient>orange curacao</amf:Ingredient>
  <amf:Ingredient>orange juice</amf:Ingredient>
  <amf:Instructions>Shake the ingredients in a shaker with ice and put into...
  </amf:Instructions>
</amf:Recipe>
```

However, the dataset is available also through a dedicated *RecipeFilter* widget in ConCreTe-Flows, as shown in Figure 5. This widget is made to provide access and filtering for all RDF recipe datasets of ConCreTe. Based on the selected RDF file and the ingredient, it outputs the recipes that contain the specified ingredient in two RDF formats (RDF-XML and N3) and as a native Python object (Graph of the *rdflib* library). By omitting the ingredient and leaving empty string as input, the widget outputs the entire RDF contents of the given file.

This prototype dataset contains 22 data entries at the time of writing this deliverable.

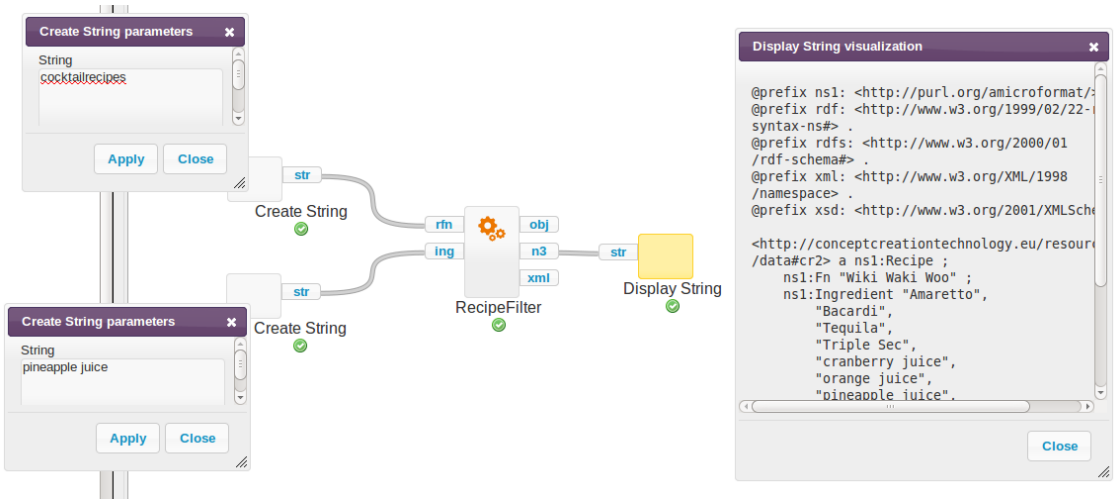


Figure 5: The *RecipeFilter* widget with inputs and outputs open for preview.

### 2.2.2 Pancakes recipes dataset

The *Pancake recipes dataset* is an RDF dataset of pancake recipes. In its raw form it is an XML-RDF text file. It is available directly at:

`http://concreteflows.ijs.si/static/creativity/CCdata/pancakerecipes.rdf`

and contains entries in the following format:

```
<amf:Recipe rdf:about="http://conceptcreationtechnology.eu/resource/data#cr1">
  <amf:Fn>Lemon Pancakes</amf:Fn>
  <amf:Ingredient>lemon zest</amf:Ingredient>
  <amf:Ingredient>salt</amf:Ingredient>
  <amf:Ingredient>eggs</amf:Ingredient>
  <amf:Ingredient>lemon juice </amf:Ingredient>
  <amf:Ingredient>poppy seeds</amf:Ingredient>
  <amf:Ingredient>sugar</amf:Ingredient>
  <amf:Ingredient>flour</amf:Ingredient>
  ...
  <amf:Instructions>Mix all ingredients and bake pancakes.
  </amf:Instructions>
</amf:Recipe>
```

The current prototype of the *Pancake recipes dataset* contains 20 entries. Similar to the *Cocktail recipes dataset* it is possible to filter this dataset with various kinds of search words through a dedicated *RecipeFilter* widget in ConCreTeFlows (Figure 5).

### 2.2.3 Aesop's fables' morals dataset

Aesop was a Greek fabulist, known as author of numerous fables, which are characterized by animals, which usually have human characteristics and solve everyday problems. As fables are narrations intended to convey a useful truth, their morals make them interesting for use in computational creativity research [13, 10].

The Aesop's fables' morals dataset is a collection of 25 morals. Each entry in the dataset contains one fable's moral. The entries have the following RDF format:

```
<rdf:Description rdf:about="http://conceptcreationtechnology.eu/resource/data#moral1">
  <dc:Description>It is easy to propose impossible remedies.</dc:Description>
</rdf:Description>
```

The dataset is available directly at:

`http://concreteflows.ijs.si/static/creativity/CCdata/Aesop_morals2.rdf`

#### 2.2.4 English proverbs dataset

English proverbs are concise and simple sayings reflecting the life wisdom or wise thoughts, often written in a cynical or humorous way. Their symbolic meaning offers immense potential for use in computational creativity, especially in the domain of metaphor creation.

The English proverb dataset consists of 661 proverbs obtained from English Wikipedia. Each entry in the dataset is related to one English proverb. Entries are in the following XML format:

```
<proverb>Actions speak louder than words.</proverb>
```

The dataset is available at:

`http://concreteflows.ijs.si/static/creativity/CCdata/englishproverbs.xml.`

#### 2.2.5 Single sentences datasets

In the following, we describe three datasets of single sentences from three different domains. All of them contain random sentences from their specific sources. The datasets allow for a number of analyses to be conducted (for example the analyses of style, vocabulary, sentence structure and other specifics of each source). The datasets are prototypes and are small in size, however, for the science papers and news articles we have put in place a system for automatic collection, enabling the production of very large datasets of the kind.

**Science paper dataset.** Science paper sentence dataset is a collection of 1,000 sentences, which were randomly chosen from 1,000 random articles, published on PubMed Central (PMC) in the last 10 years. Each line in the dataset file represents one such sentence. The dataset is currently available upon request only to the project partners.

**Fairy tales dataset.** This dataset consists of 100 sentences sampled randomly from fairy tales of Hans Christian Andersen. The dataset is in RDF format and is available from:

`http://concreteflows.ijs.si/static/creativity/CCdata/Andersen_100randomsentences.rdf`

**News articles dataset.** The news articles dataset contains 1,000 distinct sentences gathered from various English-language news sources. Each sentence comes from a different, randomly chosen, news article in the period between January 1, 2014 and August 1, 2014. The dataset is currently available upon request only to the project partners.

## 3 Data access and annotation

The data prepared for the use in the project will be accessed and edited through the shared workflow composition tool ConCreTeFlows (see Deliverable D6.4). Other means of access can be made available in case of such a need, but the access through ConCreTeFlows is the default one.

In ConCreTeFlows, data is accessed through widgets. The currently developed widgets for this purpose are presented in Section 2.2. A single dataset can be accessed in several ways, by several widgets. Similarly, various kinds of data annotation operations can be designed and implemented this way.

Regarding the involvement of the user in data annotation operations, the annotation can be (I) programatic, (II) programatic with user-defined parameters, or (III) manual. In Section 3.1 we decribe the first two kinds of annotations and in Section 3.2 we describe the latter one. In both we provide example solutions and present demonstrations of their use in ConCreTeFlows.

## 3.1 Programmatic annotation

In many cases we can automate or semi-automate a data annotation process and whenever this is possible, a programmatic (automatic) solution is preferred over the manual one as it allows for annotations of larger quantities of data [7].

In ConCreTeFlows, automatic annotation of data can be done by widgets, but it can even be done by calls to Web services. In the following, we present an example of a ConCreTeFlows widget for automatic annotation. Its purpose is automatic and semi-automatic annotation of recipe data sets (such as the ones described in Sections 2.2.1 and 2.2.2) with allergy warnings. Hence, this widget is named *AllergyAnnotation*. As input it takes a recipe in RDF (N3) format and annotates each recipe with warnings for the usual allergens[5]. The widget allows also for interactive semi-automatic annotation, by considering also the user-defined allergens, for which it checks the input recipes and creates the appropriate annotations.

An example workflow with the annotation widget is shown in Figure 6, where we can see one of the recipes before annotation (Display String 1) and after the annotation (Display String 2), with also the input window for the user defined allergens (Create String 3 parameters). Note that in the annotated data item, there is a new namespace ($ns2$) used for the predicate. The URI used for this predicate is:

`http://conceptcreationtechnology.eu/resource/model#allergywarning`

and it follows the prescriptions set in Deliverable D6.1.

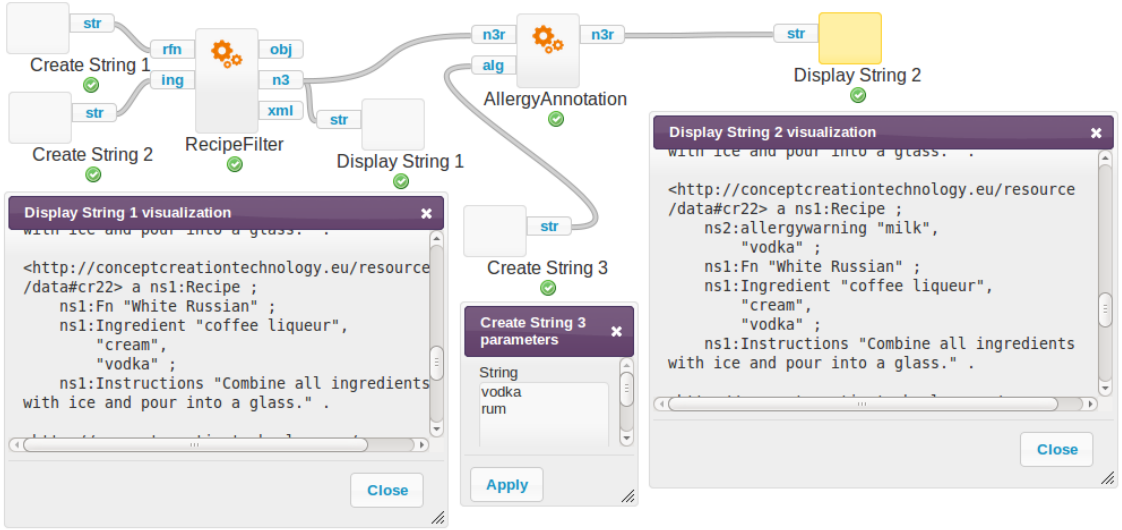


Figure 6: Example of using programatic annotation. The inputs to the *AllergyAnnotation* widget and its effects are shown.

[5]Currently only the milk allergy is considered in this prototype.

Programmatic components of this kind, even in the form of independent Web services, can be developed and used for automated and semi-automated annotation in ConCreTeFlows. A completely manual annotation requires a different approach.

## 3.2 Manual annotation

In some situations and for some use cases, manual annotation cannot be avoided. This is the case when we deal with annotations that are not programmable for some reason (for example subjective opinions and assessments).

The common software platform of ConCreTe supports manual annotation with a special type of widgets. Namely, widgets can be designed in a way to open their graphical user interface in the form of a dedicated Web page. Any user input on such a page can be collected and can influence the software components that follow it in the workflow. Such a dedicated Web page interface can be for example designed as an interactive data annotation form. An example of such an interface is shown in Figure 7 on an example of sentiment annotation.

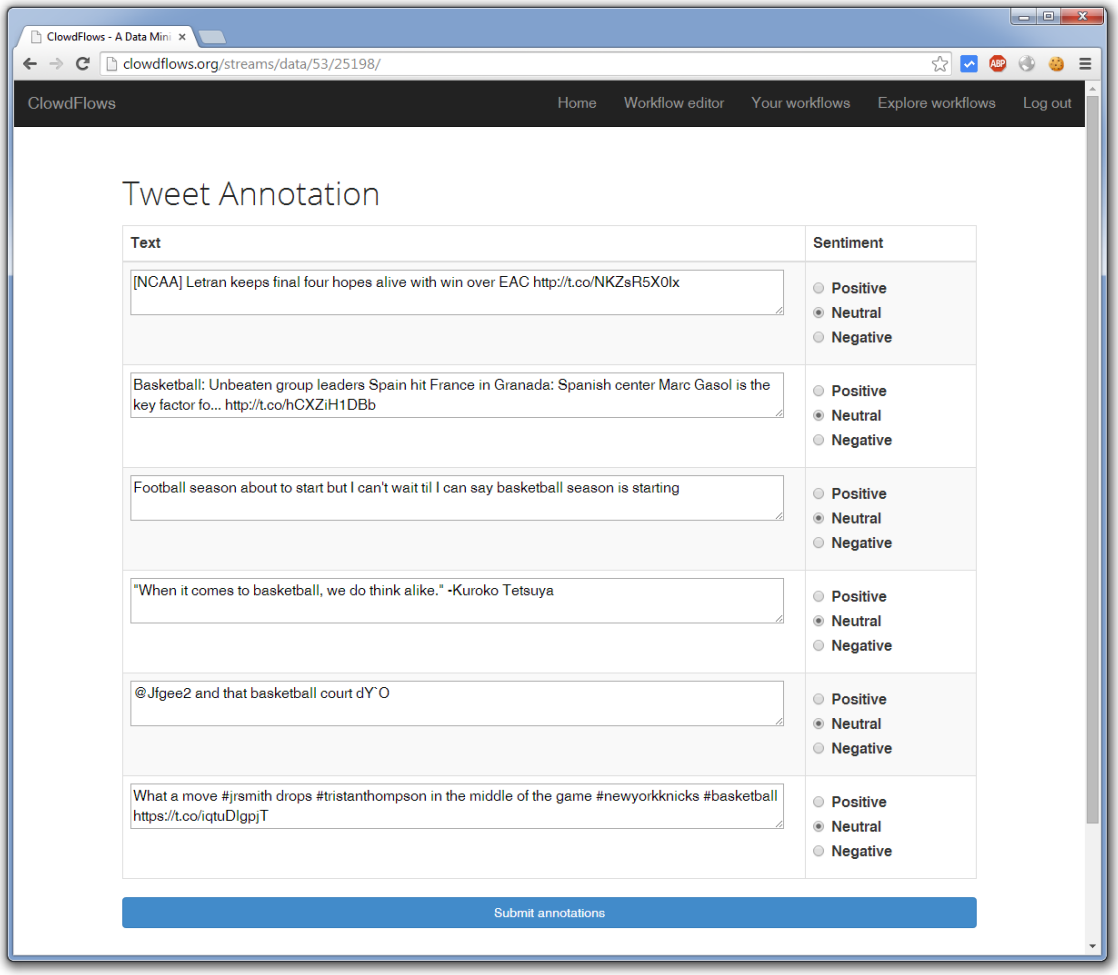


Figure 7: Example of an interactive user interface of a widget. The use case depicts short sentence (tweet) annotation from a ClowdFlows workflow.

Having the user interface of a widget in the form of a Web page is particularly convenient for data annotation, as the annotation interface can be distributed simply by distributing the URL of

the annotation form, which can be used by the annotators not only without any installation, but also without even being logged into the online platform.

Any need for manual data annotation in ConCreTe, either directly in the workflows of ConCreTeFlows, or otherwise, can thus be met by implementing a widget with an interactive Web page interface. Such an implementation is case specific and might not be trivial, but the ease of task distribution and the reduced and practically eliminated overhead of the annotation set-up help in reducing the effort of data annotation where it is the most important.

# 4 Conclusion

The framework for data representation, management and use, which was proposed in Deliverable D6.1, was to a large extent followed in the development of prototype software components reported in this document. At the time of writing this report, the developed dataset and data service prototypes are used primarily for demonstrating ConCreTeFlows workflows. The resources and the corresponding resource management methodologies, however, enable their use also in more elaborate experiments and applications which are already emerging from some of the software components as reported in Deliverable D6.4.

Further work on data resources and tools will be determined by the requirements and needs that will be spurred both by the experimentation with the prototypes reported here and by the advances made in the methodological tasks of the project.

# References


[1] Junia Anacleto, Henry Lieberman, Marie Tsutsumi, Vânia Neris, Aparecido Carvalho, Jose Espinosa, and Silvia Zem-Mascarenhas. Can common sense uncover cultural differences in computer applications? In *World Computer Congress*, 2006.

[2] AP Davis. Digital gastronomy: When an IBM algorithm cooks, things get complicated–and tasty, 2013.

[3] Nienke Eckhardt. *A Kid's Open Mind Common Sense*. PhD thesis, Tilburg University, 2008.

[4] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[5] Christiane Fellbaum. Wordnet and wordnets. In Alex Barber, editor, *Encyclopedia of Language and Linguistics*, pages 2–665. Elsevier, 2005.

[6] Fernando Gomez. Grounding the ontology on the semantic interpretation algorithm. In *Proceedings of the Second Global WordNet Conference*, pages 124–129, Brno, Czech Republic, 2004.

[7] Tony Hey and Anne Trefethen. The data deluge: An e-science perspective. *Grid Computing: Making the Global Infrastructure a Reality*, pages 809–824.

[8] Stan Szpakowicz Li, Xiaobin and Stan Matwin. A wordnet-based algorithm for word sense disambiguation. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1368 – 1374, Montreal, 1995.

[9] Maria Teresa Llano, Rose Hepworth, Simon Colton, John Charnley, and Jeremy Gow. Automating fictional ideation using conceptnet. In *Proceedings of the AISB14 Symposium on Computational Creativity*, 2014.

[10] Maria Teresa Llano, Rose Hepworth, Simon Colton, Jeremy Gow, John Charnley, Mark Granroth-Wilding, and Stephen Clark. Baseline methods for automated fictional ideation. In *Proceedings of the International Conference on Computational Creativity*, 2014.

[11] Birte Lonneker. Is there a way to represent metaphors in wordnets? insights from the hamburg metaphor database. In *Proceedings of the ACL-03 Workshop on the Lexicon and Figurative Language*, Sapporo, Japan, 2003.

[12] Richard G Morris, Scott H Burton, Paul M Bodily, and Dan Ventura. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *Proc. Int. Conf. Comput. Creativity (ICCC 2012)*, pages 119–125, 2012.

[13] Matic Perovšek, Bojan Cestnik, Tanja Urbančič, Simon Colton, and Nada Lavrač. Towards narrative ideation via cross-context link discovery using banded matrices. In *Advances in Intelligent Data Analysis XII*, pages 333–344. Springer, 2013.

[14] Nan Shao, Pavankumar Murali, and Anshul Sheopuri. New developments in culinary computational creativity. In *Proc. Int. Conf. Comput. Creativity (ICCC 2014)*, 2014.

[15] Lav R Varshney, Florian Pinel, Kush R Varshney, Debarun Bhattacharjya, Angela Schoergendorfer, and Yi-Min Chee. A big data approach to computational creativity. *arXiv preprint arXiv:1311.1213*, 2013.