Internal Report

Deliverable 2.2: Implementing workflows, developed in WP1, in the ClowdFlows development environment

Jozef Stefan Institute Version 1 FINAL

Document administrative information	
Project acronym:	HinLife
Project number:	J7-7303
Deliverable number:	D2.2
Deliverable full title:	Implementing workflows, developed in WP1, in the ClowdFlows development environment
Document identifier:	HinLife-del-D2.1-ClowdFlows-workflows-v0.2.docx
Lead partner short name:	Jožef Stefan Institute
Report version:	0.2, final
Report preparation date:	20/02/2018
Lead author:	Jan Kralj
Co-authors:	Nada Lavrač
Status:	Final

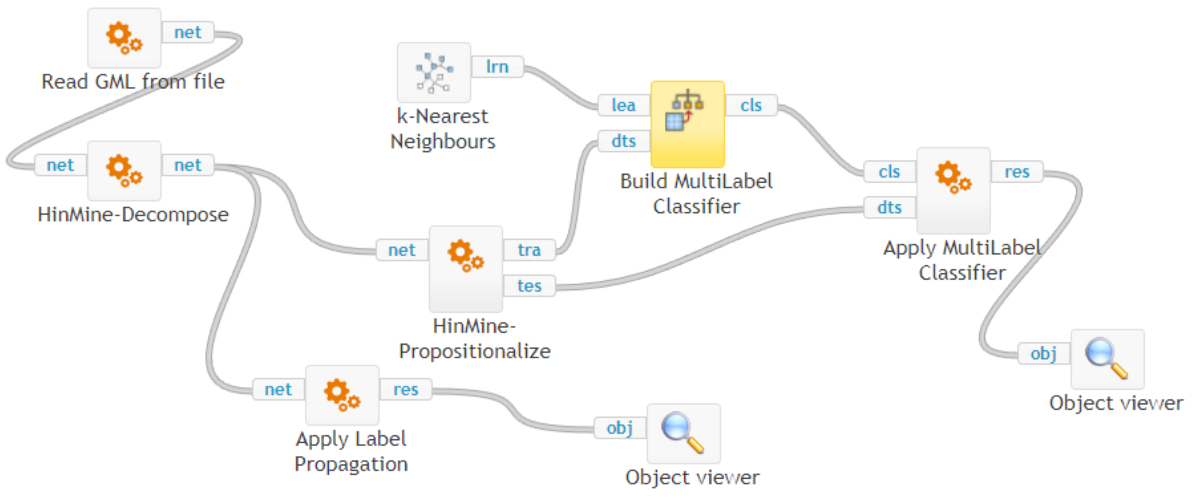


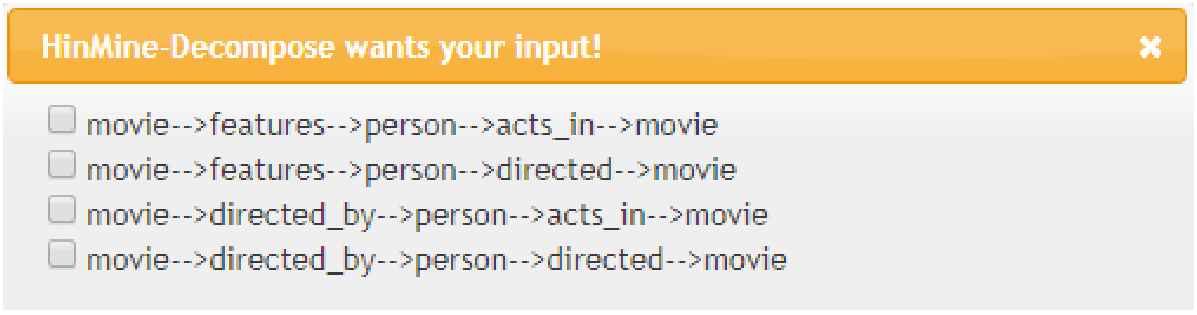
Figure 1: Overview of the HinMine methodology as a workflow in the Clowdflows platform.

HinMine Workflow in ClowdFlows

We implemented all the functions, used in our experiments with the HinMine methodology, in the ClowdFlows platform. The resulting workflow is shown in Figure 1. The workflow begins by loading a data set encoded as a `.gml` file. The GML (Graph Modeling Language) [2] is a text format that allows for easy representation of network data. For the HinMine methodology, the input requires that each node in the network is of a given type. In the online example, the methodology is run on a subset of the IMDB data set containing nodes of type `person` and nodes of type `movie`. One node type (the base node type for the HinMine methodology) must be labeled, and if more than one label is applicable for a node, the labels must be separated by a `------` separator. An example of a node looks as follows:

```
node [
  id 6336
  label "movie_303"
  labels "Action---Adventure---Western"
  type "movie"
  name "movie_the-quick-and-the-dead"
]
```

The methodology loads the GML file in the widget *Read GML from file*, where it identifies the base node type (the node type that is labeled) and training instances (all instances that are labeled). The loaded network is passed as the variable `net` to the *HinMine-Decompose* widget where network decomposition is applied. This is an interactive widget that, when first run, provides all possible decompositions of the input network. The widget discovers all possible decomposition paths and allows the user to choose which decompositions to perform. After performing the decompositions, the widget returns the decomposed network in the variable `net`. Figure 2 shows the possible decompositions of the online example. After decomposition, the methodology has two options:


HinMine-Decompose wants your input!

- [ ] movie-->features-->person-->acts_in-->movie
- [ ] movie-->features-->person-->directed-->movie
- [ ] movie-->directed_by-->person-->acts_in-->movie
- [ ] movie-->directed_by-->person-->directed-->movie

Figure 2: The decomposition selection in HinMine.

1. If we classify the data set with label propagation, we can use the *Apply Label Propagation* widget. This widget performs label propagation on the network and returns the results as a numpy [3] array (variable `res`).
2. If we classify the data using propositionalization, the *HinMine-Propositionalize*} widget performs the network propositionalization described in Section 3.1 of Deliverable 1.3b. This widget constructs the feature vectors for the labeled (training, variable `tra`) and unlabeled (test, variable `tes`) nodes separately. In this way, the classifier can be trained using the *Build MultiLabel Classifier* widget on the training set. In classifier construction using the *Build MultiLabel Classifier*, any learner capable of predicting labels on data sets containing numeric values can be used as the inuput variable `lea`. In the online example, we use the *k*-nearest neighbours classifier. Finally, the induced classifier is applied to the test set and a numpy array is returned as a result in the variable `res`.

Both options above result in the workflow returning a numpy array. The array's columns represent the labels of the data set and the rows represent the unlabeled nodes. Each row contains results of label propagation applied to the given unlabeled node. The result is a vector of values between 0 and 1, and the higher the value, the more likely it is that the label is applicable to the given node.

NetSDM Workflow in ClowdFlows

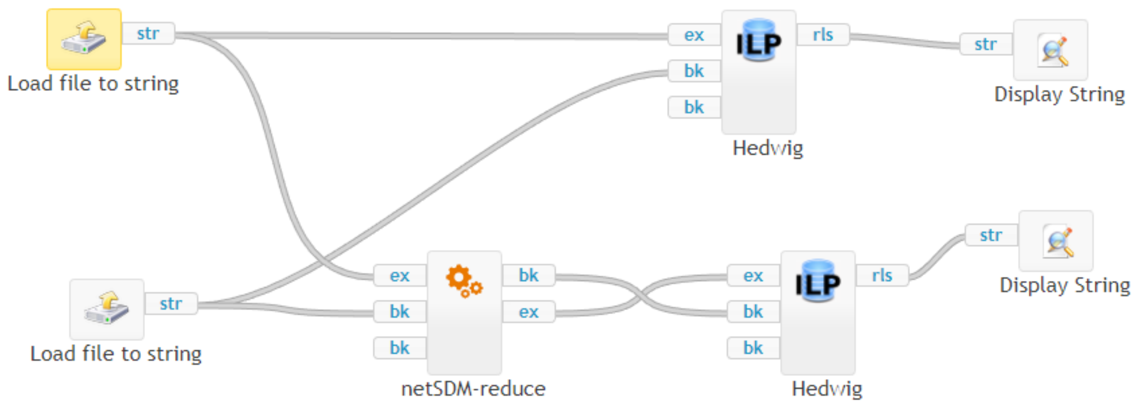


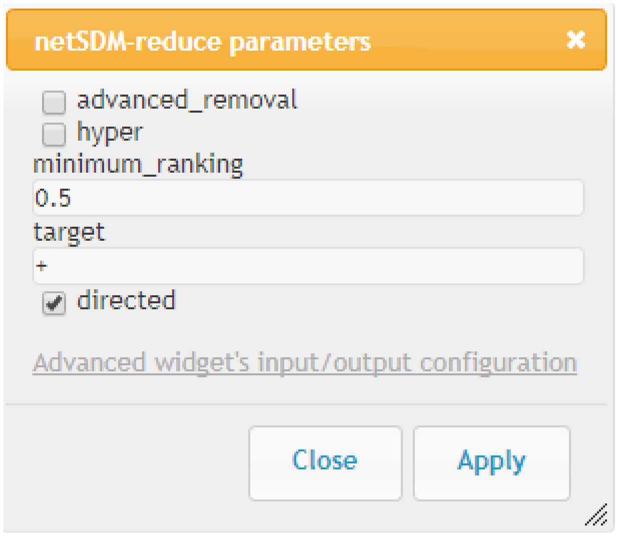
Figure 3: Overview of the NetSDM methodology as a workflow in the Clowdflows platform.

The workflow, implementing the NetSDM methodology which was described in more detail in Deliverable 1.3c is shown in Figure 3 and is also available online[2]. The workflow begins by loading the background knowledge (denoted as the input variable bk) and the set of examples (denoted as the variable ex). This step is the same as the first step of the Hedwig [4] methodology which is also available in the ClowdFlows platform[3]. The background knowledge file is then loaded into the *netSDM-reduce* widget which prunes the background knowledge network. Double-clicking on the widget allows the user to change the parameters of the NetSDM algrithm:

- the *advaced_removal* checkbox determines whether the algorithm will use the advanced or naive node removal method,
- checking the *hyper* checkbox causes the algorithm to construct a hypergraph out of the background ontologies instead of using the naive network conversion,
- the *directed* checkbox tells the algorithm whether to take directions of network edges into account when calculating network scores,
- \emph *minimum ranking* determinines how much of the background knowledge should remain in the pruned data set.

[2] http://clowdflows.org/workflow/11015/
[3] http://clowdflows.org/workflow/7031/

The widget returns two objects: the pruned background knowledge set `bk`) and the newly annotated set of examples `ex`. These two objects can be used to discover rules in the widget *Hedwig* that runs the Hedwig SDM algorithm [4].


netSDM-reduce parameters

☐ advanced_removal
☐ hyper
minimum_ranking
0.5
target
+
☑ directed

Advanced widget's input/output configuration

Figure 4: Selecting the parameters for the NetSDM widget.

References

[1] Kranjc, J., Podpečan, V., & Lavrač, N. (2012). Clowdflows: A Cloud Based Scientific Workflow Platform *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 816-819). Springer.
[2] Himsolt, M. (1997). GML: A Portable Graph File Format . *Universität Passau.*
[3] Walt, S. v. d., Colbert, S. C., & Varoquaux, G. (2011). The Numpy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering* , 13 (2), 22-30.
[4] Vavpetič, A. (2016). *Semantic Subgroup Discovery* (Doctoral dissertation, Jozef Stefan International Postgraduate School).
[5]