RelaViz: Graph Visualization of Learned Relations Between Entities

CPSC 533C 2011 Project Proposal – October 28, 2011
Joel Ferstay
joelaf@cs.ubc.ca

Description

The purpose of this visualization tool is to inspect predicted relations between entities produced by a machine-learning algorithm. This activity is necessary to determine whether relations match reality, and make intuitive sense.

Entity-relation-entity data usually come in the form of a triplet (e^l, r, e^r) where e^l stands for left entity, r stands for relation, and e^r stands for right entity. A particular instantiation of this general triplet could be (grass, grows_in, dirt). Some researchers in artificial intelligence have suggested that we might simulate human common sense by storing enough relational facts about the world, and generalizing from them appropriately to unobserved propositions [1]. To illustrate this process, suppose we are given propositions such as (mug, used_for, drinking), (mug, can_contain, coffee), (mug, can_contain, juice), (mug, can_contain, water), (cup, can_contain, juice), (cup, can_contain, water), (cup, can_contain, milk). From these propositions, one might predict the propositions (cup, used_for, drinking), (cup, can_contain, coffee), and (mug, can_contain, milk) [1]. Existing entity-relation-entity data needs to come from somewhere, however, and to produce high quality proposition predictions, machine learning algorithms typically require a lot of data. This is where accessible relational data on the web comes into play, along with Knowledge Bases (KB), and the need for structured embeddings of these KBs, to make their data accessible to machine-learning algorithms.

In order to gather, organize, and make deliberate use of massive amounts of information generated daily, special kinds of web-based relational database specifically designed for knowledge management, collection, and retrieval called Knowledge Bases have been built [2]. A large amount of data regarding general and specific knowledge are now available online: OpenCyc, WordNet, Freebase, DBpedia, etc. [2]. The problem for web-scale machine learning is that these KBs are highly structured and organized. Although this is a benefit for the original purpose a KB is intended for, it does mean that their knowledge is “locked-up” in a symbolic framework that is not flexible enough to be exported for use in machine-learning algorithms [2]. To solve this problem, Bordes et al describe and implement a statistical machine learning approach that learns to represent elements of any KB into a low (e.g. 50) dimensional vector space [2]. These embeddings are important for the production of relational learning data because they can then be used as input to a learning algorithm for learning relations of the type (e^l, r, e^r) [2]. This project will use new relation data learned that was learned from Wikipedia using the methods of Bordes et al.

So why is visualization necessary for this new, predicted entity-relation-entity data? One reason is there is currently no way of validating the new, predicted relations produced from learning algorithms automatically. Therefore, visualization can be used as a kind of verification tool for whether or not the relations between entities make sense. To illustrate why good visualization is needed, let us look at a current example for displaying new, learned relations: In the study by Borges et al, some new, learned relations are simply displayed in a table format for the reader to assess their plausibility, see Fig. 1 [2].

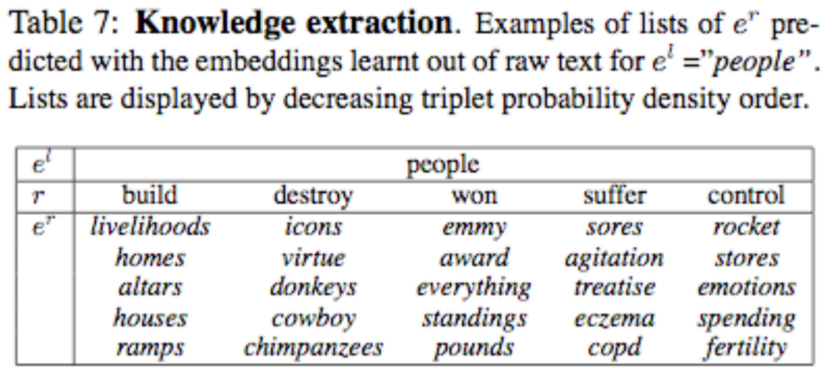


Table 7: **Knowledge extraction**. Examples of lists of e^r predicted with the embeddings learnt out of raw text for e^l ="*people*". Lists are displayed by decreasing triplet probability density order.

e^l	people				
r	build	destroy	won	suffer	control
e^r	*livelihoods*	*icons*	*emmy*	*sores*	*rocket*
	homes	*virtue*	*award*	*agitation*	*stores*
	altars	*donkeys*	*everything*	*treatise*	*emotions*
	houses	*cowboy*	*standings*	*eczema*	*spending*
	ramps	*chimpanzees*	*pounds*	*copd*	*fertility*

Figure 1. The display used in Borges et al for conveying their relational learning algorithm's effectiveness. Presenting the data in table format will not scale up to the visual inspection of many entity-relation-entity data and many relations between entities. *Note: Table 7 is taken from Borges et al* [2].

With no truly expressive tools for assessing a relational learning algorithm's performance, and the inability to automate validation of new, learned relations, it is clear there is a need for an effective visualization tool.

Personal Expertise

I have previous experience in designing, implementing, and testing machine learning algorithms, but not in the domain of relational learning. I have not seen any papers that approach the problem of validating a relational learning algorithm through visualization, and I feel that a visualization approach would be especially useful here.

For the visualization display, I do have experience in programming, graphics, and human-computer interface design, but I do not have experience with any of the visualization toolkits.

Proposed Solution

The proposed data abstraction for this relational learning data visualization is the node-link graph. Graphs are a natural choice for relational data to begin with, and have the flexibility to show many directed relationships between node encoding entities, if needed [3]. Adjacency Matrixes are another possibility, but I would argue they are not a sufficient solution for visualizing relational data, since it is difficult to tell what the semantics of the relation is in the Adjacency Matrix. Furthermore, it will be even more difficult to extend this approach if there are multiple, directed relations between entities. Given the node link graph abstraction for displaying relational data, some activities a relational learning algorithm designer would like to perform are:

1) Examining the identity of the relation between two entities: for instance, given that there exists a relation between entities cat and milk, what is it?
2) Examining the directionality of the relation between two entities: Given cat likes milk, it is not necessarily the case that milk likes cat.
3) Examining the probability that a relation is true: relational learning algorithms may produce relations with an associated probabilistic quantity of how likely the relation is.
4) The ability to differentiate between existing, known relations from the training data, and the relational learning algorithm's proposed relations: exploring the differences between the ground truth relations and the predicted relations may lead to further hypothesis generation regarding the algorithm's performance.

To solve 1), I propose having labels indicating the relation appear on the links between entities when the user hovers over them with a cursor.

To solve 2), I propose indicating the directionality of the relation with arrows. This will allow for unidirectional or bidirectional relations between entities to be visible.

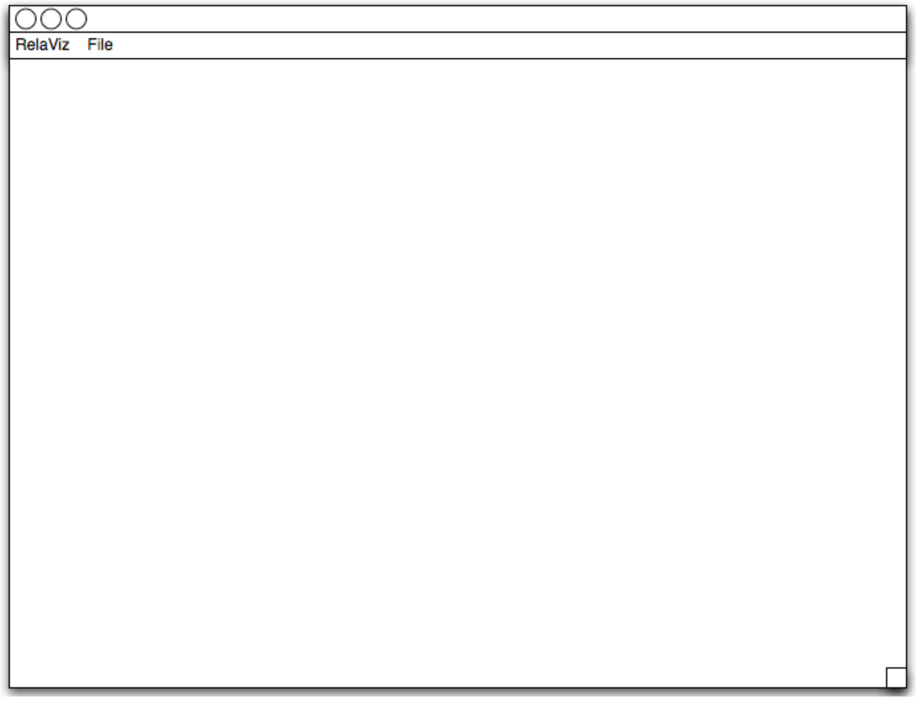
To solve 3), I propose having labels indicating the percentage probability estimate of how likely the relation is to appear on the link between entities. The thickness of the link and the curvature of the link will also encode how likely a relation is.

To solve 4), I propose allowing the user to highlight all ground truth relations with a button, and all learned relations with another button. This will allow the user to distinguish between the two, and perhaps see if there are any distinguishable patterns in the way the algorithm is learning from the ground truth relations.

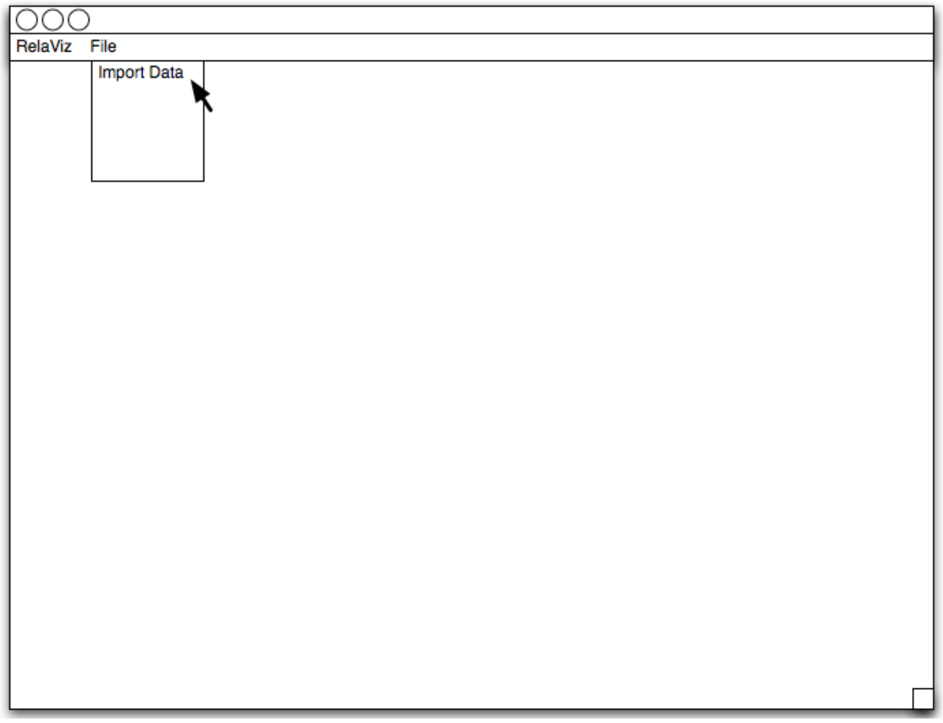
Scenario of Use

What follows is a description of some of the features the tool will provide for a user to explore whether or not their relational learning algorithm is producing relations between entities that make sense.

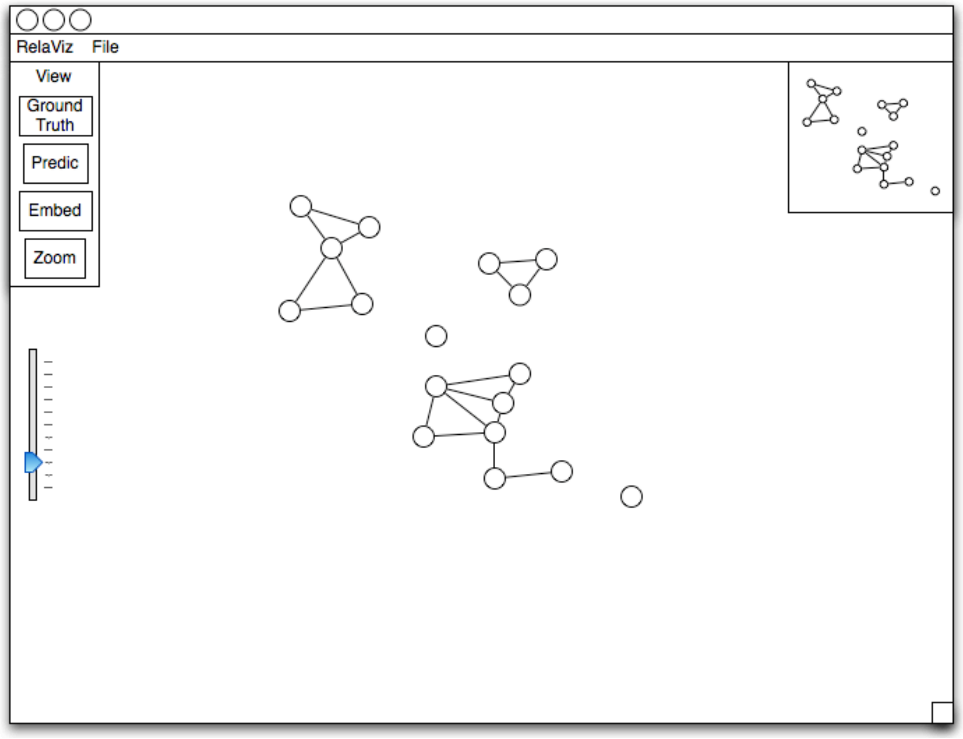
When the user initializes the tool, they will see the following window:



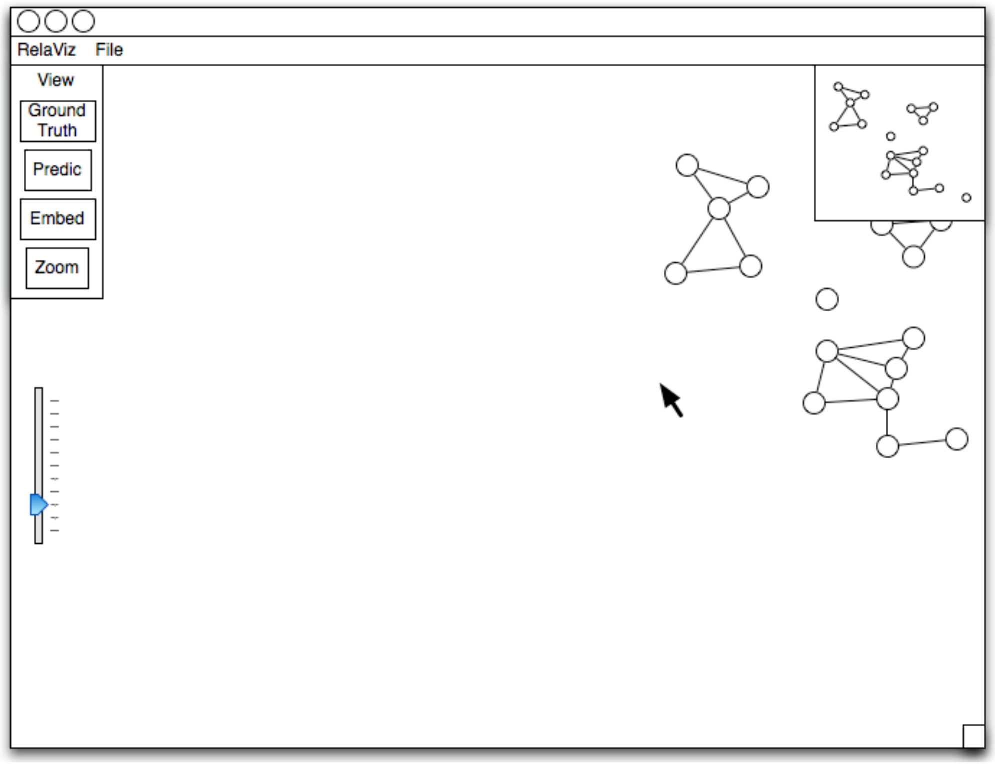
To load relational learning data, the user will select File -> Import Data:



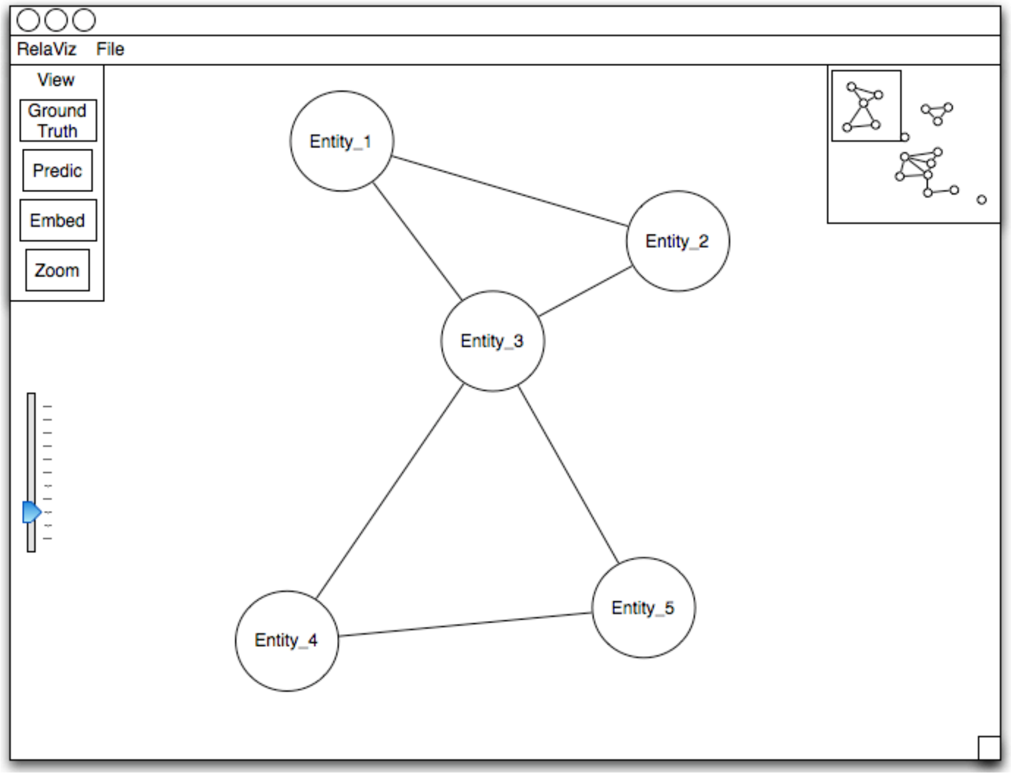
After selecting the appropriate file from the OS's native file browser, the graph will appear in the window, along with several options and a zoom slider controller to the left of the window. An overview of the graph in the top right-hand corner will also be visible:



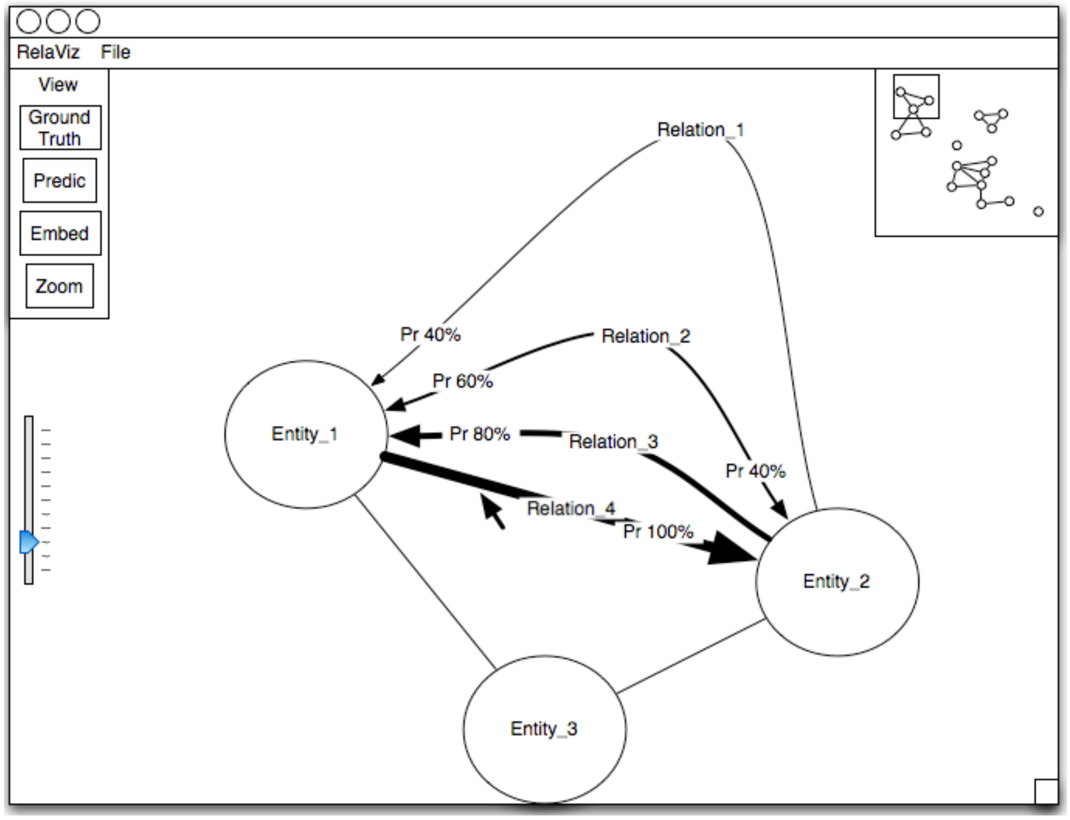
The user can translate the view of the graph by clicking and dragging on the whitespace:



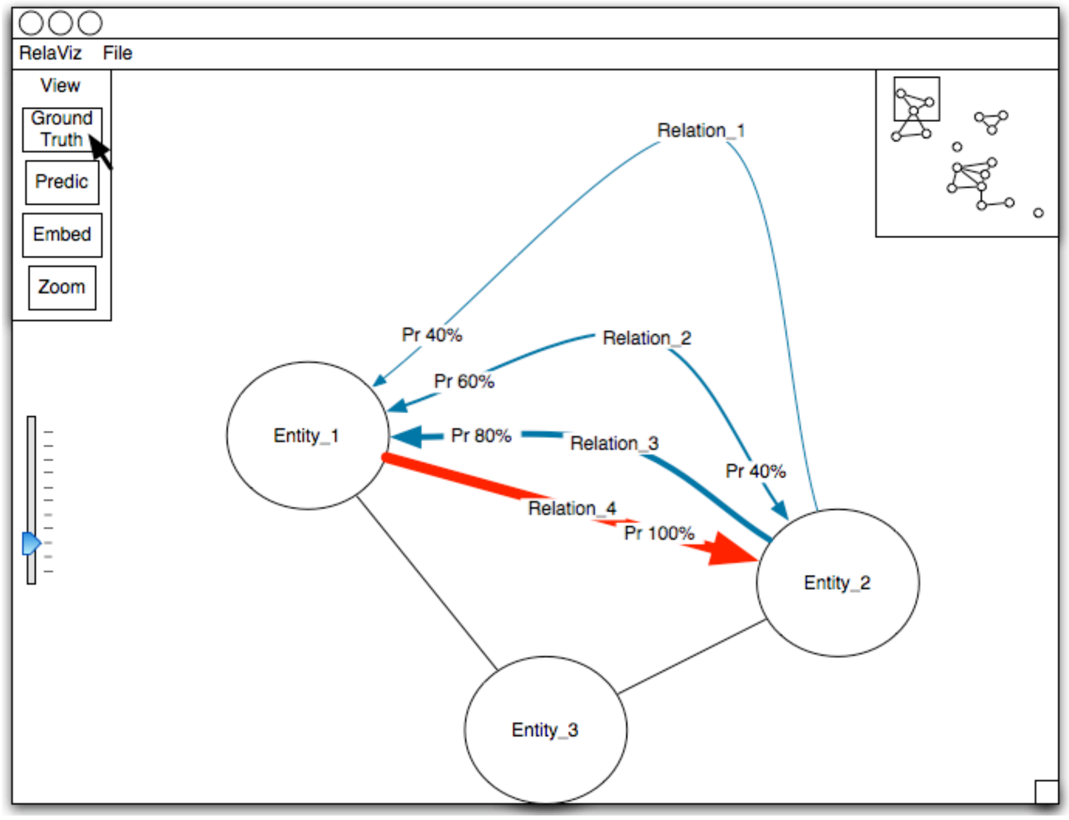
The slider on the far left allows the user to zoom in and out on the graph using the center of the window as the zoom axis. If the user would like to zoom in on a particular area of interest, they can select the "Zoom" button on the left of the display, and click on the area they would like to zoom in on:



Note that the overview window on the top right has updated to show a border indicating where the user is in the graph. At this level, Entity labels are now visible. To view the nature, directedness, and probability that a relation is correct, the user can hover their cursor over a link:



When the cursor is placed on the link, the link splits into multiple links indicating the various relations between the two entities. From the above display, the user can tell several things: the identity of individual relations, the directedness of a relation, and a measure of how likely a learned relation is. For instance, from the above diagram, the user can see that Relation_3 is directed from Entity_2 to Entitiy_1, and has a likelihood of about 80%. Higher probability links are thicker than lower probability links. Lower probability links are also more distant and curved. The user can also distinguish between which relational links were in the training set and are ground truth, and which relational links were learned, by selecting the "Ground Truth" button:



Selecting “Ground Truth” indicates the true, known relational links in red, and shows the relations learned by the algorithm in blue.

This completes the user scenario.

Proposed Implementation Approach

The relational learning visualization tool will be implemented using one of the Gephi, GraphViz, or Tulip visualization toolkits, on the Mac OS 10.5.8 [4,5,6]. One of Gephi, GraphViz, or Tulip will be chosen based on a balance of whether or not they can express the system described in the Proposed Solution section of this document, and their learning curve. Strengths and weaknesses of each toolkit based on preliminary experimentation are mentioned in Table 1.

Table 1: Strengths and weaknesses of the Gephi, GraphViz, and Tulip visualization toolkits based on preliminary experimentation.

Toolkit Identity	**Strengths**	**Weaknesses**
Gephi	- Well supported - Expressive	- Moderate learning curve - Not as expressive as Tulip
GraphViz	- Probably easiest to learn of the three toolkits - Well supported	- Moderate learning curve - Not as expressive as Gephi or Tulip
Tulip	- Probably the most expressive of the three toolkits	- Probably the steepest learning curve of the three toolkits

Based on preliminary testing, it is likely that Gephi will be used to implement this project; however, this will be fully confirmed after several iterations of the project design have been completed, and the toolkit requirements are completely nailed down.

Note that this project is being performed in parallel with a project in CPSC 540, which involves implementing the algorithm for learning structured embeddings and predicting new relations between entities proposed by Borges et al [2]. A strong dependency is on whether and when the entity-relation-entity data will be available for processing by the visualization tool. In the meantime, either synthetic entity-relation-entity data will be produced to feed into the system during development, or existing entity-relation-entity data will be acquired from the web. This dependency is noted in the milestone schedule. Table 2 outlines the project's milestones.

Table 2: Milestone Schedule.

Week Number and Date	**Activities**
Week 1 (October 31, 2011)	- Refinement of graph visualization design using paper mock-ups for rapid prototyping. - Familiarization with the strengths and weaknesses of the Gephi, GraphViz, and Tulip toolkits. - Selection of toolkit and exploration of implementing small toy examples using the toolkit, through tutorials.

Week 2 (November 7, 2011)	- Validation of prototype design through user testing. - Make changes to design based on tests and finalize design. - Outline class design for implementing the visualization in a modular fashion. - Decide how to format data to leverage the toolkit's existing data import capabilities. - Prepare project update materials.
Week 3 (November, 14 2011)	- Revisions based on feedback from update. - Assess state of CPSC 540 project, and either produce synthetic relational data or acquire it somehow, or use actual data if it is ready. - Implement displaying the entity nodes. - Implement zooming. - Implement displaying the relation link between each entity if at least one exists. - Start working on the relation link splitting to show multiple relations.
Week 4 (November, 21 2011)	- Finish work on the relation link splitting to show multiple relations. - Implement ground truth versus prediction relational link visualization. - Implement "View" button window. - Implement zoom slider.
Week 5 (November, 28 2011)	- Tie up any loose ends in missing functionality. - Test and optimize. - Start working on presentation and report.
Week 6 (December, 5 2011)	- Continue working on presentation and report. - Finish presentation.
Week 7 (December, 12 2011)	- Finish report.

Previous and Related Work

At present, I could not find any visualization tools that deal explicitly with assessing a relational learning algorithm's performance. There are, however, other visualization tools that deal with the problem of debugging an algorithm by visualizing its output in other domains; I will draw inspiration for my design from these projects.

One such visualization tool is Constellation [7]. Constellation uses a graph layout algorithm to allow users to examine a large semantic network [7]. Techniques used in this paper could be quite useful for this project, since both projects are graph based.

Another visualization tool used to assess algorithm performance is MizBee [8]. MizBee is a visualization tool for browsing synteny in comparative genomics [8]. This tool can help a designer assess their algorithm's performance by visualizing its output.

The relational graph used in this project will be quite large. There are many studies that could help guide the presentation of this graph, including van Ham and Perer's work on supporting large graph exploration, and Wattenberg's study on the visual exploration of large multivariate graphs [9,10].

References

[1] Ilya Sutskever, Ruslan Salakhutdinov, and Josh B. Tenenbaum. Modelling relational data using Bayesian clustered tensor factorization. In Advances in Neural Information Processing Systems (NIPS), 2009.

[2] A. Bordes, J. Weston, R. Collobert and Y. Bengio. Learning Structured Embeddings of Knowledge Bases. AAAI 2011.

[3] Ivan Herman, Guy Melancon, M. Scott Marshall. Graph Visualisation in Information Visualisation: a Survey. IEEE Transactions on Visualization and Computer Graphics, 6(1), pp. 24-44, 2000

[4] Gephi, an open source graph visualization and manipulation software. [http://gephi.org/]

[5] Graphviz – Graph Visualization Software [http://www.graphviz.org/]

[6] Tulip – Data Visualization Software [http://tulip.labri.fr/TulipDrupal/]

[7] Tamara Munzner, Francois Guimbretiere, and George Robertson. Constellation: A Visualization Tool For Linguistic Queries from MindNet. Tamara Munzner, Francois Guimbretiere, and George Robertson. Proc. InfoVis 1999, p 132-135.

[8] Miriah Meyer, Tamara Munzner, and Hanspeter Pfister. MizBee: A Multiscale Synteny Browser. IEEE Trans. Visualization and Computer Graphics 15(6):897-904 (Proc. InfoVis 09), 2009.

[9] Frank van Ham and Adam Perer. "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. IEEE Trans. Vis. Comput. Graph. 15(6): 953-960 (2009)

[10] Martin Wattenberg. Visual Exploration of Multivariate Graphs. CHI 2006.