Multilanguage Dynamic Wordnet: Timeflow Hydra

Anonymous ACL submission

Abstract

Hydra is a Wordnet management system that integrates synsets from multiple languages into a shared relational structure, modeled as a Kripke frame. It features an intuitive graphical user interface (GUI) for searching, editing, and aligning language objects. Data retrieval is facilitated by a modal logic query language, enabling flexible interaction with the multilingual wordnets.

The original Hydra system, like other existing Wordnet systems, captures only the current state of the Wordnet, leaving open crucial questions about how Wordnet data, structure, and consistency evolve over time. To address this limitation, Time Flow Hydra introduces a **dynamic Wordnet model** with discrete time embedding. This model not only stores all historical states of Wordnet objects but also allows simultaneous access to them.

Time Flow Hydra manages data from multiple languages within a unified relational structure and enables users to query the data at any point in time. Additionally, it provides fine-grained access control through a robust permission framework, ensuring that users can manage and interact with the data securely.

With Time Flow Hydra, users can track changes over time and analyze the effects of data evolution, both desired and undesired. For example, users can query which synsets had two hyponyms 10 days ago and three hyponyms 5 days later.

1 Introduction

Wordnets across the world are continuously evolving and expanding, with new languages and lexical databases being developed regularly. Over the past few decades, various tools for managing and visualizing these databases have been created. One of these tools is Hydra, which stood out for its modal logic query language tailored to Wordnet management.

However, one might ask: What would the ideal Wordnet management system look like? In our vision of such a system, the "nirvana" would:

- Support multiple languages,
- Allow concurrent user access with fine-grained permissions,
- Store every version of every object in the database, providing access to it in every single moment,
- Offer a powerful query language,
- Provide an intuitive graphical user interface (GUI), and
- Have a robust model that cannot be compromised by misuse or by malicious users.

Several years ago, we introduced a new web-based system called Hydra for Web that brought us closer to this vision. It featured a simple, fast, and mobile-first web interface, allowing multiple users to visualize and edit Wordnets across several languages simultaneously. One of its standout features is the ability to clone or replicate data from other languages, which accelerates the creation of new synsets and supports linguistic comparisons between languages.

One of the key challenges in the Wordnet community has been the alignment of databases developed for different languages. Hydra addresses this by enabling teams working on various languages to collaborate concurrently within a unified environment, significantly streamlining the overall Wordnet development process.

In this paper, we present a new dynamic model for Hydra that not only ensures data integrity but also captures every intermediate stage of Wordnet development. This model helps detect data and structural inconsistencies in real time, saving valuable human resources. With access to data across

all stages of its evolution, users are also provided with a powerful modal query language enhanced with new temporal modalities. Additionally, the system prevents data loss and ensures protection even against malicious user behavior.

The new Hydra system also offers a vastly improved database model, with queries processed much faster than in the previous versions—some by orders of magnitude.

We begin with a traditional view of Wordnet, framed as a relational data structure. Next, we introduce Hydra, outlining its features and limitations. Finally, we present the new dynamic model of Wordnet and its implementation in the Time Flow Hydra system.

2 Wordnet for Many Languages

Wordnet development began with the creation of the English WordNet at Princeton (Miller et al., 1990), and the concept has since been applied to more than 40 languages. Most of these Wordnets are developed, or are still under development, using a so-called synchronous model, where the hypernymy structure mirrors that of the Princeton WordNet. Using common identifiers or alignment mappings, synsets encoding similar concepts across different languages are linked. This relation, or identifier, is known as the *Interlingual Index* (ILI).

These large multilingual Wordnet databases, built using a relational model, have proven to be highly valuable for numerous linguistic tasks. However, they also encounter several significant challenges. Since different teams develop these Wordnets using different software platforms, file formats, and database systems, the databases are stored and maintained separately. The alignment process (i.e., maintaining the ILI) is typically performed periodically, often focusing on specific language pairs and particular versions of these Wordnet databases.

The development of a *Collaborative Interlingual Index* aimed to reduce the sparseness of ILI mappings, but it has achieved limited success. Among the key issues that persist are the separation of language databases and the inconsistent synset identifiers within the central Princeton WordNet database.

3 Static Model for Wordnet

At any given moment in time, a Wordnet consists of a collection of synonymous sets, called **synsets**, which represent concepts in the language. These synsets are interconnected through semantic relations such as hyperonymy (is-a) and meronymy (part-of). Within each synset, additional associated data can be found, such as the part of speech and the specific words or phrases that comprise the synset. A word belonging to a particular synset is called a **literal**. It is important to note that while a single word can appear in multiple synsets, each literal is unique and can be thought of as a $\langle \textit{synset}, \textit{word/compound} \rangle$ pair. Literals themselves are connected by lexical relations such as antonymy.

In addition to these structural elements, Wordnet databases often include text data such as example sentences, notes on particular synsets or literals, and other descriptive annotations. This text data is referred to as **notes**, and can be conceptualized as $\langle \textit{synset/literal}, \textit{text} \rangle$ pairs.

4 Wordnet as a Kripke Frame

We can model a Wordnet using three types of objects: **Synset**, **Literal**, and **Note**. To capture the relationships between these objects, we define specific binary relations. For example, the *Literal relation* links a literal to its parent synset, and the *Usage relation* connects a note to the synset for which it serves as a usage example. In addition to these, we have the usual semantic relations such as hyperonymy, meronymy, antonymy, and others.

From this structure, we derive a Kripke frame $\langle W, R \rangle$, where W represents the universe of the three sorts (Synset, Literal, and Note), and R is the set of binary relations between these objects. This Kripke frame naturally lends itself to a modal logic language, which we will present in subsequent sections.

Each object in the Wordnet can be viewed as a feature structure, with a fixed set of attributes depending on its type. Synsets, for instance, are characterized by features such as *pos* (part of speech), *lang* (language code), and *ili* (an identifier shared across languages for the same concept). Literals have attributes like *word* and *lemma*, while Notes possess the *note* feature (the text they represent). All three types of objects share common attributes, such as *id* (a unique identifier in the static model) and *userId* (the identifier of the user who created the object).

5 Hydra

The Wordnet database management system **Hydra** (Rizov, 2008) was developed in 2006 to address the challenges encountered during the creation of BulNet, the Bulgarian Wordnet. Hydra introduced a novel model for Wordnet based on a Kripke frame, where linguistic data from various languages—including Princeton WordNet and BulNet—are integrated into a single relational structure.

Unlike earlier tools used to manage the BulNet (such as VisDic (HORÁK and SMRŽ, 2003)), which managed Wordnet data using XML files, Hydra adopted a relational database management system (RDBMS) to store the data. Initially, early Wordnets even did not utilize XML, which later became the standard format for several years. However, XML proved insufficient for efficiently managing the complexity of Wordnet data, and transitioning to a relational database represented a significant improvement, offering greater scalability and performance.

Several years later, Hydra evolved into a modern single-page application (SPA) (Rizov and Dimitrova, 2016), providing a more user-friendly web interface. This version of the system is currently in production `http://dcl.bas.bg/bulnet` and supports Wordnets for 22 different languages.

Data searches within Hydra are conducted using a modal logic query language, enabling sophisticated and flexible interaction with the multilingual Wordnet databases.

6 Dynamic Model for Wordnet

A static Wordnet database provides only an incomplete, instantaneous snapshot of a language. There are synsets that remain undefined, some relationships that are only partially instantiated, and the Wordnet databases for different languages are often at various stages of development. Additionally, certain languages may have specific concepts that are unique to them. Over time, both the language and its Wordnet representation naturally evolve, which leads to different states of Wordnet at different points in time. This evolution raises important questions about the consistency of the data and its structure, as defined by the binary relations across time.

If we consider snapshots of a static Wordnet at various moments, we obtain a collection of Kripke frames. By assigning each object in each frame a timestamp corresponding to its frame, we can take the union of this set of disjoint frames. The result is a single Kripke frame that captures all the manifestations of all objects in Wordnet across time. Formally, this is expressed as:

$$\{\langle W_t, R_t \rangle\}_{t \in T}$$

where T represents the time model. In our implementation, we adopt a discrete time model, with the assumption that at most one object or relational pair changes at any given moment. This assumption is strictly enforced, and we strengthen it by ensuring that every change in the data results in the creation of a new moment in this discrete time model. Thus, each point in time corresponds to a moment where a single object or relational instance is created, modified, or deleted.

In terms of physical time, the state of an object at any given point is equivalent to its state at the nearest preceding moment in model time. At any fixed moment in model time, we can retrieve all objects from that moment and the preceding ones, taking only the most recent (or "last version") state of each object. This yields the static Kripke frame for that particular moment in time.

The collection of all versions of all objects over time is what we refer to as the **Dynamic Wordnet Model**.

7 Query Language

The process of constructing and editing a Wordnet raises important questions about how the data and its structure evolve over time. Users may be interested in tracking the presence of specific properties or relations within the data, and detecting inconsistencies when they arise. Ideally, these issues can be addressed without needing to revert to a prior backup state that may include many changes made by multiple users.

For example, consider a scenario where object 1 has changed in the past, objects 2 and 3 have since been modified or corrected, and object 4 has been newly created. If a problem or inconsistency is later detected with object 1 and its associated relations, the dynamic model allows users to trace the history of changes and pinpoint exactly when and why the issue occurred. This capability eliminates the need to undo unrelated changes made by other users.

This is achieved through the use of a modal logic query language, initially developed for the early versions of Hydra and subsequently enhanced with

temporal modalities. The system operates using a technique known as *model checking*, where for a given modal formula, the system returns the set of objects for which the formula holds true. This enables users to efficiently investigate and correct inconsistencies in the Wordnet data across its entire evolution.

7.1 Dynamic Wordnet Language

We define the syntax and corresponding semantics of the modal formulas inductively.

7.1.1 Syntax

The language consists of the following components:

- $\mathbf{N}$: A set of individual constants (nominals). In the system, we represent them using decimal numbers.
- $\mathbf{O}$: A set of constants representing the features of objects and their values, following the schema $type('value')$. For example, *pos('n')* is a constant indicating that the part of speech is a noun.
- $\mathbf{R}$: A set of relation symbols that define the relationships between objects.
- $\mathbf{TM}$: A set of temporal modifiers.

There are four types of temporal modifiers, each representing different aspects of time:

- **Fixed timestamp** (real-world time): e.g., *t159737980000*
- **Fixed operation moment** (model time): e.g., *o1235*
- **Relative future**: e.g., *f5* (indicating 5 days into the future)
- **Relative past**: e.g., *p3* (indicating 3 days into the past)

Atomic Formulae: $\mathbf{AtomicFor}$

- $\bot$
- $\top$
- $\mathbf{N} \subseteq \mathbf{AtomicFor}$
- $\mathbf{O} \subseteq \mathbf{AtomicFor}$

Formulae: $\mathbf{For}$

- $\mathbf{AtomicFor} \subseteq \mathbf{For}$.

Let q and r be fomulae (queries), R $\in \mathbf{R}$, $\mathbf{t} \in \mathbf{TM}$, then the following are formulae:

- !q
- q & r
- q | r
- q => r
- q <=> r
- <R>q
- [R]q
- $\ll \mathbf{t} \gg$q

We also use some relation modifiers, namely:

- ~R - the reverse relation of R
- R+ - the transitive closure of R
- R* - the reflexive and transitive closure of R

7.2 Semantics

The semantics of the Dynamic Wordnet Language is defined as follows:

- A Time structure is $\langle T, t_c, < \rangle$, where $T \neq \phi$ is a finite set, $<$ is a linear ordering, t_c is $max_< T$ (the current moment)
- A Model of time is $\langle \langle T, t_c, < \rangle, m \rangle$, where $m : \mathbf{TM} \times T \to T$
- A static model (Kripke frame for a given moment t) is $\mathfrak{M}_t = \langle W_t, \mathcal{R}_t, V \rangle$, where $W_t \neq \phi$, $\mathcal{R}_t : \mathbf{R} \to \mathcal{P}(\mathbf{W_t} \times \mathbf{W_t})$, $V : \mathbf{N} \cup \mathbf{O} \to \mathcal{P}(\mathcal{W})$ and for $c \in \mathbf{N}$ $V(c)$ has at most 1 element.
- A dynamic model is $\mathcal{D} = \langle \{\mathfrak{M}_t\}_{t \in T}, \mathcal{T} \rangle$, where $\mathcal{T} = \langle \langle T, t_c, < \rangle, m \rangle$ is a model of time.

We define the **truth** of a formula in a object x in the Dynamic model $\mathcal{D}$ by induction on the formula construction:

- $\mathcal{D}, t, x \nVdash \bot$
- $\mathcal{D}, t, x \Vdash \top$

- $\mathcal{D}, t, x \Vdash c$ for $c \in \mathbf{N} \cup \mathbf{O}$ iff $x \in V_t(c)$

  Each object in the database has an identifier and it is a nominal (constant) in our language. A synset identifier is encoded so as to be portable and it depends only on ili (identifier comming from PWN), pos (part of speech code) and the language (code) of the synset. In the implemented system this semantic more concretely is:

  - $\mathcal{D}, t, x \Vdash$ \$s iff x is a Synset
  - $\mathcal{D}, t, x \Vdash$ \$l iff x is a Literal
  - $\mathcal{D}, t, x \Vdash$ \$n iff x is a Note
  - $\mathcal{D}, t, x \Vdash$ type('value') iff x.type = value (for instance x.pos=n, so x is a noun synset)

- $\mathcal{D}, t, x \Vdash$!q iff $\mathcal{D}, t, x \nVdash$ q

- $\mathcal{D}, t, x \Vdash$ q & r iff $\mathcal{D}, t, x \Vdash$ q and $\mathcal{D}, t, x \Vdash$ r

- $\mathcal{D}, t, x \Vdash$ <R>q iff

  $\exists y(x \mathcal{R}_t(R) y \& \mathcal{D}, t, y \Vdash \text{q})$

- $\mathcal{D}, t, x \Vdash \ll \mathbf{t} \gg$ q iff $\mathcal{D}, m(\mathbf{t}, t), x \Vdash$ q

- We say that a formula is true in dynamic model at point x, denoted $\mathcal{D}, x \models$ q iff $\mathcal{D}, t_c, x \Vdash$ q

  For the sake of an example we'll use concrete natural numbers in the following:

- $\mathcal{D}, t, x \Vdash \ll$o1235$\gg$q iff $\mathcal{D}, t_0, x \Vdash$ q where $m(o1235, t) = t_0$.

  As mentioned before, every data modification creates a model time moment which is refered as an operation id and t_0.id=1235.

- $\mathcal{D}, t, x \Vdash \ll$t159737980000$\gg$q iff $\mathcal{D}, t_0, x \Vdash$ where t_0 is the nearest previous model moment to this timestamp

- $\mathcal{D}, t, x \Vdash \ll$p3$\gg$q iff $\mathcal{D}, t_0, x \Vdash$ where t_0 is the nearest previous model moment to the moment $t - 3$ days

- $\mathcal{D}, t, x \Vdash \ll$f5$\gg$q iff $\mathcal{D}, t_0, x \Vdash$ where t_0 is the nearest previous model moment to the moment $t + 5$ days

7.3 Query Answering

A formula in the defined modal language is a query in Hydra. The result of such a query q at a given time moment t is the set of unique objects with respect to their IDs, such that their time is the most recent one which is prior to the time t. By default, the time t is the current moment t_c when the query is executed. This moment t can be fixed to an arbitrary time via the GUI. We call this feature *Time Machine*.

8 Example queries

Let's see some useful queries.

- Find the noun synsets that are on top of hyperonymy hierarchy in English:

  ```
  pos('n') & [hypernym]⊥ & lang('en')
  ```

- Find the synsets that are exactly two levels below the top in the hyperonymy hierarchy:

  ```
  [hypernym][hypernym][hypernym] ⊥ & <hypernym><hypernym>⊤
  ```

- Find inconsistency between Bulgarian and English:

  ```
  <ili>(lang('en') & pos('n') & [hypernym][hypernym]⊥ & <hypernym>⊤)
  & lang('bg') & [hypernym]⊥
  ```

- Find the literals that before 3 days were presenting the word 'test' and 2 days later are not:

  ```
  <p3>(word('test') & !<f2>word('test'))
  ```

9 Graphical User Interface and Implementation

Hydra is implemented in JavaScript, consisting of a modern SPA (single-page application) web app and a REST API service. The WordNet data is stored in a Postgres database, and the queries are translated to SQL. There is a preprocessing step where each subformula's model time is determined.

While most of the relations are stored in the database, some are implemented directly in the translations of the formula—such as the universal relation U, and the transitive and reflexive closures of other relations. An important feature is that no data can be lost during WordNet development, even in the presence of a hostile user. Every change in the data creates a new copy of the object or relational instance touched, with the same ID but the

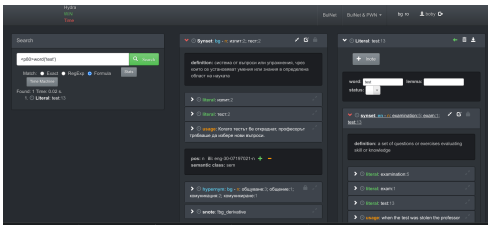

Figure 1: Hydra

new data. For instance, when an object is deleted, a new record for it is created (operation record) where it is marked as 'deleted'. At any point, the user can see the data as it existed in the past using the so-called *Time Machine*. The user opens a dialog, selects a moment in the past, and views the data as it was at that time.

The GUI is simple yet powerful. It has a Search Panel with three modes for searching—using a word, regular expression, or a formula. The first two options find synsets that have literals matching the provided word or regular expression. The latter uses the defined modal query language and is much more powerful.

There are two modes to visualize the data found. The first one (called 'Single') visualizes the object selected from the list of found items. The second one aligns a pair of language WordNets present in the system. When the user selects an item from the list, the corresponding copies in the aligned languages are visualized. This way, a user can search for a Spanish word and see the aligned corresponding entries in French and English, for example.

The visualization of an object consists of its static data (like POS and language for the synsets) and the relations—all the connected objects by all the relations. The view is recursive, and the data for related objects is visualized on demand.

Hydra is also a fully-fledged editor for WordNet data. A user with sufficient rights can put an object into edit mode, where the representational controls are replaced with edit controls, allowing the user to edit and save the data. Relational pairs are added using a wizard. These changes are sent to other users via notifications.

10 Tracking User Edits

User actions are tracked out of the box! The nature of the **dynamic model** automatically maintains a history of all editing operations. Unlike version control systems, previous versions do not need to be checked out—they are just available.

11 Real-time Updates

Edits to the Wordnet data are instantly propagated to all users via notifications, enabling smoother and more efficient collaboration. The system implements a strategy to prevent data loss during simultaneous updates to the same data. Conflicting changes are detected and managed to ensure consistency and integrity of the Wordnet data.

12 Conclusion and Future Work

The primary achievement of our work is the enhancement of Hydra's capabilities through the incorporation of time operators. Among these enhancements, the most valuable feature is the ability to easily identify and repair issues within the Wordnet. This functionality not only facilitates the correction of errors but also aids in understanding the underlying causes and identifying the users responsible for these inconsistencies.

However, one limitation of the current system is that its effective use necessitates a certain level of proficiency in logical languages. To address this challenge, we are developing a graphical user interface (GUI) assistant designed to assist linguists by providing predefined queries and schema queries. While this will streamline many tasks, it is important to note that some degree of expertise will still be required for more complex queries.

13 Deployment and Access

The **Time Flow Hydra** system, the latest and most advanced version of Hydra, now implements all of the features and concepts discussed. It is fully deployed and accessible for use. Users can explore its functionalities at the following link:

`https://hydra.fmi.uni-sofia.bg/`

References

Aleš HORÁK and Pavel SMRŽ. 2003. Visdic - wordnet browsing and editing tool. In *Proceedings of the Second International WordNet Conference - GWC 2004*, pages 136–141. Brno,Czech Republic: Masaryk University.

George A. Miller, Richard Beckwith, Christiane. Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to Wordnet: an on-line lexical database. *International journal of lexicography*, 3(4):235–244.

Borislav Rizov. 2008. Hydra: a modal logic tool for wordnet development, validation and exploration. In

Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, Marrakech, Morocco.

Borislav Rizov and Tsvetana Dimitrova. 2016. Hydra for web: A browser for easy access to wordnets. In *Proceedings of the 8th Global WordNet Conference (GWC)*, pages 342–346, Bucharest, Romania. Global Wordnet Association.