Multilanguage Dynamic Wordnet: Timeflow Hydra

2025

Outline

Wordnet

- Relational model of the language
- language concepts – synonymous sets
- 20 semantic and lexical binary relations
  - super-subordinate relation hyperonymy (AKA is-a)

Wordnet

- Relational model of the language
- language concepts – synonymous sets
- 20 semantic and lexical binary relations
    - super-subordinate relation hyperonymy (AKA is-a)

Wordnet

- Relational model of the language
- language concepts – synonymous sets
- 20 semantic and lexical binary relations
  - super-subordinate relation hyperonymy (AKA is-a)

Wordnet

- Relational model of the language
- language concepts – synonymous sets
- 20 semantic and lexical binary relations
  - super-subordinate relation hyperonymy (AKA is-a)

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Wordnet for many languages

- English in Princeton
- $>$ 40 languages
- Developed or are still in development using the so called **synchronous model** - hyperonymy structure follows this of the Princeton WordNet
- ILI - Common identifier to align synsets (en - bg)

Problems:

- Developed by different teams using different software platforms, file formats, databases, etc.
- Stored and maintened separately
- The alignment (ILI manintanace) is made periodically usually for particular language pairs and particular version of these wordnet databases
- Collaborative Interlingual index (**CILI**) was developed to help reduce the sparse ILI mapping problem, but it did not succeed much

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound \rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound\rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound \rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound \rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound\rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound\rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Static model for wordnet

In a fixed moment of time:

- Family of synonymous sets (**synsets**)
- Semantic relations (hyperonymy, meronymy)
- Associated to them we have data like **POS**
- A word in a synset - **Literal** - $\langle synset, word/compound \rangle$
- Lexical relations
- Text data - **notes** - usage examples, synset or literal features like verb type, etc.

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**

We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$

W - Synset, Literal and Note objects

R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R\rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R \rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Wordnet as a Kripke frame

3 types of objects - **Synset**, **Literal**, **Note**
We define special binary relations to encode the relationships between them.

- **Literal** relation connects a particular **literal** to its **parent synset**
- **Usage** relation connects a Usage example (**Note** object) with its **parent synset**
- **ILI** relates Synset in different languages representing the same notion.
- etc.

$\langle W, R\rangle$
W - Synset, Literal and Note objects
R - A set of binary relations

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

Hydra

Wordnet database management system

- First version - 2006
- Wordnet as a Kripke frame
- **Many** languages in a **single** database
- Second verion - Web SPA (Hydra for Web)
  Previous, still in production, system
  `http://dcl.bas.bg/bulnet` with **22** languages
- Searching with **Modal logic query language**

The Vision

- Support multiple languages
- Concurrent user access with fine-grained permissions
- Store **every** version of **every** object in the database
- Powerful query language
- Intuitive GUI
- Robust model resistant to misuse

The Vision

- Support multiple languages
- Concurrent user access with fine-grained permissions
- Store **every** version of **every** object in the database
- Powerful query language
- Intuitive GUI
- Robust model resistant to misuse

The Vision

- Support multiple languages
- Concurrent user access with fine-grained permissions
- Store **every** version of **every** object in the database
- Powerful query language
- Intuitive GUI
- Robust model resistant to misuse

The Vision

- Support multiple languages
- Concurrent user access with fine-grained permissions
- Store **every** version of **every** object in the database
- Powerful query language
- Intuitive GUI
- Robust model resistant to misuse

The Vision

- Support multiple languages
- Concurrent user access with fine-grained permissions
- Store **every** version of **every** object in the database
- Powerful query language
- Intuitive GUI
- Robust model resistant to misuse

The Vision

- Support multiple languages
- Concurrent user access with fine-grained permissions
- Store **every** version of **every** object in the database
- Powerful query language
- Intuitive GUI
- Robust model resistant to misuse

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Dynamic model for wordnet

A static wordnet database is an incomplete instantaneous description of the language. Over time, both the language and its wordnet representation change and evolve. If we take the snapshots of wordnet in the static model, we get a set of Kripke frames. Let's take the union of the resulting set of disjoint frames.

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

Dynamic wordnet model:

- Discrete time model
- **Only one** instance of object or relation can be changed in a single time moment
- For a moment we have an instance of the static model - with the most recent verion of the objects and relations (nearest previous version).
- The collection of all the static Kripke frames we call **Dynamic model**

Query language

The construction of wordnet and its editing opens questions about the evolution of the data and the structure over time. We would like to easily detect a problem, when it occured and who did it. We want it **without** reverting the data to previous state. We accomplish this and much more. We use model checking for searching.

Query language

The construction of wordnet and its editing opens questions about the evolution of the data and the structure over time. We would like to easily detect a problem, when it occured and who did it.

We want it **without** reverting the data to previous state.

We accomplish this and much more.

We use model checking for searching.

Query language

The construction of wordnet and its editing opens questions about the evolution of the data and the structure over time. We would like to easily detect a problem, when it occured and who did it.
We want it **without** reverting the data to previous state.
We accomplish this and much more.
We use model checking for searching.

Query language

The construction of wordnet and its editing opens questions about the evolution of the data and the structure over time. We would like to easily detect a problem, when it occured and who did it.
We want it **without** reverting the data to previous state.
We accomplish this and much more.
We use model checking for searching.

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema $type('value')$. For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t1597379800000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema $type('value')$. For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t15973798000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema *type*('*value*'). For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t1597379800000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema *type*(*′value′*). For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t159737980000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema *type*('*value*'). For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t159737980000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema $type('value')$. For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t159737980000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema $type('value')$. For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t159737980000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema *type*('*value*'). For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t159737980000;
  - o1235;
  - f5;
  - p3;

Dynamic language for wordnet

- **N** Individual constants (nominals) - in the system we use decimal numbers for them.
- **O** A set of constants for the features in the objects and their values. They use the schema $type('value')$. For instance pos('n') is such constant.
- A set of relation symbols (hypernym, literal)
- We have 4 types of temporal modifiers - for a fixed timestamp (real time moment), fixed operation moment (model time moment), relative future and relative past like this:
  - t159737980000;
  - o1235;
  - f5;
  - p3;

Atomic formulae

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

Atomic formulae

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

Atomic formulae

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

Atomic formulae

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** ⊆ **For**.

Let q and r be fomulae (queries), R ∈ **R**, **t** ∈ **TM**, then the following are formulae:

- !q
- q & r
- q | r
- q => r
- q <=> r
- <R>q
- [R]q
- ≪ **t** ≫q

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Formulae

- **AtomicFor** $\subseteq$ **For**.

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

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

Relation modifiers

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

Relation modifiers

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

Relation modifiers

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

Semantics

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

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

Semantics

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

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

Semantics

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

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

Semantics

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

- $\mathcal{D}, t, x \Vdash$ $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

Semantics

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

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

Semantics

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

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

Semantics

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

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

Semantics

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

- $\mathcal{D}, t, x \Vdash$ \$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

Semantics

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

- $\mathcal{D}, t, x \Vdash$ \$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

Temporal queries

- $\mathcal{D}, t, x \Vdash$ ≪t159737980000≫q iff $\mathcal{D}, t_0, x \Vdash q$ where t_0 is the nearest previous model moment to this timestamp
- $\mathcal{D}, t, x \Vdash$ ≪p3≫q
- $\mathcal{D}, t, x \Vdash$ ≪f5≫q

Temporal queries

- $\mathcal{D}, t, x \Vdash$ ≪t159737980000≫q iff $\mathcal{D}, t_0, x \Vdash q$ where t_0 is the nearest previous model moment to this timestamp
- $\mathcal{D}, t, x \Vdash$ ≪p3≫q
- $\mathcal{D}, t, x \Vdash$ ≪f5≫q

Temporal queries

- $\mathcal{D}, t, x \Vdash$ ≪t159737980000≫q iff $\mathcal{D}, t_0, x \Vdash q$ where t_0 is the nearest previous model moment to this timestamp
- $\mathcal{D}, t, x \Vdash$ ≪p3≫q
- $\mathcal{D}, t, x \Vdash$ ≪f5≫q

Example queries

- Find the noun synsets that are on top of hyperonymy hierarchy in English:
  `pos('n') & [hypernym]`$\perp$ `& lang('en')`
- Find the synsets that are exactly two levels below the top in the hyperonymy hierarchy:
  `[hypernym][hypernym][hypernym]` $\perp$ `&`
  `<hypernym><hypernym>`$\top$
- Find inconsistency between Bulgarian and English:
  `<ili>(lang('en') & pos('n') & [hypernym][hypernym]`$\perp$ `&`
  `<hypernym>`$\top$`)`
  `& lang('bg') & [hypernym]`$\perp$
- Find the literals that before 3 days were presenting the word 'test' and 2 days later are not:
  `<p3>(word('test') & !<f2>word('test'))`

Example queries

- Find the noun synsets that are on top of hyperonymy hierarchy in English:
  `pos('n') & [hypernym]`$\perp$ `& lang('en')`
- Find the synsets that are exactly two levels below the top in the hyperonymy hierarchy:
  `[hypernym][hypernym][hypernym]` $\perp$ `&`
  `<hypernym><hypernym>`$\top$
- Find inconsistency between Bulgarian and English:
  `<ili>(lang('en') & pos('n') & [hypernym][hypernym]`$\perp$ `&`
  `<hypernym>`$\top$`)`
  `& lang('bg') & [hypernym]`$\perp$
- Find the literals that before 3 days were presenting the word 'test' and 2 days later are not:
  `<p3>(word('test') & !<f2>word('test'))`

Example 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'))`

Example queries

- Find the noun synsets that are on top of hyperonymy hierarchy in English:
  `pos('n') & [hypernym]`$\perp$ `& lang('en')`
- Find the synsets that are exactly two levels below the top in the hyperonymy hierarchy:
  `[hypernym][hypernym][hypernym]` $\perp$ `&`
  `<hypernym><hypernym>`$\top$
- Find inconsistency between Bulgarian and English:
  `<ili>(lang('en') & pos('n') & [hypernym][hypernym]`$\perp$ `&`
  `<hypernym>`$\top$`)`
  `& lang('bg') & [hypernym]`$\perp$
- Find the literals that before 3 days were presenting the word 'test' and 2 days later are not:
  `<p3>(word('test') & !<f2>word('test'))`

GUI

- SPA
- Search panel - word, regex, formula
- 2 modes - Single, Aligned
- Recursive view
- Editing
- Relation wizard

GUI

- SPA
- Search panel - word, regex, formula
- 2 modes - Single, Aligned
- Recursive view
- Editing
- Relation wizard

GUI

- SPA
- Search panel - word, regex, formula
- 2 modes - Single, Aligned
- Recursive view
- Editing
- Relation wizard

GUI

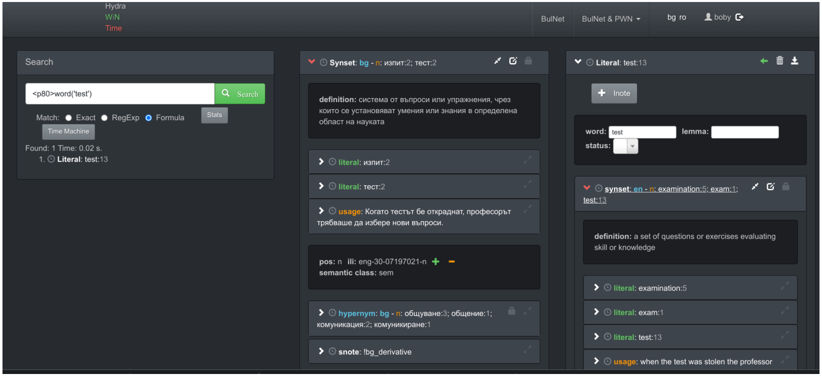
- SPA
- Search panel - word, regex, formula
- 2 modes - Single, Aligned
- Recursive view
- Editing
- Relation wizard

GUI

- SPA
- Search panel - word, regex, formula
- 2 modes - Single, Aligned
- Recursive view
- Editing
- Relation wizard

GUI

- SPA
- Search panel - word, regex, formula
- 2 modes - Single, Aligned
- Recursive view
- Editing
- Relation wizard



Search

<p80>word('test')

Search

Match: Exact RegExp [✓ Formula]

Stats

Time Machine

Found: 1 Time: 0.02 s.

1. **Literal**: test:13

Synset: bg - n: изпит:2; тест:2

definition: система от въпроси или упражнения, чрез които се установяват умения или знания в определена област на науката

literal: изпит:2

literal: тест:2

usage: Когато тестът бе откраднат, професорът трябваше да избере нови въпроси.

pos: n **ili:** eng-30-07197021-n

semantic class: sem

hypernym: bg - n: общуване:3; общение:1; комуникация:2; комуникиране:1

snote: !bg_derivative

Literal: test:13

Inote

word: test **lemma:**

status:

synset: en - n: examination:5; exam:1; test:13

definition: a set of questions or exercises evaluating skill or knowledge

literal: examination:5

literal: exam:1

literal: test:13

usage: when the test was stolen the professor

Figure: Hydra

Tracking and Real-time Updates

- **Tracking User Edits**
  - Automatic history of all editing operations
  - Previous versions always available (no checkout needed)
- **Real-time Updates**
  - Instant propagation via notifications
  - Conflict detection allows data consistency preservation

Tracking and Real-time Updates

- **Tracking User Edits**
  - Automatic history of all editing operations
  - Previous versions always available (no checkout needed)
- **Real-time Updates**
  - Instant propagation via notifications
  - Conflict detection allows data consistency preservation

Tracking and Real-time Updates

- **Tracking User Edits**
  - Automatic history of all editing operations
  - Previous versions always available (no checkout needed)
- **Real-time Updates**
  - Instant propagation via notifications
  - Conflict detection allows data consistency preservation

Tracking and Real-time Updates

- **Tracking User Edits**
  - Automatic history of all editing operations
  - Previous versions always available (no checkout needed)
- **Real-time Updates**
  - Instant propagation via notifications
  - Conflict detection allows data consistency preservation

Tracking and Real-time Updates

- **Tracking User Edits**
  - Automatic history of all editing operations
  - Previous versions always available (no checkout needed)
- **Real-time Updates**
  - Instant propagation via notifications
  - Conflict detection allows data consistency preservation

Tracking and Real-time Updates

- **Tracking User Edits**
  - Automatic history of all editing operations
  - Previous versions always available (no checkout needed)
- **Real-time Updates**
  - Instant propagation via notifications
  - Conflict detection allows data consistency preservation

Implementation

- JS
- Query → SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Implementation

- JS
- Query $\rightarrow$ SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Implementation

- JS
- Query $\rightarrow$ SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Implementation

- JS
- Query $\rightarrow$ SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Implementation

- JS
- Query $\rightarrow$ SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Implementation

- JS
- Query $\rightarrow$ SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Implementation

- JS
- Query $\rightarrow$ SQL (PostgreSQL)
- Every operation generates new record
- Permissions
- REST API
- KnockoutJS
- Bootstrap

Modernization

- Current implementation is outdated (KnockoutJS)
- Modernization plan: rewrite using AI
- Tech Stack:
  - React
  - Next.js

Modernization

- Current implementation is outdated (KnockoutJS)
- Modernization plan: rewrite using AI
- Tech Stack:
  - React
  - Next.js

Modernization

- Current implementation is outdated (KnockoutJS)
- Modernization plan: rewrite using AI
- Tech Stack:
  - React
  - Next.js

Modernization

- Current implementation is outdated (KnockoutJS)
- Modernization plan: rewrite using AI
- Tech Stack:
  - React
  - Next.js

Modernization

- Current implementation is outdated (KnockoutJS)
- Modernization plan: rewrite using AI
- Tech Stack:
  - React
  - Next.js

Deployment and Access

The **Time Flow Hydra** system is deployed and accessible:

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

Conclusion and Future work

- Every touch is stored in the database
- Concurrent access
- Safely cleaning
- More modal operators like Since and Until (Sometime in the past, Sometime in the future)
- Separation
- GUI Assistant for linguists (Predefined queries)

Conclusion and Future work

- Every touch is stored in the database
- Concurrent access
- Safely cleaning
- More modal operators like Since and Until (Sometime in the past, Sometime in the future)
- Separation
- GUI Assistant for linguists (Predefined queries)

Conclusion and Future work

- Every touch is stored in the database
- Concurrent access
- Safely cleaning
- More modal operators like Since and Until (Sometime in the past, Sometime in the future)
- Separation
- GUI Assistant for linguists (Predefined queries)

Conclusion and Future work

- Every touch is stored in the database
- Concurrent access
- Safely cleaning
- More modal operators like Since and Until (Sometime in the past, Sometime in the future)
- Separation
- GUI Assistant for linguists (Predefined queries)

Conclusion and Future work

- Every touch is stored in the database
- Concurrent access
- Safely cleaning
- More modal operators like Since and Until (Sometime in the past, Sometime in the future)
- Separation
- GUI Assistant for linguists (Predefined queries)

Conclusion and Future work

- Every touch is stored in the database
- Concurrent access
- Safely cleaning
- More modal operators like Since and Until (Sometime in the past, Sometime in the future)
- Separation
- GUI Assistant for linguists (Predefined queries)